PHD DISSERTATION

CONIC PROGRAMMING FOR ROUTING AND LOCATION PROBLEMS

Carlos Valverde Martín

Supervisor: Prof. Dr. Justo Puerto

UNIVERSIDAD D SEVILLA

Universidad de Sevilla
Instituto de Matemáticas de la Universidad de Sevilla (IMUS)
Programa de Doctorado en Matemáticas

Sevilla XX.XX.2023

Resumen

Esta tesis doctoral desarrolla nuevos problemas en el campo de los modelos de transporte, con una perspectiva orientada al uso de drones. En este trabajo se definen nuevos modelos de transporte, tomando ciertos elementos de los problemas de localización, los cuales se resuelven con herramientas de la investigación operativa, como la optimización. La tesis se divide en dos partes.

La Parte I se compone de cinco capítulos que abordan diferentes aspectos. El Capítulo 1 introduce el marco teórico que permite comprender los problemas que se desarrollan y cuáles son los últimos avances alcanzados en la literatura. En el Capítulo 2 se establecen qué objetivos se persiguen en este trabajo. El Capítulo 3 presenta los resultados obtenidos hasta la fecha. A continuación, en el Capítulo 4, se lleva a cabo una discusión detallada de dichos resultados. Finalmente, en el Capítulo 5 se presentan las conclusiones generales extraídas de la tesis.

La Parte II también está compuesta por seis capítulos, del 6 al 11. Cada uno de los capítulos en la Parte II puede ser abordado de manera independiente y representa una contribución original de investigación en sí mismo.

En el Capítulo 6, se aborda una extensión del problema del cartero rural que se centra en diseñar rutas que deben visitar diferentes elementos dimensionales en lugar de simplemente aristas. Este problema modela la planificación de rutas para drones u otros vehículos, donde es necesario visitar múltiples ubicaciones geográficas para entregar bienes o servicios, y luego pasar directamente a la siguiente ubicación utilizando desplazamientos en línea recta. En este capítulo, se presentan dos familias de formulaciones de programación matemática. La primera familia se basa en un modelo por etapas y captura diversas características con aplicaciones prácticas, pero tiene la desventaja de utilizar índices de tres variables. La segunda familia de formulaciones prescinde de estas etapas y utiliza propiedades de conectividad para garantizar la correcta definición de las rutas. Estas formulaciones se comparan utilizando instancias con diferentes formas, como conjuntos representables con conos de segundo orden (SOC), entornos poliédricos y poligonales. Los resultados computacionales presentados en este trabajo demuestran que los modelos son efectivos y que las formulaciones pueden resolver de manera óptima instancias de tamaño mediano, similares a otros problemas combinatorios con entornos que han sido estudiados en la literatura. Para resolver instancias más grandes, también se presenta un algoritmo heurístico que consta de dos fases: agrupación y un metaheurístico de búsqueda local. Este algoritmo ofrece buenos resultados al generar soluciones factibles cercanas al óptimo que, además, puede utilizarse para inicializar los solvers con dichas soluciones.

El Capítulo 7 se enfoca en dos problemas distintos de diseño de rutas en el espacio continuo que involucran entornos y barreras: el problema del camino más corto y el problema del viajante de comercio con entornos y barreras. La presencia de estos dos elementos, entornos y barreras, hace que los problemas sean más desafiantes en comparación con sus contrapartes estándar. Al combinar ambos aspectos, surge un nuevo problema que hasta la fecha no ha sido abordado. No obstante, este problema tiene aplicaciones relevantes en actividades de inspección y vigilancia, así como en la industria de reparto, especialmente cuando existe una demanda uniformemente distribuida en ciertas regiones. En el capítulo se presentan formulaciones de programación matemática para ambos problemas, asumiendo barreras lineales y entornos representables con conos de segundo orden. Estos supuestos conducen a formulaciones enteras mixtas con conos de segundo orden , las cuales son sometidas a preprocesamiento y se refuerzan mediante desigualdades válidas. Además, se llevan a cabo experimentos computacionales que demuestran que el método exacto puede resolver instancias con 75 entornos y un rango de 125 a 145 barreras.

El Capítulo 8 aborda problemas de localización de instalaciones en un espacio continuo con vecinos y barreras. Específicamente, se analiza el problema de la p-mediana con vecinos y barreras lineales en dos situaciones diferentes. Como primer bloque de construcción, se aborda el problema asumiendo que los entornos no son visibles entre sí y, por lo tanto, no existen rutas rectilíneas que unan dos entornos sin cruzar barreras. Bajo esta hipótesis, se obtiene una formulación válida de programación lineal entera mixta. Al eliminar esa hipótesis, se obtiene el problema más general y realista, pero con el inconveniente de ser más desafiante. Adaptando los elementos de la primera formulación, también se desarrolla otra formulación válida de programación bilineal entera mixta. Ambas formulaciones manejan barreras lineales y entornos que son representables con conos de segundo orden, los cuales se preprocesan y fortalecen con desigualdades válidas. Estas formulaciones de programación matemática también son fundamentales para generar un algoritmo matheurístico adaptado que proporciona soluciones de buena calidad para ambos problemas en un tiempo de cómputo corto. El capítulo también detalla una amplia experiencia computacional que demuestra que los enfoques exactos y heurísticos son útiles: el enfoque exacto puede resolver instancias con hasta 50 entornos y diferentes números de barreras en una hora de tiempo de CPU, mientras que el matheurístico siempre devuelve excelentes soluciones factibles en menos de 100 segundos.

El Capítulo 9 se centra en mejorar la planificación de rutas utilizando drones. Se examina la coordinación entre un nave principal y un dron para encontrar las rutas más eficientes que deben seguir para visitar diferentes objetivos representados como grafos. El objetivo es minimizar la distancia total recorrida por ambos vehículos, al mismo tiempo que se cumplen los requisitos de visitas a los objetivos en términos de porcentajes. Se analizan distintos enfoques según las suposiciones realizadas sobre la ruta de la nave principal: i) la nave se puede mover en un plano continuo (plano euclídeo), ii) en una poligonal, o iii) en un grafo general. En todos los casos, se desarrollan formulaciones exactas mediante modelos de programación cónica entera mixta de segundo orden que se comparan en un conjunto de pruebas para evaluar su rendimiento. La complejidad de estos métodos exactos dificulta

la búsqueda de soluciones óptimas en un tiempo de cálculo reducido. Por lo tanto, además de las formulaciones exactas, también presentamos un procedimiento matheurístico que permite obtener soluciones de alta calidad en un tiempo razonable. Los experimentos computacionales demuestran la utilidad de nuestros métodos en diferentes escenarios.

En el Capítulo 10 se examina un modelo que combina el movimiento de un dron con cierta autonomía que puede visitar múltiples puntos, junto con un vehículo base que puede moverse libremente en el espacio continuo. Este vehículo desempeña el papel de cargar la batería del dron, mientras que el dron se encarga de visitar diferentes objetivos, representados por puntos o poligonales. En el caso de las poligonales, se establece el requisito de que el dron atraviese una fracción específica de sus longitudes, que representan actividades de vigilancia o inspección. El objetivo principal del problema consiste en minimizar la distancia total ponderada recorrida por ambos vehículos. Para abordar este problema, se desarrolla y mejora una formulación de programación cónica entera mixta de segundo orden, utilizando desigualdades válidas y proporcionando límites adecuados para las M grandes que aparecen en el modelo. Además, se propone una estrategia matemática refinada que permite obtener soluciones de calidad en un tiempo de cálculo reducido. La calidad de las soluciones generadas por ambos enfoques se compara y analiza exhaustivamente utilizando un conjunto aleatoria de instancias con diferentes números y formas de objetivos, lo que demuestra la utilidad de nuestro enfoque y su aplicabilidad en diversas situaciones.

En el Capítulo 11 se analizan los desafíos de optimización asociados a la coordinación de un sistema compuesto por un vehículo principal y una flota de drones. Cada dron es lanzado desde el vehículo principal para llevar a cabo una tarea específica. Una vez completada la tarea, los drones regresan al vehículo principal para recargar sus baterías y prepararse para una nueva tarea. Estas tareas implican visitar parcialmente grafos con una longitud determinada, con el propósito de brindar servicios o realizar actividades de vigilancia e inspección. El objetivo principal consiste en minimizar el tiempo total de los desplazamientos realizados por el vehículo principal, al mismo tiempo que se cumplen ciertos requisitos en términos de porcentajes de visitas a los grafos objetivo. Para abordar este problema, se desarrollan formulaciones exactas utilizando programas de conos de segundo orden con variables enteras, los cuales son comparados en un conjunto de pruebas para evaluar su rendimiento. Además, se presenta un algoritmo matheurístico que genera soluciones razonables. Los experimentos computacionales demuestran la utilidad de esta metodología en diversos escenarios.

Abstract

This thesis develops new problems in the field of transportation, with a perspective orientated to the use of drones. In this work, new transportation models are defined taking certain elements from localisation theory, which are solved with elements from operations research, such as optimisation. The thesis is divided into two parts.

Part I consists of five chapters dealing with different aspects. Chapter 1 introduces the theoretical framework that allows us to understand the problems that are developed and the latest advances that have been achieved in the literature. Chapter 2 sets out the objectives of this work. Chapter 3 presents the results obtained to date. This is followed by a detailed discussion of these results in Chapter 4. Finally, Chapter 5 presents the general conclusions drawn from the thesis.

Part II is also composed of six chapters, from Chapter 6 to Chapter 11. Each of the chapters in Part II can be approached independently and represents an original research contribution in itself.

In Chapter 6, an extension of the rural postman problem is addressed that focuses on designing routes that must visit different dimensional elements rather than simply edges. This problem models route planning for drones or other vehicles, where it is necessary to visit multiple geographic locations to deliver goods or services and then move directly to the next location using straight-line travel. In this chapter, two families of mathematical programming formulations are presented. The first family is based on a model by stages and captures several features with practical applications, but has the disadvantage of using three-variable indices. The second family of formulations avoids stages and uses connectivity properties to ensure the correct definition of routes. These formulations are compared using instances with different shapes, such as second-order cone (SOC) representable sets, polyhedral, and polygonal neighbourhoods. The computational results presented in this paper demonstrate that the models are effective and that the formulations can optimally solve medium-sized instances, similar to other combinatorial problems with neighbourhoods that have been studied in the literature. To solve larger instances, a heuristic algorithm is also presented that consists of two phases: clustering and a variable neighbourhood search metaheuristic. This algorithm gives good results by generating feasible solutions close to the optimum, which can also be used to initialise the solvers with these solutions.

Chapter 7 focuses on two distinct route design problems in continuous space involving neighbourhoods and barriers: the shortest path problem and the travelling salesman problem with neighbourhoods and barriers. The presence of these two elements, neigh-

bourhoods and barriers, makes the problems more challenging compared to their standard counterparts. When both aspects are combined, a new problem arises that has not been addressed to date. However, this problem has relevant applications in inspection and surveillance activities, as well as in the delivery industry, especially when there is a uniformly distributed demand in certain regions. The chapter presents mathematical programming formulations for both problems, assuming linear barriers and second-order cone (SOC) representable neighbourhoods. These assumptions lead to mixed-integer formulations with second-order cones, which are preprocessed and strengthened by valid inequalities. In addition, computational experiments are carried out that show that the exact method can solve instances with 75 neighbourhoods and a range of 125 to 145 barriers.

Chapter 8 addresses facility location problems in a continuous space with neighbourhoods and barriers. Specifically, the p-median problem with linear neighbourhoods and barriers is analysed in two different situations. As a first building block, the problem is approached assuming that the neighbourhoods are not visible to each other, and therefore there are no rectilinear routes linking two neighbourhoods without crossing barriers. Under this assumption, a valid mixed-integer linear programming formulation is obtained. Removing that assumption yields the more general and realistic problem, but with the drawback of being more challenging. By adapting the elements of the first formulation, another valid mixed-integer bilinear programming formulation is also developed. Both formulations handle linear barriers and neighbourhoods that are representable with second-order cones (SOCs), which are preprocessed and reinforced with valid inequalities. These mathematical programming formulations are also instrumental in generating an adapted matheuristic algorithm that provides good quality solutions for both problems in a short computational time. The chapter also details extensive computational experience that demonstrates that exact and heuristic approaches are useful: the exact approach can solve instances with up to 50 neighbourhoods and different numbers of barriers in one hour of CPU time, while the matheuristic approach always returns excellent feasible solutions in less than 100 seconds.

Chapter 9 focuses on improving route planning using drones. It examines the coordination between a mothership and a drone to find the most efficient routes to follow to visit different targets represented as graphs. The aim is to minimise the total distance travelled by both vehicles, while meeting the target visit requirements in terms of percentages. Different approaches are analysed depending on the assumptions made about the mothership route: i) the mothership can move in a continuous plane (Euclidean plane), ii) in a polygonal, or iii) in a general graph. In all cases, exact formulations are developed using second-order mixed-integer conic programming models that are compared on a test set to evaluate their performance. The complexity of these exact methods makes it difficult to find optimal solutions in a short computational time. Therefore, in addition to exact formulations, a matheuristic procedure is also presented that allows us to obtain high quality solutions in a reasonable time. Computational experiments demonstrate the usefulness of our methods in different scenarios.

Chapter 10 examines a model that combines the movement of a drone with a limited endurance that can visit multiple points, together with a base vehicle that can move freely in continuous space. This vehicle plays the role of charging the battery of the drone, while the drone is in charge of visiting different targets, represented by points or polygonals. In the case of polygonals, there is a requirement for the drone to traverse a specific fraction of their lengths, representing surveillance or inspection activities. The main objective of the problem is to minimise the total weighted distance travelled by both vehicles. To address this problem, a second-order mixed-integer conic programming formulation is developed and improved, using valid inequalities and providing appropriate bounds for the bigM appearing in the model. In addition, a refined mathematical strategy is proposed that allows for the generation of quality solutions in a reduced computational time. The quality of the solutions generated by both approaches is thoroughly compared and analysed using a random set of instances with different numbers and forms of targets, which demonstrates the usefulness of our approach and its applicability in various situations.

Chapter 11 discusses the optimisation challenges associated with coordinating a system composed of a mothership and a fleet of drones. Each drone is launched from the mothership to perform a specific task. Once the task is completed, the drones return to the main vehicle to recharge their batteries and prepare for a new task. These tasks involve partially visiting networks of a certain length with the purpose of providing services or performing surveillance and inspection activities. The main objective is to minimise the total travel time of the mothership while meeting certain requirements in terms of the percentage of visits to the target networks. To address this problem, exact formulations are developed using second-order cone programmes with integer variables, which are compared in a test set to evaluate their performance. In addition, a matheuristic algorithm that generates reasonable solutions is presented. Computational experiments demonstrate the usefulness of this methodology in various scenarios.

Agradecimientos

Con estas palabras finaliza una etapa muy importante en mi vida. Una etapa que comenzó con muchas dudas y miedos por saber lo que deparaba el futuro, pero que ha terminado con la satisfacción de haber disfrutado de los mejores años a nivel personal. Puedo decir que he tenido la suerte de poder dedicarme a lo que ha sido siempre mi pasión: las matemáticas.

Esta etapa no se puede explicar sin mi director, mi padre académico y mi guía, Justo. Gracias a tus consejos, tu paciencia y todo tu trabajo, he podido desarrollarme en todos los aspectos de esta etapa. Gracias por confiar en mí y brindarme la oportunidad de poder dedicarme profesionalmente a las matemáticas, algo que llevaré siempre por bandera. Aprender de ti ha sido todo un placer. Espero poder seguir teniendo esta suerte y podamos seguir aprendiendo juntos en el futuro.

Este trabajo tampoco se puede entender sin la ayuda de mi compañera Lavinia. Gracias, también, por tu esfuerzo y dedicación. Que este tándem nos permita lograr conjuntamente los objetivos que nos queramos proponer.

Por otro lado, el apoyo incondicional que me ha dado mi familia en todas mis decisiones ha sido muy importante para mí. Gracias por quererme y cuidarme tanto. Gracias por el esfuerzo del día a día que me ha permitido haber estudiado lo que más me gustaba. Gracias por hacerme la vida más sencilla. Gracias a mi hermano por traernos al mundo la luna más bonita que existe. Es una suerte teneros a todos en mi día a día. No me olvido tampoco de aquellas personas que nos han abandonado, y que nos están cuidando allí donde estén.

Otro grupo de personas del que tampoco me olvido son mis amigos. Mi fuerza, vitalidad y felicidad no existiría sin ellos. Gracias a Abel por ser mi amigo, mi compañero, mi consejero, mi cantante. Espero que todas las vivencias que hemos tenido juntos en estos años no cesen y sean cada vez más numerosas. Gracias a Ramón por ser un hermano para mí, por estar atento a todo y estar ahí para todo lo que te he necesitado; me debes una partida a algún videojuego. Gracias, también, a dos personas que han entrado en mi vida durante estos años y que me han acompañado y seguramente, me acompañarán para el resto de ella. Gracias a Cristina por mimarme, consentirme y hacerme disfrutar de la vida como no lo había hecho antes. Gracias a Paco, mi futbolero y fiestero, por todos los esfuerzos que has tenido que hacer para compartir momentos conmigo. Gracias, también, a Gonzalo y a Antonio, por todos los fines de comidas, cenas y planes que me habéis propuesto. Quiero que todos vosotros continuéis en las nuevas etapas de mi vida, siendo tal como sois.

También quiero agradecer a todos mis compañeros del IMUS y compañeros del grupo

de investigación. Habéis conseguido hacerme sentir como uno más de la familia a pesar de haber estado asintóticamente en el edificio. Siempre habéis contado conmigo para todo. Gracias a los Albertos y a Gabri, compañeros del despacho 9. Me habéis ayudado siempre en todo lo que he necesitado. En particular, quiero dedicarle unas palabras a mi amigo y compañero Moisés. Siempre has estado ahí para todo. Siento no poder haberte ayudado a superar tus dificultades y poder estar celebrando juntos nuestras tesis, algo que todos hubiéramos querido. Sin embargo, estoy seguro de que estarás feliz viéndonos conseguir lo que tú siempre has querido conseguir...

Mi etapa como doctorando termina, pero empieza otra aún más ilusionante, llena de gente maravillosa que me acompaña y gente nueva que aportará más valor a mi vida. Una etapa en comunión con las matemáticas y la enseñanza. No puedo esperar más para pasar al siguiente nivel. Que comience el juego.

Contents

Chapter 1

Introduction

Operations Research (OR) plays an important role in contemporary decision-making processes in various domains. It provides a systematic and analytical approach to solving complex problems, optimising processes, and making informed decisions. OR offers value in resource allocation, cost minimisation, risk assessment, and performance improvement, leading to enhanced efficiency and productivity. OR techniques have been widely applied in supply chain management, transportation, healthcare, finance, and other industries. It has resulted in cost savings, improved quality of services, and better utilisation of resources. Furthermore, it enables organisations to address real-world challenges and make data-driven decisions, contributing to increased competitiveness and sustainable development (Augier and Teece, 2021).

Conic optimisation is a powerful mathematical programming framework that has gained significant attention in recent years. It extends traditional linear programming by incorporating second-order cones, semidefinite cones, and other convex cones into the optimisation formulation. This extension allows for the efficient modelling and solution of complex optimisation problems with nonlinear and nonconvex constraints. Conic programming has found applications in various fields such as portfolio optimisation, control theory, machine learning, and logistics (Lobo et al., 1998).

This thesis explores the applications of conic optimisation in the context of location and transportation problems. Location models are designed to determine the optimal placement of various services, taking into account factors such as user distance and cost considerations (nic, 2005). Location problems often arise in various fields, including logistics and supply chain management, urban planning, facility management, and telecommunications (Drezner and Hamacher, 2004). On the other hand, transportation models focus on finding the most efficient way to transport resources from a source to multiple destinations while minimising costs (Ahuja et al., 1993; Gutin and Punnen, 2006). In this case, this thesis specifically concentrates on drone transport models because of their immense applicability and promising future potential. Drones have revolutionised the field by introducing a level of flexibility and freedom in route selection that was previously unattainable (Otto et al., 2018). This paradigm shift has opened up new possibilities for employing continuous optimisation techniques in problems that traditionally relied on discrete optimisation methods. This thesis seeks to integrate location and transportation models, thereby paving the way for solving more realistic problems. By combining the strengths of location and transportation models and leveraging the flexibility of drone transport, researchers hope to address real-world challenges more effectively.

Combinatorial optimisation problems, including location and transportation problems, have been extensively studied in the specialised literature for several decades (Papadimitriou and Steiglitz, 1998). However, the emergence of drones has caused a significant shift in the way these problems are approached. Previously, discrete optimisation techniques were prevalent due to the restrictions imposed on traditional transportation methods. However, drones have alleviated many of these restrictions, making continuous optimisation approaches, such as conic optimisation, more viable and advantageous. In this thesis, new problems of combinatorial optimisation are studied by examining their key properties,

solution schemes, and contributions. In this introduction, the necessary background and initial motivation are presented. The aim is to build upon existing knowledge and address specific challenges within the context of these problems and contribute to the development of the field.

1.1 Conic programming

Let $\mathcal{K}$ be a convex, pointed, closed, and with a non-empty interior cone, a conic program on $\mathcal{K}$ is the optimisation program:

$$\min_x \quad c^t x \tag{CP}$$
$$\text{s.t.} \quad b \leq_{\mathcal{K}} Ax \Leftrightarrow Ax - b \in \mathcal{K}, \tag{1.1a}$$

where (c, A, b) is the tuple that represents the data values. Note that (CP) optimises a linear function over a convex set and thus is a convex program. It is possible to represent every convex program as a conic one. In addition, a huge spectrum of applications are covered using only three families of cones: nonnegative orthants, finite products of Lorentz cones, and direct products of semidefinite cones. The advantages of these families are the fully symmetric duality, heavily utilised by solution algorithms, the existence of polynomial-time interior point methods and its extremely powerful expressive abilities (Nemirovski, 2006).

Duality

In mathematical programming, duality is introduced as having a systematic way of taking good lower bounds of the optimal value over a minimisation problem. For the conic program in $\mathcal{K}$ described in (CP), its dual problem is defined as follows.

$$\min \quad b^t u \tag{D}$$
$$\text{s.t.} \quad A^t u = c, \tag{1.2a}$$
$$0 \leq_{\mathcal{K}^*} u \Leftrightarrow u \in \mathcal{K}^*, \tag{1.2b}$$

where $\mathcal{K}^* = \{u : u^t \xi \geq 0, \forall \xi \in \mathcal{K}\}$ is the cone dual of $\mathcal{K}$. The resulting problem is again a conic problem, and assuming the columns of A to be linearly independent, the duality is fully symmetric.

The power of duality in conic programming is summarised in the following theorem (see, e.g. Nesterov and Nemirovski (1992), Nesterov and Nemirovski (1994), Alizadeh (1995)):

Theorem 1 *Assuming A in* (CP) *is a full column rank matrix, it verifies:*

1. (D) *is symmetric: it is a conic program, and its conic dual is equivalent to* (CP).

2. ***Weak Duality:*** *The optimal value of the dual is a lower bound of the optimal value of the primal, i.e., if* $\bar{x}$ *and* $\bar{u}$ *are the optimal solution of the primal and dual, respectively:*

$$c^t\bar{x} \geq b^t\bar{u}.$$

3. ***Strong Duality:*** *If one of the programs* (CP) *or* (D) *is bounded and strictly feasible, the other is solvable and the optimal value coincide:*

$$c^t\bar{x} = b^t\bar{u}.$$

4. ***Optimality conditions:*** *If, additionally,* (CP) *and* (D) *are both strictly feasible, then a feasible solution* (x, u) *is optimal if and only if* $u^t[Ax - b] = 0$.

Interior point methods

An interior penalty scheme to solve the conic program (CP), introduced in Fiacco and McCormick (1990), consists of:

1. Defining a smooth and strictly convex function F that converges to ∞ for every sequence $\{x_i^k\}$ in the interior of $\mathcal{K}$ convergent to its boundary.

2. Associating (CP) with a parametric optimisation problem:

$$x_*(s) = \arg\min_{x \in \mathcal{K}} sc^t x + F(x). \tag{1.3}$$

Under mild assumptions, $x_*(s)$ is well defined and converges to the optimal of (CP), when $s \to +\infty$. The path traced using this parametric optimisation problem is described below. Given (x, s) with x close to $x_*(s)$:

1. Replace s with $s^+ > s$.

2. Minimise $s^+c^tx + F(x)$ by the Newton method, using x as the starting point, until a point x^+ is close to $x_*(s^+)$ is built.

3. Replace (x, s) with (x^+, s^+) and iterate.

The advantages of interior point methods in conic programming are that every cone $\mathcal{K}$ admits a *self-concordant* barrier, and the path-following scheme presented above becomes polynomial when the barrier is self-concordant. Hence, every conic programming problem admits a polynomial interior point method (see Nesterov and Nemirovski (1994) for more details). Essentially, all convex programming is within the grasp of polynomial time interior point methods (Nemirovski, 2006).

In the following, two of the most important families of cones and its applications are introduced.

1.1.1 Linear programming

The nonnegative orthant of dimension n is described as:

$$\mathbb{R}^n_+ = \{x = (x_1, \ldots, x_n) \in \mathbb{R}^n : x_i \geq 0, i = 1, \ldots, n\}.$$

Then a linear programming program (LP) is defined as

$$\begin{aligned} \min \quad & c^t x \\ \text{s.t.} \quad & 0 \leq Ax - b \Leftrightarrow Ax - b \in \mathbb{R}^n_+. \end{aligned} \tag{LP}$$

Linear programming was invented around 1947 by George Dantzig. LP was the starting point to the development of Mathematical Programming, in which a general function is optimised under different assumptions. It allowed modelling and processing logistic, routing and planning problems. In addition, the great performance of the Simplex algorithm for LP together with the improvement of the computational devices developed the theory and algorithms of LP programs. However, it took over 30 years to find a polynomial-time algorithm to solve a LP program (Khachiyan, 1980).

Linear programming has been applied to a wide spectrum of problems, like regression in statistics or portfolio selection or option pricing in financials. However, the most important special class of linear programming is formed by network flow problems. They consist of minimising the transportation cost of moving some commodities through a network to meet demands at various locations given sources of commodities at other locations. The most important applications of network flow problems are the transportation problem, the assignment problem, the shortest-path problem, and the maximum-flow problem. This class of problems also includes facility location, resource management, financial planning, and others.

1.1.2 Second-order cone programming

Let $\|\cdot\|_2$ denote the standard Euclidean norm derived from the dot product in $\mathbb{R}^n$, i.e., $\|u\|_2 = (u^t u)^{1/2}$. The second-order cone (or Lorentz cone) of dimension $k+1$ is defined as:

$$\mathcal{L}^{k+1} = \{(x, y) \in \mathbb{R}^k \times \mathbb{R} : \|x\|_2 \leq y\}.$$

Then the second-order conic program (SOCP) is defined as

$$\begin{aligned} \min \quad & c^t x \\ \text{s.t.} \quad & \|A_i x - b_i\| \leq c_i^t x - d_i \Leftrightarrow (A_i x - b_i, c_i^t x - d_i) \in \mathcal{L}^{k_i+1}, \quad i = 1, \ldots, m, \end{aligned} \tag{SOCP}$$

where $A_i \in \mathbb{R}^{k_i \times n}, b_i \in \mathbb{R}^{k_i}, c_i \in \mathbb{R}^n$ and $d_i \in \mathbb{R}$ are the problem parameters.

Second-order cone programming can be used to represent common convex constraints. If $A_i \equiv 0$ and $b_i \equiv 0$, for some $i \in 1, \ldots, m$, the corresponding second-order cone constraint

is reduced to a linear. On the other hand, if $c_i \equiv 0$ and $d_i \leq 0$, the constraint is reduced to a convex quadratic constraint.

1.2 The Travelling Salesman Problem

The travelling salesman problem (TSP) consists of finding a tour who starts from a home location, visits a prescribed set of points, and returns to the original location in such a way that the total distance travelled is minimum and each point is visited exactly once.

Let $G = (V, E)$ be a complete graph, where $V = \{1, \ldots, n\}$ is the set of nodes that represent the cities and E, the set of edges that join each pair of cities. Each edge $e \in E$ has a cost associated c_e. Let $C = (c_{ij})_{n \times n}$ denote the cost matrix, where c_{ij} corresponds to the cost of the edges that join cities i and j in G.

Depending on the nature of the cost matrix, the TSP can be classified into two classes. If C is symmetric, then the TSP is called the symmetric travelling salesman problem (STSP). If C is not symmetric, then it is called the asymmetric travelling salesman (ATSP).

The classical integer formulation of the ATSP is based on the assignment problem described as follows:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1, && j = 1, \ldots, n, \\
& \sum_{j=1}^{n} x_{ij} = 1, && i = 1, \ldots, n, \\
& x_{ij} \in \{0, 1\},
\end{aligned}$$

where the binary variable x_{ij} indicates if the edge (ij) is used in the tour. Note that the edges chosen in the solution of this problem represent a tour or a collection of subtours in G. Hence, additional restrictions are included to avoid the existence of subtours. The two most well-known ways to avoid them are:

- **Clique Packing Constraints:** Introduced by Dantzig et al. (1954). There are an exponential number which are given by

$$\sum_{i \in Q}\sum_{j \in Q} x_{ij} \leq |Q| - 1, \qquad \emptyset \neq Q \subset \{2, 3, \ldots, n\}. \tag{1.4}$$

- **MTZ Constraints:** Miller et al. (1960) proposed a compact representation using $(n-1)^2$ additional constraints and $n-1$ additional continuous variables u_i that are described as

$$u_i - u_j + (n-1)x_{ij} \leq n - 2, \qquad i, j = 2, 3, \ldots, n. \tag{1.5}$$

The formulation of the TSP can be stated as:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} && && \text{(TSP)} \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1, && j = 1,\dots,n, \\ & \sum_{j=1}^{n} x_{ij} = 1, && i = 1,\dots,n, \\ & \text{(1.4) or (1.5)}, \\ & x_{ij} \in \{0,1\}. \end{aligned}$$

In the literature, some variations of the TSP that have been studied are the generalised TSP (Laporte and Nobert, 1983), the time dependent TSP (Fox et al., 1980), the orienteering problem (Golden et al., 1987), or the angle TSP (Aggarwal et al., 2000). Other extensions that could be discussed, depending on various application scenarios, include the travelling salesman location problem (Burness and White, 1976), k-best TSP (van der Poort et al., 1999) or the minmax TSP (França et al., 1995). For a detailed survey, the reader is referred to Gutin and Punnen (2006).

The TSP with Neighbourhoods (TSPN), introduced by Arkin and Hassin (1994), has played a special role in this thesis. In this extension, each point to be visited is associated with a region, denoted as neighbourhood, that contains it. The goal is to determine the shortest tour that visits all nodes, where a node is visited when the tour traverses or reaches the region associated with the node. Several real-world problems are modelled by the (TSPN), including metre reading problems (Shuttleworth et al., 2008) and localisation, monitoring, and diagnostic reconnaissance problems (Poikonen et al., 2017; Di Placido et al., 2022).

1.3 The Mothership and Drone Routing Problem

In Poikonen and Golden (2020), the mothership and drone routing problem (MDRP) is introduced. The MDRP considers the routing of a two-vehicle tandem. A larger vehicle is called the mothership and a smaller vehicle is called the drone. The mothership and the drone begin at a starting location, denoted *orig*. The drone is launched from the mothership to visit, in each operation, a target $v_i \in V$ and then returns to the mothership. When all nodes are visited, both vehicles must return to a final location, denoted *dest*. The MDRP assumes that:

- Both vehicles are capable of moving freely in $\mathbb{R}^2$. There are no obstacles to prevent mothership and drone to travel in straight-line segments.

- Both vehicles are not required to arrive at a rendezvous location at the same time; the drone can wait for the mothership to be retrieved.

- The drone cannot be separated from the mothership for more than N^d time units due to limited battery endurance.

- The mothership has a maximum unit speed; the drone has a maximum speed of $\nu_D > 1$.

- The drone can not visit multiple targets consecutively.

The MDRP is a generalisation of the Euclidean travelling salesman problem. The aim is to find a path of minimum duration that begins at *orig* and ends at *dest* and where every $v_i \in V$ is visited by the drone.

A formulation based on Poikonen and Golden (2020) is stated. Let us denote $\mathcal{O} = \{0, \ldots, n+1\}$ the set of operations that the mothership and the drone have to perform. An operation is termed as the process in which the mothership launches a drone from a take-off location, denoted by x_L^o and then returns to a rendezvous location x_R^o. To take into account the different times among the decision variables of the model, same continuous variables are defined:

- $time_L^{io}$: time taken by the drone to go from the launching point on the mothership to the target point v_i associated with the operation o.

- $time_R^{io}$: time taken by the drone from the target point v_i to the rendezvous point associated with the operation o.

- $time_{LR}^{o}$: time spent by the mothership from the launching point to the rendezvous point at operation o.

- $time_{RL}^{o}$: time spent by the mothership and drone from the rendezvous point at operation o to the launching point at the operation $o+1$.

To include the definition of the routes followed by the drone in the mathematical programming formulation, the optimal assignment of each target v_i to each operation o is required. The binary variable χ^{io} attains value one if the drone visits the target v_i in the operation o.

The goal of the MDRP is to find a feasible solution that minimises the total time spent by the mothership and the drone.

$$\min \quad \sum_{o=1}^{n} time_{LR}^{o} + \sum_{o=0}^{n} time_{RL}^{o} \tag{MDRP}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \chi^{io} = 1, \qquad o = 1, \ldots, n, \tag{1.6a}$$

$$\sum_{o=1}^{n} \chi^{io} = 1, \qquad i = 1, \ldots, n, \tag{1.6b}$$

$$\frac{1}{\nu_D} \|x_L^o - v_i\| \leq time_L^{io}, \qquad i, o = 1, \ldots, n, \tag{1.6c}$$

$$\frac{1}{\nu_D} \|v_i - x_R^o\| \leq time_R^{io}, \qquad i, o = 1, \ldots, n, \tag{1.6d}$$

$$\|x_L^o - x_R^o\| \leq time_{LR}^{o}, \qquad o = 1, \ldots, n, \tag{1.6e}$$

$$\|x_R^o - x_L^{o+1}\| \leq time_{RL}^{o}, \qquad o = 0, \ldots, n, \tag{1.6f}$$

$$\sum_{i=1}^{n} \chi^{io} \left(time_L^{io} + time_R^{io} \right) \leq time_{LR}^{o}, \qquad o = 1, \ldots, n, \tag{1.6g}$$

$$time_{LR}^{o} \leq N^d, \qquad o = 1, \ldots, n, \tag{1.6h}$$

$$x_L^0 = orig, \tag{1.6i}$$

$$x_R^0 = orig, \tag{1.6j}$$

$$x_L^{n+1} = dest, \tag{1.6k}$$

$$x_R^{n+1} = dest. \tag{1.6l}$$

The objective function sets the duration of the tour as the sum of the times during which the mothership and drone are operating. Constraints (1.6a) and (1.6b) ensure that the drone visits only one node in each operation. Constraints (1.6c) state that $time_L^{io}$ is at least the time that the drone spends since it is launched at x_L^o from the mothership until it visits the node v_i. Constraints (1.6d) ensure that $time_R^{io}$ is at least the time it spends visiting node v_i until it is retrieved in x_R^o, respectively. The restrictions (1.6e) and (1.6f) represent the time spent by the mothership on each operation. Constraints (1.6g) state that the time that the drone spends doing the operation o is less than or equal to the time that the mothership needs to move from the launching point to the rendezvous point during this operation. Constraints (1.6h) explain that the time spent by the drone to visit the node v_i must not exceed the time endurance N^d. Constraints (1.6i) through (1.6l) establish the origin and destination of the path.

The relationship between the TSP and the MDRP is as follows. If $obj(TSP)$ and $obj(MDRP)$ denote the optimal objective value of the TSP and the MDRP for the set of locations $V \cup \{orig\}$, respectively. Then

$$\frac{1}{\nu_D} obj(TSP) \leq obj(MDRP) \leq obj(TSP).$$

In Poikonen and Golden (2020), the authors also define a more general case in which the drone can visit more than one target in the same operation. They proposed a branch-

and-bound algorithm capable of solving small-sized instances. They describe heuristics based on greedy approaches and local search strategies for larger instances.

The MDRP is a variant of the flying sidekick TSP, introduced by Murray and Chu (2015). They were the first authors to introduce the concept of truck-drone systems in which each node is served by either a truck or a drone. It assumes that the drone is only allowed to depart from and return to the truck at a node. This variant is also known as the TSP with drone (TSP-D), considered in Agatz et al. (2018). This problem has been extended to the case of a single truck with multiple drones in Murray and Raj (2020). An excellent survey of truck-drone systems from an operational research perspective is elaborated in Boysen et al. (2021).

1.4 The Mixed Windy Rural Postman Problem

One of the most important applications of the travelling salesman problem is the possibility of solving arc routing problems (ARPs), such as the mixed windy rural postman problem (MWRPP), which is stated in the following.

Let $G = (V, A \cup E)$ be a mixed graph, where A are arcs and E are edges. The MWRPP aims to find a minimum cost closed walk on G that contains all arcs in a subset $A' \subset A$ and all edges in $E' \subset E$. Laporte (1997) describes a transformation into a TSP that associates each arc or edge with one vertex in a graph D whose edge costs are associated with the shortest paths in the original graph G.

This problem extends a family of arc routing problems including the mixed Chinese postman problem (Papadimitriou, 1976), the windy postman problem (Minieka, 1979), the windy rural postman problem (Benavent et al., 2007), or the stacker crane problem (Frederickson et al., 1976). Corberán and Laporte (2015) provides a thorough discussion of arc routing problems and its applications.

An special case of the MWRPP is the rural postman problem (RPP), in which only edges are considered ($A = \emptyset$). In Ghiani and Laporte (2015), some mathematical formulations for the RPP are presented followed by exact algorithms and improvement heuristics. Based on this problem, Garfinkel and Webb (1999) introduced the crossing postman problem (XPP), which relaxes the RPP to the case in which it is allowed to leave the edges of the network and cross from one edge to another at points other than the original vertices. The authors proved that the XPP is NP-hard and presented a number of results that allow certain non-required edges to be eliminated without sacrificing the optimal solutions. In Campbell et al. (2018), the Drone Rural Postman Problem (Drone RPP) extended the XPP for curved lines and developed a solution based on the undirected RPP with polygonal chains that approximates the curved lines.

1.5 The discrete p-median problem

The p-median, one of the most studied discrete facility location problems, concerns the optimal placement of one or several new facilities/plants to satisfy the demands of customers. Here, the positions of both the customers and the potential new facilities are part

of the input, as well as the travel costs between them. It was introduced by Hakimi (1965) as a generalisation of the concept of median. The p-median problem is concerned with the location of p facilities within a given region to minimise the total distance between the facilities and a set of demand points.

Let I be the set of potential facility locations and J the set of demand points. The cost or distance between a demand point $i \in I$ and a facility location $j \in J$ is represented by c_{ij}. The classical formulation of the p-median is described as follows:

$$
\begin{aligned}
\min \quad & \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} && && (p\text{-median}) \\
\text{s.t.} \quad & \sum_{j \in J} x_{ij} = 1, && i \in I, \\
& x_{ij} \leq y_j, && i \in I, j \in J, \\
& \sum_{j \in J} y_j = p, \\
& x_{ij}, y_j \in \{0, 1\},
\end{aligned}
$$

where x_{ij} is a binary variable which takes the value one if the demand point i is assigned to the facility j and y_j, which takes the value one if the facility location j is selected. The first constraint states that the demand point is assigned to only one facility, the second ensures that the demand point i is assigned to the location j only if a facility is opened in j. Finally, the third constraint means that the number of facilities to be located is exactly p.

The problem arises in various domains, including facility location planning, supply chain management, transportation, and public services. The p-median problem can have additional constraints or variations depending on the specific context and requirements of the problem. For example, capacity constraints can be introduced to limit the number of demand points assigned to each facility location (Kuehn and Hamburger, 1963; Puerto, 2008). Robust or stochastic formulations may also consider uncertain demand Albareda-Sambola et al. (2011); Correia et al. (2018); Correia and Saldanha-da-Gama (2019). In Tansel et al. (1983), an extended survey of some extensions of the p-median problem can be reviewed.

An unified approach for p-facility location problems that studies a wide family of objective functions in facility location problems is the ordered p-median problem. The goal of the ordered p-median location problem is to minimise the ordered weighted average of the distances or transportation costs, between the clients/demand points and the server, once we have applied rank dependent compensation factors on them. This problem, introduced by Puerto and Fernández (2000), allows modelling location problems in which customers support the median (p-median) or the maximum (p-center) travel costs, among many other robust alternatives (see Puerto and Rodríguez-Chía (2019) for a recent survey).

1.6 Location problems with polyhedral barriers

An interesting extension of the discrete p-median is the p-median problem with barriers. In this case, there exist some areas that cannot be traversed and it is necessary to compute the minimum distance between each pair of points in the graph induced by the facility locations and demand points.

Formally, let $\mathcal{B} = \{B_1, \ldots, B_m\}$ be a finite set of closed and pairwise disjoint barrier sets in $\mathbb{R}^n$. The placement of a facility or travelling through the interior of these sets is forbidden. Then, the feasible region is given by $\mathcal{F} = \mathbb{R}^n \setminus \text{int}(\mathcal{B})$. Let $d_{\mathcal{B}}$ be the barrier distance, that is, the length of the shortest path that does not cross $\text{int}(\mathcal{B})$, joining a pair of points in $\mathcal{F}$. Let I be the set of potential facility locations and J the set of demand points that are assumed to be located in $\mathcal{F}$. The distance between a demand point $i \in I$ and a facility location $j \in J$ is represented by c_{ij}. In this case, the p-median with barriers has the same mixed-integer linear programming formulation as in (p-median). Note that the difficulty of this problem is based on a previous step of computing the shortest path that joins each pair of points. However, when the set of potential facility locations is not fixed 'a priori', and it is allowed to be located at any point in $\mathcal{F}$, the problem becomes much harder.

This problem, in the context of planar location modelling, represents restrictions that appear in a real-life context. One of them is the case in which there are regions (called forbidden regions) in which the placement of a facility is forbidden, but transportation is still possible. They can model state parks or regions where geographic characteristics forbid the construction of the facility. For a survey of location problems with forbidden regions, see Hamacher and Nickel (1995) or Nickel (1995). The case in which transportation is possible, but only at a higher cost (called congested regions) is studied in Butt and Cavalier (1996) or Mitchell and Papadimitriou (1991). In this case, in these regions, different travel speeds or travel costs are considered. However, the existence of military areas, mountain ranges, lakes, big rivers, and highways does not allow transportation. These examples of barrier regions involve circuit board design (LaPaugh, 1980), pipe network design for ships (Wangdahl et al., 1974), or location and routing with robots (Lozano-Pérez and Wesley, 1979).

The first time barrier regions were considered in location modelling was in Katz and Cooper (1981). In this paper, the authors consider the 1-median in the plane with the Euclidean distance and one circular barrier. The same authors extended their work to cases with several circles and introduced those with one polyhedral barrier set in the plane and distances measured by any l_τ metric, $1 < \tau < \infty$ (see Katz and Cooper (1979a,b)). Aneja and Parlar (1994); Butt and Cavalier (1996); Katz and Cooper (1981) studied the 1-median problem for polyhedral barriers and any l_τ norm.

A variation of the 1-median problem, in which the barrier is a line with a finite number of passages, such as rivers or highways, and any distance induced by a norm, is studied in Klamroth (2001). This variation was extended to a bi-objective version in Klamroth and Wiecek (2002).

On the other hand, Batta et al. (1989) considered the stochastic queue median prob-

lem in the presence of barriers while employing the distance l_1. The authors developed connections between network location problems and planar location problems with this metric. Finally, Fekete et al. (2005) introduced the p-median problem with continuous demand over some given polyhedral set, possibly with holes acting as barriers to travel, and l_1 metric.

Chapter 2

Goals

In this chapter, the objectives of this thesis are established. The main objective of this thesis is the use of conic programming to extend state-of-the-art routing and location models. These new models seek for the applicability to more realistic scenarios. For each model studied in this thesis, the following objectives are considered:

- To analyse the current state-of-the-art in the field. This involves studying the assumptions made in existing works to identify opportunities for relaxing some of these assumptions and incorporating new elements to enrich the problem.
- To introduce new mathematical programming formulations that provide exact solutions to the problem. Additionally, to explore the possibilities of reinforcing these formulations with bound-tightening and valid inequalities to effectively reduce the feasible solution space.
- To study the structure of the problem and propose theoretical results that can be exploited to simplify the problem. This will include employing preprocessing techniques and variable fixing strategies.
- To develop a matheuristic that take advantage of the structure of the problem to obtain high-quality solutions within a reasonable time. This matheuristic will serve as initialisation for the exact model and will aim to obtain near-optimal solutions in certain scenarios.
- To conduct extensive computational experiments to compare the performance of the proposed formulations, as well as the effectiveness of reinforcement and variable fixing techniques. The study will also assess the quality of the approximate solutions obtained through the matheuristic approach.

Chapter 3

Results

The following chapter describes the results attained related to the objectives proposed in the previous chapter of this thesis. These results have given rise to the publication of four articles: Puerto and Valverde (2022), Amorosi et al. (2021), Amorosi et al. (2022) and Amorosi et al. (2023), which correspond, respectively, to the chapters 6, 9, 10 and 11. The results can be summarised as follows.

- New routing and location continuous problems have been proposed by exploiting the possibility of modelling distances with conic programming tools. A variant of the travelling salesman problem with neighbourhoods is considered that includes the traversing of a percentage of the length of the polygonal chains. Linear barriers that cannot be crossed have been included in the classical travelling salesman problem and p-median problem with neighbourhoods. The mothership drone routing problem has been extended in different directions: the possibility of the mothership of moving on a continuous framework or only on a general network; target graphs whose edges must be traversed in some percentage of their lengths; coordination of multiple drones and drones that have the possibility of visiting more than one target in a single operation.

- Mixed-integer second-order conic programming formulations have been developed for the problems described above. They have been reinforced by studying the best bounds for the variables and including valid inequalities that break symmetries. Using these formulations, some matheuristics are proposed to obtain feasible solutions in a reasonable time.

- Theoretical study of the exact formulations that include relations in terms of the lower bounds of the continuous relaxations between formulations; use of the Lagrangian dual to construct a Benders-like decomposition algorithm; study of geometric properties of some problems to generate dominating sets; NP-hardness results, among others.

- An extensive series of computational experiments have been conducted to evaluate the effectiveness of the proposed formulations and matheuristics to solve the problems. Various types of instances were used in these experiments, including benchmark instances borrowed from the existing literature, randomly generated instances, and those inspired by real-world scenarios.

Chapter 4

Discussion of the results

The results presented in the previous chapter, related to the material included in chapters 6 to 11 of this thesis, are discussed in terms of contributions to the state-of-the-art.

In Chapter 6, an extension of the travelling salesman problem with neighbourhoods is considered. This assumes a cost structure, that is, the travel cost of moving between neighbourhoods and the travel cost of crossing a neighbourhood. Unlike the classical TSPN, it is necessary to locate exit and entry points in each neighbourhood. It also includes arc routing elements, like polygonal chains, that must be traversed in some specified percentage of their lengths. To solve this problem, neighbourhoods and polygonal chains are modelled by using unions of second-order cone representable sets. Then, two different approaches are used to state the problem. First, a time-dependent formulation is employed that allows us to include a number of specific characteristics in the modelling phase, such as time-dependent travel distances, time windows, or time-dependent discount factors. Then another formulation that does not make reference to stages in the routes and that simplifies the model is described by using McCormick envelopes. To provide good quality solutions, a matheuristic approach is proposed by integrating a clustering phase based on the 1-median to find the point candidates and a variable neighbourhood search phase to find the best route with the points obtained in the previous phase. Later, the geometry of the neighbourhoods is explored to fix a priori some variables in the formulations and to increase the efficiency of the model. In addition, the linearisation of the objective function yields bigM constants. Therefore, good upper and lower bounds are provided for these constants. Subsequently, an alternative row generation approach is presented to solve the problem based on a Benders decomposition (Benders, 1962) observing that fixing the binary variables, the continuous problem is convex and easy to solve. Finally, a battery of computational experiments have been performed by generating three types of neighbourhood: circles, regular polygons, and polygonal chains of different sizes. For each typology, time- and non-time-dependent formulations are compared. Since the latest provides better results, the Benders cuts approach is compared with the non-time-dependent approach. Again, formulations yield better results. Finally, new experiments are devoted to conclude that initialising the proposed formulations with an initial solution provided by the matheuristic helps in solving the problem.

Chapter 7 extends the neighbourhood version of the TSP by including linear barriers that cannot be crossed. The associated shortest path problem with neighbourhoods and barriers is described, and later used as a building block for the TSP extension. Since points are not fixed a priori, to check if a path is valid in the problem, a computational geometry result is used to check if the path intersects a barrier in terms of the orientations of these points with respect to the barrier. This geometric result makes use of determinants. Hence, mixed-integer quadratic programming formulations that take into account this fact are proposed. The chapter distinguishes between the case when two neighbourhoods cannot be joined by a rectilinear path that does not cross any barrier (hidden version) and when this is possible (general version). A proof of NP-completeness is given for this extension. This discussion is motivated because the hidden version is easier to solve since the

problem becomes convex, while the general version is not. Then, a preprocessing result is proposed that allows us to fix some variables based on analysing the relative position between the neighbourhoods and the barriers. In addition, there are some results that adjust the bigM constants that appear in the constraints that bound the determinants in the formulations. Afterwards, computational experiments are conducted to study the performance of the formulations. In this section, the process of data generation is firstly described, taking into account the hidden version and assuming that barriers do not cross each other. Then, a proposition is introduced that establishes an upper bound for the number of balls that can be generated in this context. Next, the creation of neighborhoods is addressed, involving both random and fixed sizes. The random case serves to examine the performance of both versions of the problem, while the fixed case focuses on evaluating the effectiveness of the general model in terms of the overlapping ratio of the neighborhoods. A relevant insight from Mennell (2009) is presented, indicating that the instance difficulty increases with a higher overlap ratio in the neighborhoods. The results obtained state that solving the problem considering balls as neighbourhoods is harder than solving with segments. In addition, strengthening increases the number of instances in which the solver finds a solution and the fraction of gaps certified after the execution time and improves significantly the time spent by the solver to get the optimal solution. On the other hand, the other experiment allows us to conclude that the larger the radii of the neighbourhood, the higher the complexity of the problem to be solved in line with the existing trend in the literature.

In Chapter 8, the p-median version with neighbourhoods and barriers is described. In this case, the formulations are modelled using a shortest-path geodesic representation, based on the problems studied in Mitchell (2017). Again, quadratically constrained mixed-integer formulations are stated. However, the objective function takes into account both the Euclidean and link weighted distances to join the selected sources with their assigned targets. Then, since finding even feasible solutions becomes a challenge for large-size instances, motivated by the results obtained for the TSP, a matheuristic is proposed. This procedure is based on a reduction to the classical p-median problem with barriers. The basic idea of this procedure is to consider only the centres of each neighbourhood as the points chosen in each of them. By fixing those points, the resulting problems become mixed-integer linear. This reduction allows to obtain feasible solutions for them. To computationally compare the models, random-sized neighbourhoods and barriers are generated taking into account the same approach as in Chapter 7. In this case, the parameter p that controls the number of neighbourhoods in which a facility is located and the number of barriers considered for the same instance are also studied. In addition, formulations are tested without and with the initial solution provided by the matheuristic. The results state that the solver is capable of finding feasible solutions up to 80 neighbourhoods when the hidden formulation is considered while the general version the solver cannot find even a feasible solution for instances with 50 and 80 neighbourhoods. It is remarkable that the initial solution found by the matheuristic helps to certify optimality in less CPU time and also to get better gaps whenever the optimal solution is not found within the time limit.

For medium to large-size instances, the matheuristic algorithm always finds a feasible solution that cannot be improved by the solver. Another remarkable fact is that the higher the percentage of barriers that are set, the greater the difficulty of the problem.

Chapter 9 develops the mothership and drone routing problem. In this version of the problem, target points to be visited by the drone are extended to graphs that permit one to model several real situations like roads or wired networks inspection. A graph is considered to be visited if a fraction of the length of each edge is traversed or the total length of the graph is traversed. In addition, two different versions of the problem are considered depending on the assumptions made on the movements of the mothership vehicle: it can move freely on the continuous space (all terrain ground vehicle, boat on the water, or aircraft vehicle); it must move on a road network (that is, it is a normal truck or van). Firstly, new second-order cone programming formulations are proposed for these models. A first attempt to model this problem uses stages identified with the order in which the different graphs in the problem are visited. Then, the use of stages is modified by including constraints that ensure connectivity, associating the launching and rendezvous point with each graph. Second, the bigM constants that appear in the formulations to model the different distances covered by both the drone and the mothership are tightened for each of the variants considered in the chapter. Thirdly, given the difficulty of these types of problems, a general matheuristic framework is designed to address a feasible solution for the problem. The idea of the algorithm is to solve a TSP over the neighbourhoods of those centroids of the graphs using the TSP with neighbourhoods from Chapter 6. Then, given such an order of visits, the launching/rendezvous points where the mothership must stop are determined by solving the problem but limited to one single graph at a time, following the sequence previously computed. Finally, all the formulations are tested in small instances of two typologies, Delaunay and grid graphs. A novel way to randomly generate the latest is described. The main results are that connectivity formulations provide a better gap than stages, but, in many cases, all these problems are hard to solve and only it is possible to obtain a feasible solution. In addition, the matheuristic is capable of improving the quality of the solutions when its solution is used as initial one in the exact model. Finally, one observes that using Delaunay or grid graphs and visiting a fraction of each edge or all graphs are not very different in terms of complexity.

Chapter 10 reports the all terrain mothership and drone routing problem that visits points and polygonal chains, but allowing to visit more than one target in a single operation if drone has enough endurance. First, a mixed-integer second-order cone programming formulation is described that includes the variant of visiting more targets. Then, the formulation is strengthened by fixing a priori some variables in terms of the endurance of the drone; presenting valid inequalities that compactify all operations made by the drone for the first operations and tightening bigM constants for the distances covered by the drone. Second, a matheuristic based on the formulation of the TSPN of Chapter 6 is used to generate the order of a feasible solution. This order is set by fixing only the binary variables of the model and solving the resulting problem to obtain a complete feasible solution that includes also the continuous variables. Once again, both procedures are

computationally tested in randomly generated instances that include only point targets, only polygonal targets, and a third set with points and polygonals. The results show that for small instances, considering points as targets is easier to solve than polygonals or a mixture. However, for the biggest instances, this behaviour changes, and points are harder to be solved in terms of gap. A second test with a larger time limit allows one to conclude that polygonal targets are still more difficult to be solved than the instances with point targets. The different structure of the targets provides an initial advantage in the lower bound of the polygonal target instances, due to the constraint related to the minimum ratio of each polygonal to be visited. Another remarkable result is that the matheuristic is a very good alternative to the exact model. This approach provides fast solutions with a small relative gap with respect to the exact formulation.

In Chapter 11, a multiple drone version of the mothership and drone routing problem with graphs is studied. Again, given the complexity of the problem, only the all-terrain case is analysed. To deal with this, two versions of the problem are presented that make the mathematical programming approach flexible with respect to different working principles of the tandem system. In the first version (called complete overlapping), operations consisting of the launch and retrieval of a set of drones are carried out sequentially so that no two consecutive launches are possible without the retrieval of previously launched drones. The second (called partial overlapping), which is an extension of the previous, allows consecutive launch or retrieval actions so that the visits of several drones to their target graphs are allowed to partially overlap over time. First, for the sake of simplicity, the fleet is assumed to be homogeneous, and drones have the same characteristics. Once mixed-integer second-order cone programming formulations for both versions are set, a result that links the two models is stated. The proof builds a feasible solution for the partial overlapping version as a solution from the complete overlapping. Then, it is shown that the models are not equivalent. Again, these formulations are strengthened by means of valid inequalities that concentrate all drone activities on the first operations and tightening the bigM constants. Afterwards, a matheuristic that groups the graphs into clusters is proposed for the complete overlapping version. This allows the fleet of drones to visit all the graphs in the clusters in the same operation. This algorithm is also used to generate solutions for the partial overlapping model taking advantage of the relationship between models. Formulations and matheuristic are also tested in small-sized instances, giving rise to the conclusion that the more general version is the hardest to be solved. In addition, a sensitivity test is performed to conclude that the higher the number of drones and endurance, the lower the makespan of the mothership route. Then, a realistic application of the system studied in this paper to perform surveillance operations is described. This case study considers the Cordoba Courtyards Festival to monitor the situation to avoid the concentration of people in the context of the pandemic. Finally, an extension of this model to deal with the case of nonhomogeneous fleets of drones is reported. A mixed integer second-order cone formulation is proposed, which is strengthened and tested computationally asserting that these kind of models are really hard to be solved and a matheuristic procedure is necessary even for finding feasible solutions.

Chapter 5

Conclusions

This thesis focuses on the analysis of various mathematical programming problems that combine location and routing approaches. Routing-location problems continue to be a captivating field of study, and this research has explored novel insights in this area. New routing-location problems are introduced and studied in order to be applicable in logistics, especially, in a context of widespread use of drones. To address these problems, several mathematical programming formulations have been developed that serve as the basis for other solution approaches. In addition, the properties of these problems have been investigated. It has allowed us to design supplementary algorithms, like matheuristics, to aid in their solution. This theoretical methodology has been validated through several computational experiments using generated instances and, in other cases, case study situations. Each chapter of this thesis (Chapters 6-11) is independently presented and provides its own conclusions. To provide a comprehensive overview of the significant accomplishments of the thesis, these conclusions are summarised in the following. In addition, some potential avenues for future research are proposed.

Chapter 6 investigates a novel variant of the travelling salesman problem with neighbourhoods. This problem can be formulated as mixed-integer second-order cone program. Several exact formulations are presented, which have been extensively tested on a set of instances. Furthermore, a heuristic algorithm is proposed that produces high-quality solutions. This algorithm is suitable for large-scale problems and can serve as an initialisation procedure for exact solvers when dealing with formulations. Computational experiments reveal the considerable difficulty of the problem, with exact approaches failing to find optimal solutions within a two-hour CPU time limit for instances containing only 20 neighbourhoods. This study paves the way for various research directions and extensions of the core problem, which can be incorporated into the model. For instance, potential avenues include improving formulations or decomposition schemes to enable exact solutions for larger instances, developing alternative heuristic algorithms capable of handling large-scale problems, incorporating conditions for nonlinear neighbourhood borders (e.g., circles), and addressing the limited endurance of drones by requiring them to return to a depot for recharging before completing the route. Some of these topics will be explored further in the future.

Chapter 7 focuses on two problems, namely the shortest path problem with neighbourhoods and barriers and the travelling salesman problem with neighbourhoods and barriers, where direct movements between neighbourhoods are prohibited due to barriers. The more general case gives rise to non-convex mixed-integer problems. It remains an open question whether there exists a finite dominating set with polynomial cardinality for the general version, which could potentially simplify the problem formulation. Additionally, it would be valuable to integrate different types of barriers, such as polygonals and second-order cone representable sets, into a unified model. These problems represent natural extensions to the ones addressed in this chapter and will be the focus of future investigations.

Chapter 8 addresses the p-median problem with hidden neighbourhoods and barriers and its general version, where the assumption that neighbourhoods are not visible to each other is removed. The latter version leads to non-convex mixed-integer problems, whereas

the former results in second-order cone mixed-integer problems. Despite their similarities, these two problems exhibit significant differences in terms of computational difficulty, as discussed in the computational experiments section. Nevertheless, the proposed mathematical programming approaches offer a formal treatment that enables optimal solutions for small to medium-sized instances. For larger instances, this approach also inspires a matheuristic algorithm that delivers high-quality solutions within short computation times by leveraging the structure of the problem. The existence of a finite dominating set with polynomial cardinality for the more constrained version remains an open question, which has the potential to simplify the underlying graph structures and problem solutions. Additionally, due to the complexity of the problem, investigating valid inequalities that reduce the space of feasible solutions will play a crucial role in efficiently solving larger instances. Moreover, extending these problems by assuming limited path lengths between the source and its associated target can be considered. Combining different types of barriers, such as piecewise-linear and second-order cone representable sets, in the same model would also be an intriguing avenue to explore. Furthermore, studying three-dimensional barriers that represent buildings paths would bring the drone delivery industry closer to real-life applications. All of these problem extensions mentioned above are natural continuations from those addressed in this chapter and are likely to attract the attention of researchers in the future.

Chapter 9 focuses on analysing the coordination problem between a mothership vehicle and a drone, aiming to minimise travel distances while visiting a set of targets represented by graphs. Exact formulations for different versions of the problem are provided, depending on the constraints imposed on the movement of the mothership (freely in a continuous space or constrained to a given network). Computational results indicate that the problem is highly challenging, with only small to medium-sized instances solvable to optimality. To address this, a matheuristic algorithm that can be applied to all problem versions with minimal modifications is proposed, producing feasible solutions within short computation times. This algorithm serves as a promising alternative to exact methods. Future research in this area will explore the coordination of operations involving multiple drones with a mothership in combination with the possibility of visiting more than one target per operation. Although these topics are highly appealing, they exceed the scope of this chapter and will be the focus of a follow-up research.

Chapter 10 addresses a coordination problem between a mothership vehicle and a drone, considering the scenario where the drone can visit multiple targets during its operation. The objective is to minimise the overall travel distance by synchronising the displacements of both vehicles. The mothership is capable of freely moving in a continuous space, allowing flexibility in launching and recovering the drone at optimal points. An exact model for this problem is presented that can accommodate both point-wise and graph-like targets. The model is formulated as a mixed-integer second-order cone problem, which can be effectively solved using modern solvers. However, the computational results demonstrate that the problem is challenging and that only small to medium-sized instances can be solved to optimality. To address the difficulty of larger instances, a matheuristic

algorithm has been developed as a computationally efficient alternative to exact methods. This algorithm produces feasible solutions within short computation times, providing acceptable results. An intriguing related problem is the coordination of operations involving one or more motherships and multiple drones, where each drone is allowed to visit multiple targets per operation. This realistic and challenging problem mirrors the situations encountered in drone delivery scenarios. However, while this problem is highly interesting and deserves attention, it exceeds the scope of this chapter and will be explored in a subsequent research publication.

Chapter 11 addresses the coordination problem between a mothership vehicle and a fleet of drones, aiming to minimise the makespan while visiting a set of targets represented by graphs. Exact formulations of the problem are provided using mixed-integer nonlinear programming for both complete and partial overlapping versions. To strengthen the models, some valid inequalities are incorporated. The computational results reveal that the problem is highly challenging, even for small and medium-sized instances. Consequently, a matheuristic algorithm is proposed as an alternative to exact methods, which produces high-quality feasible solutions within a short computing time. Extensive computational experiments are conducted using randomly generated instances, and a case study related to COVID-19 inspection activities is presented at the Courtyards Festival in Córdoba. The application of the system described in this paper is illustrated, showcasing the solutions obtained through the problem formulation and the initialisation provided by the proposed matheuristic. The formulation and algorithms proposed in this chapter serve as essential components for coordination systems consisting of a mothership and multiple drones. Future research in this area should focus on developing faster and more accurate algorithms capable of solving larger instances. Additionally, extensions can be explored, such as considering the time required for the mothership to launch and retrieve drones, treating the speeds of the mothership and drones as decision variables, and accommodating different shapes of graph edges, such as curve lines instead of straight lines. These problems are of significant interest and go beyond the scope of this paper, which warrants further investigation in subsequent research efforts.

Bibliography

(2005). Related Problems and Outlook. In Nickel, S. and Albandoz, J. P., editors, *Location Theory: A Unified Approach*, pages 419–422. Springer, Berlin, Heidelberg.

Agatz, N., Bouman, P., and Schmidt, M. (2018). Optimization Approaches for the Traveling Salesman Problem with Drone. *Transportation Science*, 52(4):965–981.

Aggarwal, A., Coppersmith, D., Khanna, S., Motwani, R., and Schieber, B. (2000). The Angular-Metric Traveling Salesman Problem. *SIAM Journal on Computing*, 29(3):697–711.

Ahuja, R., Magnanti, T., and Orlin, J. (1993). *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, N.J.

Albareda-Sambola, M., Fernández, E., and Saldanha-da-Gama, F. (2011). The facility location problem with Bernoulli demands. *Omega*, 39(3):335–345.

Alizadeh, F. (1995). Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization. *SIAM Journal on Optimization*, 5(1):13–51.

Amorosi, L., Puerto, J., and Valverde, C. (2021). Coordinating drones with mothership vehicles: The mothership and drone routing problem with graphs. *Computers & Operations Research*, 136:105445.

Amorosi, L., Puerto, J., and Valverde, C. (2022). An extended model of coordination of an all-terrain vehicle and a multivisit drone. *International Transactions in Operational Research*, n/a(n/a).

Amorosi, L., Puerto, J., and Valverde, C. (2023). A multiple-drone arc routing and mothership coordination problem. *Computers & Operations Research*, 159:106322.

Aneja, Y. P. and Parlar, M. (1994). Technical Note—Algorithms for Weber Facility Location in the Presence of Forbidden Regions and/or Barriers to Travel. *Transportation Science*, 28(1):70–76.

Arkin, E. M. and Hassin, R. (1994). Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197–218.

Augier, M. and Teece, D. J. (2021). *The Palgrave Encyclopedia of Strategic Management*. Palgrave Macmillan.

Batta, R., Ghose, A., and Palekar, U. S. (1989). Locating Facilities on the Manhattan Metric with Arbitrarily Shaped Barriers and Convex Forbidden Regions. *Transportation Science*, 23(1):26–36.

Benavent, E., Carrotta, A., Corberán, A., Sanchis, J. M., and Vigo, D. (2007). Lower bounds and heuristics for the Windy Rural Postman Problem. *European Journal of Operational Research*, 176(2):855–869.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252.

Boysen, N., Fedtke, S., and Schwerdfeger, S. (2021). Last-mile delivery concepts: A survey from an operational research perspective. *OR Spectrum. Quantitative Approaches in Management*, 43(1):1–58.

Burness, R. C. and White, J. A. (1976). The Traveling Salesman Location Problem. *Transportation Science*, 10(4):348–360.

Butt, S. E. and Cavalier, T. M. (1996). An efficient algorithm for facility location in the presence of forbidden regions. *European Journal of Operational Research*, 90(1):56–70.

Campbell, J. F., Corberán, Á., Plana, I., and Sanchis, J. M. (2018). Drone arc routing problems. *Networks. An International Journal*, 72(4):543–559.

Corberán, Á. and Laporte, G., editors (2015). *Arc Routing*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics.

Correia, I., Nickel, S., and Saldanha-da-Gama, F. (2018). A stochastic multi-period capacitated multiple allocation hub location problem: Formulation and inequalities. *Omega*, 74:122–134.

Correia, I. and Saldanha-da-Gama, F. (2019). Facility Location Under Uncertainty. In Laporte, G., Nickel, S., and Saldanha da Gama, F., editors, *Location Science*, pages 185–213. Springer International Publishing, Cham.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a Large-Scale Traveling-Salesman Problem. *Journal of the Operations Research Society of America*, 2(4):393–410.

Di Placido, A., Archetti, C., and Cerrone, C. (2022). A genetic algorithm for the close-enough traveling salesman problem with application to solar panels diagnostic reconnaissance. *Computers & Operations Research*, 145:105831.

Drezner, Z. and Hamacher, H. W. (2004). *Facility Location: Applications and Theory*. Springer Science & Business Media.

Fekete, S. P., Mitchell, J. S. B., and Beurer, K. (2005). On the Continuous Fermat-Weber Problem. *Operations Research*, 53(1):61–76.

Fiacco, A. V. and McCormick, G. P. (1990). *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Society for Industrial and Applied Mathematics.

Fox, K. R., Gavish, B., and Graves, S. C. (1980). An N-Constraint Formulation of the (Time-Dependent) Traveling Salesman Problem. *Operations Research*, 28(4):1018–1021.

França, P. M., Gendreau, M., Laporte, G., and Müller, F. M. (1995). The M-Traveling Salesman Problem with Minmax Objective. *Transportation Science*, 29(3):267–275.

Frederickson, G. N., Hecht, M. S., and Kim, C. E. (1976). Approximation algorithms for some routing problems. In *17th Annual Symposium on Foundations of Computer Science (Sfcs 1976)*, pages 216–227.

Garfinkel, R. S. and Webb, I. R. (1999). On crossings, the Crossing Postman Problem, and the Rural Postman Problem. *Networks. An International Journal*, 34(3):173–180.

Ghiani, G. and Laporte, G. (2015). Chapter 5: The Undirected Rural Postman Problem. In *Arc Routing*, MOS-SIAM Series on Optimization, pages 85–99. Society for Industrial and Applied Mathematics.

Golden, B. L., Levy, L., and Vohra, R. (1987). The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318.

Gutin, G. and Punnen, A. P. (2006). *The Traveling Salesman Problem and Its Variations*. Springer Science & Business Media.

Hakimi, S. L. (1965). Optimum Distribution of Switching Centers in a Communication Network and Some Related Graph Theoretic Problems. *Operations Research*, 13(3):462–475.

Hamacher, H. W. and Nickel, S. (1995). Restricted planar location problems and applications. *Naval Research Logistics (NRL)*, 42(6):967–992.

Katz, I. N. and Cooper, L. (1979a). Facility location in the presence of forbidden regions, II: Euclidean distance and several forbidden circles. Technical Report OREM 79006, Southern Methodist University, Dallas, TX 75275.

Katz, I. N. and Cooper, L. (1979b). Facility location in the presence of forbidden regions, III: Lp distance and polygonal forbidden regions. Technical Report OREM 79011, Southern Methodist University, Dallas, TX 75275.

Katz, I. N. and Cooper, L. (1981). Facility location in the presence of forbidden regions, I: Formulation and the case of Euclidean distance with one forbidden circle. *European Journal of Operational Research*, 6(2):166–173.

Khachiyan, L. G. (1980). Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72.

Klamroth, K. (2001). Planar weber location problems with line barriers. *Optimization*, 49(5-6):517–527.

Klamroth, K. and Wiecek, M. M. (2002). A Bi-Objective Median Location Problem With a Line Barrier. *Operations Research*, 50(4):670–679.

Kuehn, A. A. and Hamburger, M. J. (1963). A Heuristic Program for Locating Warehouses. *Management Science*, 9(4):643–666.

LaPaugh, A. S. (1980). *Algorithms for Integrated Circuit Layout: An Analytic Approach.* PhD thesis, Massachusetts Institute of Technology.

Laporte, G. (1997). Modeling and solving several classes of arc routing problems as traveling salesman problems. *Computers & Operations Research*, 24(11):1057–1061.

Laporte, G. and Nobert, Y. (1983). Generalized Travelling Salesman Problem Through n Sets Of Nodes: An Integer Programming Approach. *INFOR: Information Systems and Operational Research*, 21(1):61–75.

Lobo, M. S., Vandenberghe, L., Boyd, S., and Lebret, H. (1998). Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1-3):193–228.

Lozano-Pérez, T. and Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570.

Mennell, W. K. (2009). *Heuristics for Solving Three Routing Problems: Close-Enough Traveling Salesman Problem, Close-Enough Vehicle Routing Problem, Sequence-Dependent Team Orienteering Problem.* PhD thesis, University of Maryland, College Park.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM*, 7(4):326–329.

Minieka, E. (1979). The Chinese Postman Problem for Mixed Networks. *Management Science*, 25(7):643–648.

Mitchell, J. S. B. (2017). Shortest Paths and Networks. In *Handbook of Discrete and Computational Geometry.* Chapman and Hall/CRC, third edition.

Mitchell, J. S. B. and Papadimitriou, C. H. (1991). The weighted region problem: Finding shortest paths through a weighted planar subdivision. *Journal of the ACM*, 38(1):18–73.

Murray, C. C. and Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109.

Murray, C. C. and Raj, R. (2020). The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies*, 110:368–398.

Nemirovski, A. (2006). Advances in convex optimization: Conic programming. *Proceedings oh the International Congress of Mathematicians, Vol. 1, 2006-01-01, ISBN 978-3-03719-022-7, pags. 413-444*, 1.

Nesterov, Y. and Nemirovski, A. (1994). *Interior-Point Polynomial Algorithms in Convex Programming*. Studies in Applied and Numerical Mathematics. Society for Industrial and Applied Mathematics.

Nesterov, Yu. and Nemirovski, A. (1992). Conic formulation of a convex programming problem and duality. *Optimization Methods and Software*, 1(2):95–115.

Nickel, S. (1995). *Discretization of Planar Location Problems*. Shaker Verlag, Aachen.

Otto, A., Agatz, N., Campbell, J., Golden, B., and Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks*, 72(4):411–458.

Papadimitriou, C. H. (1976). On the complexity of edge traversing. *Journal of the ACM*, 23(3):544–554.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. Courier Corporation.

Poikonen, S. and Golden, B. (2020). The Mothership and Drone Routing Problem. *INFORMS Journal on Computing*, 32(2):249–262.

Poikonen, S., Wang, X., and Golden, B. (2017). The vehicle routing problem with drones: Extended models and connections. *Networks*, 70(1):34–43.

Puerto, J. (2008). A New Formulation of the Capacitated Discrete Ordered Median Problems with {0, 1}-Assignment. In Kalcsics, J. and Nickel, S., editors, *Operations Research Proceedings 2007*, Operations Research Proceedings, pages 165–170, Berlin, Heidelberg. Springer.

Puerto, J. and Fernández, F. R. (2000). Geometrical properties of the symmetrical single facility location problem. *Journal of Nonlinear and Convex Analysis*, 1:321–342.

Puerto, J. and Rodríguez-Chía, A. M. (2019). Ordered Median Location Problems. In Laporte, G., Nickel, S., and Saldanha da Gama, F., editors, *Location Science*, pages 261–302. Springer International Publishing, Cham.

Puerto, J. and Valverde, C. (2022). Routing for unmanned aerial vehicles: Touring dimensional sets. *European Journal of Operational Research*, 298(1):118–136.

Shuttleworth, R., Golden, B. L., Smith, S., and Wasil, E. (2008). Advances in Meter Reading: Heuristic Solution of the Close Enough Traveling Salesman Problem over a Street Network. In Golden, B., Raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43, pages 487–501. Springer US, Boston, MA.

Tansel, B. C., Francis, R. L., and Lowe, T. J. (1983). Location on Networks: A Survey. Part I: The p-Center and p-Median Problems. *Management Science*, 29(4):482–497.

van der Poort, E. S., Libura, M., Sierksma, G., and van der Veen, J. A. A. (1999). Solving the K-best traveling salesman problem. *Computers & Operations Research*, 26(4):409–425.

Wangdahl, G. E., Pollock, S. M., and Woodward, J. B. (1974). Minimum-Trajectory Pipe Routing. *Journal of Ship Research*, 18(01):46–49.

Chapter 6

Routing for unmanned aerial vehicles: Touring dimensional sets

European Journal of Operational Research 298 (2022) 118–136

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Routing for unmanned aerial vehicles: Touring dimensional sets

Justo Puerto[1], Carlos Valverde[1,*]

Department of Statistics and Operations Research, University of Seville, Seville 41012, Spain

ARTICLE INFO

Article history:
Received 18 January 2021
Accepted 29 June 2021
Available online 9 July 2021

Keywords:
Routing
Networks
Logistics
Conic programming and interior point methods

ABSTRACT

In this paper we deal with an extension of the crossing postman problem to design routes that have to visit different shapes of dimensional elements rather than edges. This problem models the design of routes of drones or other vehicles that must visit a number of geographical elements to deliver some good or service and then move directly to the next using straight line displacements. We present two families of mathematical programming formulations. The first one is time-dependent and captures a number of characteristics of real applications at the price of using three indexes variables. The second family of formulations is not time-dependent, instead it uses connectivity properties to ensure the proper definition of routes. We compare them on a testbed of instances with different shapes of elements: second order cone (SOC) representable and polyhedral neighborhoods and polygonal chains. The computational results reported in this paper show that our models are useful and our formulations can solve to optimality medium size instances of sizes similar to other combinatorial problems including neighborhoods that have already been studied in the literature. To address larger instances we also present a heuristic algorithm that runs in two phases: clustering and Variable Neighborhood Search. This algorithm performs very well since it provides promising feasible solutions and, in addition, it can be used to initialize the solvers with feasible solutions.

1. Introduction

Drones, or UAVs (unmanned aerial vehicles), provide new opportunities for improving logistics in a variety of settings. Specifically, we would like to emphasize, among other characteristics, their capability for moving without an underlying network using straight line displacements. Recent technological improvements as battery life, better communication devices and reduction in manufacturing costs have increased the use of drones in logistics. Thus, this technology has increased its use in many different fields as disaster management in remote regions (see Knight, 2016), parcel delivery as shown in Lavars (2015), communication coverage, worked in Amorosi, Chiaraviglio, D'Andreagiovanni, & Blefari-Melazzi (2018), traffic monitoring, infrastructure inspection, coastal surveying and many other applications. The reader is referred to the review by Otto, Agatz, Campbell, Golden, & Pesch (2018) for further references.

The availability of this new technology has brought new business opportunities and, at the same time, has opened a lot of new challenges in the Operations Research field to propose solutions to new emerging problems in the areas of logistics and routing. As drones play a growing role in business operations, questions of planning and optimization increase in practical and academic importance. However, some of the characteristics of drone's displacement are not fully exploited by most previous routing models in literature. Unlike standard ground vehicles that must follow paths, drones can use direct connections by straight lines between destinations because they can fly across areas, but their limited battery autonomy range makes the problem of coordination with mothership vehicles a challenging problem.

In 1962, Meigu Guan introduced the undirected Chinese Postman Problem (CPP) whose aim is to determine a least-cost closed route that traverses all edges of the graph. Orloff (1974) extended the CPP to travel through a subset of required edges that is known as the Rural Postman Problem (RPP). Based on this idea, Garfinkel & Webb (1999) introduced the Crossing Postman Problem (XPP) which relaxes the RPP to the case in which it is permitted to leave the edges of the network and cross from one edge to another at points other than the original vertices. These Arc Routing Problems (ARP) are studied in depth in Corberán & Laporte (2015). On the other hand, some drone routing problems inherit some of the structure of the well-known Traveling Salesman Problem with

* Corresponding author.
E-mail addresses: puerto@us.es (J. Puerto), cvalverde@us.es (C. Valverde).
[1] Both the authors contributed equally to this work.

https://doi.org/10.1016/j.ejor.2021.06.061

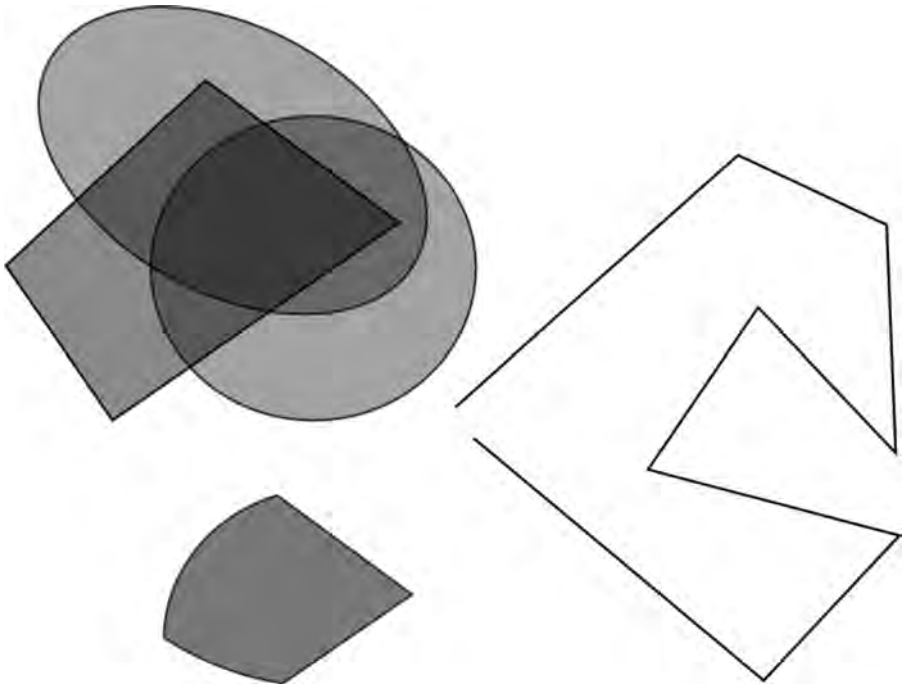


Fig. 1. An example of convex sets and polygonal chains considered in the problem.

neigborhoods (TSPN) that was first introduced by Arkin & Hassin (1994) and later was studied, among others, by Gentilini, Margot, & Shimada (2013) using convex sets and Yuan & Zhang (2017) presenting a hybrid framework in which metaheuristics and classical TSP solvers are combined strategically to produce high quality solutions for TSPN with arbitrary neighborhoods. Some other combinatorial optimization problems analyzed with neighborhoods are shortest paths in Disser, Mihalák, Montanar, & Widmayer (2014), minimum spanning trees in Yang, Lin, Xu, & Xie (2007), Blanco, Fernández, & Puerto (2017), ordered *p*-median location, in Blanco (2019) and hub location, Blanco & Puerto (2021).

The aim of this paper is motivated by the design of drones' routes that must connect a number of dimensional targets with given shapes, that we will call from now on *elements*, that are located on an area. The use of this terminology is not new and the interested reader is referred to Schöbel (2015), Díaz-Báñez, Mesa, & Schöbel (2004) and Mallozzi, Puerto, & Rodríguez-Madrena (2019) for further details and references on the concept of dimensional facilities. In addition, in some cases it will be required some extra service beyond the simple visit to an element. For instance, one has to visit a percentage of its total length (assuming that its dimension is one). In our approach we would like to exploit some new features of Mixed Integer Non-Linear Programming (MINLP) to develop formulations and solution algorithms. Obviously, we have to impose some limits to the shapes of the considered elements to achieve tractable models. As a first building block, we restrict ourselves to two main types of elements (see Fig. 1): convex bodies and piecewise linear chains (including segments). For the case of the convex bodies, they can represent regions that the drone must reach and where the customers are willing to pick up the orders (they can be seen as uniform probability densities). On the other hand, polygonal chains can be used to model different paths that the drone must follow to do some inspection or to avoid some barriers that can appear in a real-world scenario. The use of convex bodies in the model can be extended to more general shapes as explained in Section 2.

We assume a structure of costs trying to capture the main features of these situations. We assume that there are two types of costs: 1) the travel cost of moving between elements, and 2) the travel cost of crossing (moving on) an element. The travel costs of moving between elements may change over time: it may be cheaper to go from A to B at time 1 than at time 2. On the other hand, the cost of crossing an element may be cheaper or more expensive than moving between them: controlling the drone over polygonal chains to do some inspection may be more expensive than flying directly between targets. However, one may obtain some discount for flying over some large area (parks, lakes, natural reserves...) because the drone can do a secondary job, as reporting information in its way back to the base. This fact is represented as a weighting factor in the objective function as explained in Section 2. A survey of these coverage path planning problems can be found in Otto et al. (2018).

The goal of the model considered in this paper is to find a minimum total cost route that visits all the elements and traverses some proportions of those with dimension one. In the rest of the paper, we will refer to this problem as the Crossing Postman Problem with Neighborhoods (XPPN).

The contribution of this paper is to introduce new models for the design of routes that combine several characteristics that have not been previously analyzed simultaneously: design of routes without underlying graph structure, required targets (like in the RPP) defined on dimensional elements (as in the TSP with neighborhoods) that can be polygonal chains or other kind of more general sets and free entry and exit points over the elements. Combining these features altogether gives rise to a challenging new problem that is analyzed for the first time in this paper.

The paper is structured in 8 sections. The first section is the introduction. In the second section we describe the problem and set the notation followed in the rest of the paper. Section 3 is devoted to present different valid formulations of the problem. In Section 4 we present a heuristic algorithm for solving XPPN. This heuristic has two phases: clustering and Variable Neighborhood Search (VNS). The results show that it provides good quality solution in very limited computation time. Section 5 deals with some strengthening of our formulations: pre-processing variables

and deriving valid inequalities to be added to the formulations. Next, in Section 6 we present a decomposition algorithm *'a la'* Benders that can be also applied to solve the problem. We derive all the details of that decomposition and show preliminary computational results.

An extensive computational experience is reported in Section 7. There, we compare the different formulations in terms of final gaps and computing time. The paper ends with a section devoted to conclusions and extensions, where we list some interesting open lines of research connected with the problems addressed in this paper.

2. Description of the problem

Let V be a set of points (vertices) embedded in $\mathbb{R}^2$. (The reader may note that extensions to the three dimensional space are possible at the price of increasing the models' complexity.) Associated with each vertex $v \in V$, we assign an element $\mathcal{N}_v$ that can belong to two different types: either a convex set or a polygonal chain. Later, we will show how to extend the elements to deal with union of convex sets. In the former case, let $\mathcal{C}_v \subset \mathbb{R}^2$ denote the convex set associated to v that must contain v in its interior. In the latter, let $\mathcal{P}_v \subset \mathbb{R}^2$ denote the polygonal chain assigned to v that we assume to be parameterized by its breakpoints $A_v^1, \ldots, A_v^{n_v+1}$, where n_v is the number of line segments of the polygonal chain. We denote

$$V_{\mathcal{C}} = \{v \in V : v \text{ is associated with a convex set}\},$$

$$V_{\mathcal{P}} = \{v \in V : v \text{ is associated with a polygonal chain}\}.$$

Let us denote by $x_v^i \in \mathcal{N}_v : i = 1, 2,\ v \in V$ the access (x_v^1) and exit (x_v^2) points to the elements $\mathcal{N}_v$ associated with vertices $v \in V$. A feasible solution to the XPPN problem consists of a set of pairs of access and exit points, $X = \bigcup_{v\in V}\{x_v^1, x_v^2\}$, together with a tour $\mathcal{T}$ that the drone must traverse on the graph $G = (X, E)$, with edge set $E = E_{\text{out}} \cup E_{\text{in}}$, where:

$$E_{\text{out}} = \{(x_v^1, x_w^2) : v \neq w \in V\}, \quad E_{\text{in}} = \{(x_v^1, x_v^2) : v \in V\}.$$

Edges in the set E_{out} are links between different elements whereas those in E_{in} are those that define the part of the tour that is traveled within the convex neighborhoods or the polygonal chains while the drone is doing a secondary job. Observe that all the links in E_{in} are required and therefore they must be visited by the route. Edge lengths of an outside link (x_v^1, x_w^2) is given by the Euclidean distance, $d_{vw}(x_v^1, x_w^2) = \|x_v^1 - x_w^2\|_2$, between their endpoints. Edge length, $d_v(x_v^1, x_v^2)$, of an inner link (x_v^1, x_v^2) is computed as the distance measured over the corresponding element (polygonal or convex set). Observe that in the case of a polygonal the distance is computed as the sum of the lengths of the corresponding edges or partial edges, since the drone is following the path given by the polygonal chain.

The cost of a feasible solution $(X, \mathcal{T})$ is then given by the overall sum of *outside* edges plus the weighted sum of the inner edges:

$$d(X, \mathcal{T}) = \sum_{e_{vw}=(x_v^1,x_w^2)\in\mathcal{T}} d_{vw}(x_v^1, x_w^2) + \sum_{e_v=(x_v^1,x_v^2)\in\mathcal{T}} f_v\, d_v(x_v^1, x_v^2),$$

where f_v is a weighting factor for traveling within the neighborhoods. This factor depends on the worth given to a possible secondary job done by the drone. We point out that in case of overlapping of two or more neighborhoods the discount factor is accounted for each one of them, as shown in the above formula. The reader may note that in all our discussions we are assuming that the autonomy of the drone battery suffices to travel the whole route. Therefore, the model does not allow a route longer than the flying autonomy.

Throughout this paper we adopt the following notation:

- $\mathcal{T}_G$ as the set of incidence vectors associated with tours on G, i.e., $\mathcal{T}_G = \{z \in \mathbb{R}_+^{|E|} : z \text{ is a tour on } G\}$.
- $\mathcal{X} = \prod_{v\in V}(\mathcal{N}_v \times \mathcal{N}_v)$, the space where the access and exit points are selected.

The goal of XPPN is to find a feasible solution $(X, \mathcal{T})$ of minimal total cost. Then, it can be expressed as:

$$\begin{array}{ll} \min & \sum_{e_{vw}=(x_v^1,x_w^2)\in E_{\text{out}}} d_{vw}(e) z_e + \sum_{e_v=(x_v^1,x_v^2)\in E_{in}} f_e d_v(e) \\ \text{s.t} & z \in \mathcal{T}_G, \quad x \in \mathcal{X} \end{array} \tag{1}$$

Here it is assumed that the drone route enters and exits from an element only once. Note that, since the distance between neighborhoods is minimized, there always exists an optimal solution in which the drone visits each neighborhood only once. The reader may observe that the above formulation is only formal, but it is clearly not separable into the continuous and discrete counterparts since the access and exit point to each one of the elements (continuous part) depend on the order of the visit to the elements (discrete part) and vice versa. We also point out that the discrete part, that is a TSP, is an NP-hard problem whereas the continuous part, that is a location problem, is easily solvable by using interior-point algorithms. This structure is exploited to decompose the problem in a master problem (TSP) and a subproblem (Location Problem) in the Benders decomposition (see Section 6). Moreover, the problem involves Euclidean distances among variable points and sets, therefore it is not linearly representable. In spite of that, it is suitable to model this problem as a MINLP.

In this paper, we focus on the case where the sets $\mathcal{C}_v$ are second order cone (SOC) representable, that is, the sets can be expressed by using second-order cone constraints as follows:

$$x_v^i \in \mathcal{C}_v \iff \|A_v^j x_v^i + b_v^j\| \leq (c_v^j)^T x_v^i + d_v^j,\ j = 1, \ldots, n_v, \qquad (\mathcal{C}-\text{C})$$

where $x_v^i, i = 1, 2$ is the decision variable, A_v^j, b_v^j, c_v^j and d_v^j are parameters of the constraint j and n_v represents the number of constraints that appear in the block associated to vertex v.

Note that these inequalities can also model linear constraints (for $A_v^j, b_v^j \equiv 0$), ellipsoids and hyperbolic constraints (see Lobo, Vandenberghe, Boyd, & Lebret, 1998 for more details).

These type of elements could be extended further to unions of SOC representable sets. This type of neighborhood is obtained introducing binary variables, whose meaning is similar to those in disjunctive programming. Thus, we can determine in which set of the union happens the access or the departure points of the different sets.

Let $\{\mathcal{C}_v^1, \ldots, \mathcal{C}_v^{m_v}\}$ be the second order cone representable sets that define the neighborhood associated to the vertex v and let $\mathcal{U}_v = \bigcup_{\ell=1}^{m_v} \mathcal{C}_v^\ell$ denote the union of these sets. Consider the binary variable $\chi_v^{i\ell}$ that assumes the value of one if x_v^i is located in the set $\mathcal{C}_v^\ell$ and zero otherwise. Thus, for each $v \in V$, we can model that $x_v^i \in \mathcal{U}_v$ by using the following inequalities for each $i = 1, 2$:

$$x_v^i \in \mathcal{U}_v \iff \begin{cases} \|A_v^{j_\ell} x_v^i + b_v^{j_\ell}\| \leq (c_v^{j_\ell})^T x_v^i + d_v^{j_\ell} + M_v^{j_\ell}(1 - \chi_v^{i\ell}), \\ \quad \ell = 1, \ldots, m_v,\ j_\ell = 1, \ldots, n_{\ell v}, \\ \sum_{\ell=1}^{m_v} \chi_v^{i\ell} = 1, \end{cases} \qquad (\mathcal{U}-\text{C})$$

where $M_v^{j_\ell}$ is a big-M constant on the maximal distance between two points in the union of sets. The reader may observe that one can replace $(\mathcal{C}$-C) by $(\mathcal{U}-\text{C})$ in all our formulations without compromising their validity. Therefore, our model can deal easily with these more general forms of neighborhoods.

On the other hand, the second type of elements are the piecewise linear constraints. Let n_{Sv} be the number of line segments of the polygonal chain v. Since we need to refer to interior points of the segment, these continuum of points is parametrized by the two endpoints of the segment: $x \in [A_v^j, A_v^{j+1}]$ if and only if

$\exists \gamma \in [0,1]$ such that $x = \gamma A_v^j + (1-\gamma)A_v^{j+1}$. In order to deal with them, we introduce the following variables for each vertex $v \in V_{\mathcal{P}}$ and $i = 1, 2$:

- u_v: Binary variable that determines the traveling direction in the polygonal chain v.
- γ_v^{ij}: Continuous variable in [0,1] that represents the parameter value of the x_v^i variable in the line segment j of the polygonal chain v, $j = 1, \ldots, n_{Sv}$.
- μ_v^{ij}: Binary variable that is one when x_v^i is located in the line segment j of the polygonal chain v, and zero otherwise, for $j = 1, \ldots, n_{Sv}$.
- λ_v^i: Continuous variable in $[0, n_{Sv}]$ that models the parametrization of the entry or exit points along the polygonal chain associated with v.

Using these variables, we can determine the placement of the entry and exit points on the polygonal chain v introducing the following inequalities for each $i = 1, 2$:

$$x_v^i \in \mathcal{P}_v \Longleftrightarrow \begin{cases} \lambda_v^i - j \geq \gamma_v^{ij} - (n_{Sv}+1)(1-\mu_v^{ij}), & j = 2, \ldots, n_{Sv}+1 \\ \lambda_v^i - j \leq \gamma_v^{ij} + (n_{Sv}+1)(1-\mu_v^{ij}), & j = 2, \ldots, n_{Sv}+1 \\ \gamma_v^{i1} \leq \mu_v^{i1} & \\ \gamma_v^{ij} \leq \mu_v^{ij-1} + \mu_v^{ij} & j = 2, \ldots, n_{Sv} \\ \gamma_v^{in_{Sv}} \leq \mu_v^{in_{Sv}} & \\ \sum_{j=1}^{n_{Sv}} \mu_v^{ij} = 1 & \\ \sum_{j=1}^{n_{Sv}+1} \gamma_v^{ij} = 1 & \\ x_v^i = \sum_{j=1}^{n_{Sv}+1} \gamma_v^{ij} A_v^j & \end{cases} \quad (\mathcal{P}-\mathrm{C})$$

Observe that the first and second inequalities determine the upper and lower limits for the parametrization of each segment of $\mathcal{P}_v$. If $\mu_v^{ij} = 0$ the inequalities are always fulfilled and there is no entry or exit point in the jth segment of the polygonal v. On the contrary, if $\mu_v^{ij} = 1$ then $\lambda_v^i \in [j, j+1]$ meaning that the corresponding entry or exit point is in the jth segment of the polygonal $\mathcal{P}_v$. The third, fourth and fifth inequalities link μ_v^{ij} and γ_v^{ij} variables (and thus implicitly λ_v^i):they state that the variable γ_v^{ij} that gives the representation of a point x_v^i on the line segment j is active (non-null) only if this line segment is chosen (to enter of exit), i.e., $\mu_v^{ij} = 1$. The sixth equation sets that only one line segment is chosen for entering or leaving each polygonal chain. Finally, the seventh equation and eighth inequality set the representation of x_v^i as a convex combination of the extreme points of the adequate line segment.

In addition, we assume that the tour must traverse at least some given percentage α_v of each polygonal chain total length. Denoting by $\lambda_v^{\min}$ and $\lambda_v^{\max}$ the parameter values of λ representing the initial and final points of $\mathcal{P}_v$, respectively, we can model that condition by the following absolute value constraint:

$$|\lambda_v^1 - \lambda_v^2| \geq n_{S_v}\alpha_v \Longleftrightarrow \begin{cases} \lambda_v^1 - \lambda_v^2 = \lambda_v^{\max} - \lambda_v^{\min} \\ \lambda_v^{\max} + \lambda_v^{\min} \geq \alpha_v n_{Sv} \\ \lambda_v^{\max} \leq n_{Sv}(1-u_v) \\ \lambda_v^{\min} \leq n_{Sv} u_v. \end{cases} \quad (\alpha-\mathrm{C})$$

The above modelling assumptions are sufficient to address the range of situations that we want to model. Obviously, they could be more general at the price of not being easy to implement with off-the-shelf solvers.

2.1. Some interesting particular cases

Three very interesting well-known models appear as particular cases of the problems that can be modelled within our framework. If the element associated with each vertex is a single point the problem reduces to the standard traveling salesman problem. If the element associated with each vertex $v \in V$ is a segment $\mathcal{P}_v = [x_v^1, x_v^2]$ and $\alpha_v = 1$, then XPPN becomes the classical Rural Postman Problem in which the edges (x_v^1, x_v^2) are required, in the complete graph induced by these vertices with edge lengths given by the Euclidean norm distance (see Orloff, 1974). In addition, if $\alpha_v \neq 1$, $\forall v \in V$ then XPPN is an extension of the RPP where some edges are only partially required. On the other hand, if the considered neighborhoods are big enough so that $\cap_{v \in V} \mathcal{C}_v \neq \emptyset$, then the problem reduces to finding a degenerate one-vertex tour and the solution to the XPPN is that vertex with cost 0. Finally, if all elements $\mathcal{N}_v$, $v \in V$ are neighborhoods we obtain the Traveling Salesman problem with Neighborhoods (see Arkin & Hassin, 1994).

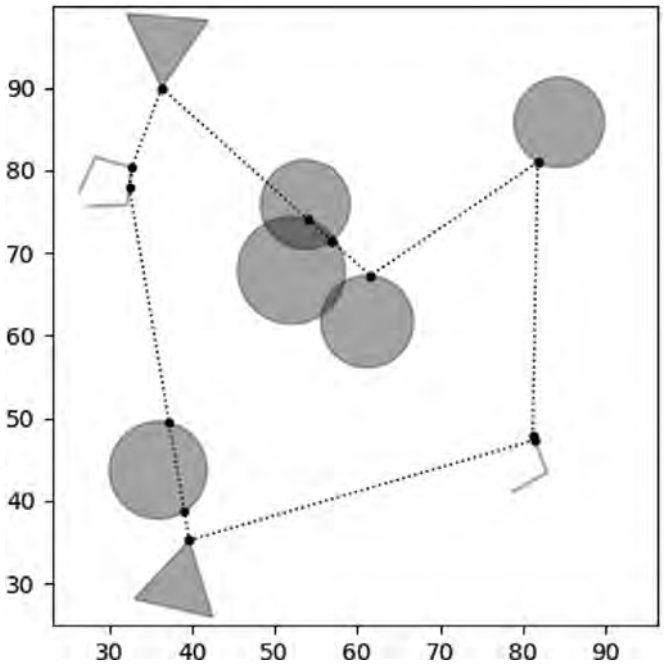


Fig. 2. An example with 9 elements: 7 convex sets and 2 polygonal chains.

Fig. 2 shows an example of the solution obtained for a case in which the elements are circles, triangles and we also have two polygonal chains to visit in our required route.

The discussion above allows us to state the complexity of the XPPN.

Theorem 1. *The decision version of the problem XPPN, given a length L deciding whether the graph G has a XPPN tour of length at most L, is NP-complete.*

The proof follows using a reduction from TSP that as shown above is a particular case of this problem.

3. Mixed integer non linear programming formulations

In this section we present alternative MINLP formulations for the XPPN that will be compared computationally in Section 7. First, we start with a time dependent formulation that allows us to include a number of specific characteristics in the modeling phase such as time dependent travel distances, time windows or time dependent discount factors. Then, we give another formulation that does not make reference to stages in the routes and that simplifies the model at the price of losing some of the above mentioned characteristics.

3.1. A time dependent formulation

One way to model the drone route in our problem is to make variables dependent on the index of the stage when an element is visited in the sequence of visited elements. Thus, this formulation requires binary variables depending on the index order when

they are chosen. Since variables depend on time parameters in the problem, weighting factors for visiting the neighborhoods (b_v^t) and distances d^t (as proxy for travel times β^t), can also be dependent on the stage when they are used.

To model the problem, we introduce a binary variable y_v^t to indicate that the element associated with vertex v is visited by the drone at stage t. In addition, we define the following variables:

- y_v^t: Binary variable whose value is one when v is visited at the tth position in the route sequence and zero otherwise.
- z_{vw}^t: Binary variable that is one when v and w are visited consecutively, assuming that v is visited at the stage t and zero otherwise.
- $z_{vw}^t = y_v^t y_w^{t+1},\ v \neq w$.
- d_{vw}^t: Continuous variable that represents the distance between pairs of chosen points v, w from different components at the stage t.
- d_v^t: Continuous variable that represents the distance between two consecutive points within the same component associated with $v \in V$ at the stage t.
- $\lambda_v^1,\ \lambda_v^2$: Continuous variables determining the position of x_v^1 and x_v^2, respectively, in the polygonal chain $\mathcal{P}_v$.

Using these variables, the first formulation follows:

$$\min \quad \sum_{t=1}^{|V|} \sum_{v \neq w} d_{vw}^t z_{vw}^t + \sum_{t=1}^{|V|} \sum_{v \in V} f_v^t d_v^t \tag{2a}$$

$$\text{s.t.} \quad d_{vw}^t \geq \beta_{vw}^t \|x_v^2 - x_w^1\|, \qquad \forall v \neq w \tag{2b}$$

$$d_v^t \geq \beta_v^t \|x_v^1 - x_v^2\|, \qquad \forall v \in V \tag{2c}$$

$$\sum_{v \in V} y_v^t = 1, \qquad \forall t \tag{2d}$$

$$\sum_{t=1}^{|V|} y_v^t = 1, \qquad \forall v \in V \tag{2e}$$

$$y_v^t + y_w^{t+1} - 1 \leq z_{vw}^t, \qquad \forall v \neq w,\ t = 1, \ldots, |\mathcal{C}| - 1 \tag{2f}$$

$$(\mathcal{C} - \mathrm{C}), (\mathcal{P} - \mathrm{C}), (\alpha - \mathrm{C}) \tag{2g}$$

The first addend of the objective function (2a) includes the drone traveling distance among different elements while the second one accounts for the distances between the entry and exit points of each component taking into account the weighting factor for traveling within this component at the stage t. Constraints (2d) and (2e) state, respectively, that in each stage the route visits one element and each component is traversed once and only once. Constraint (2f) is obtained by linearizing z_{vw}^t and ensures that if we travel from v to w, assuming that we are in v at the instant t, then we visit v in t and w in $t+1$. Constraint (2g) refers to the domain of the entry and exit points of each element in the problem, as well as the minimal required percentage of the polygonal chain length that must be traversed by the drone. They were defined in Section 2.

Despite the versatility of this formulation for capturing actual characteristics of drone routes, its drawback comes from the three index dimension of its variables which makes it difficult to handle medium size instances. In the next section, we shall simplify this formulation making it independent of time at the price of losing some of its time-dependent characteristics.

3.2. Non-time dependent formulations

The simplification mentioned above can be performed, based on the rationale of ensuring connectivity on the graph G, through different sets of inequalities. In particular, we compare Miller-Tucker-Zemlin (MTZ) inequalities and subtour elimination constraints (SEC). All formulations use the following sets of decision variables:

- Binary variables $z_e \in \{0, 1\},\ e \in E_{out}$, to represent the edges of the tours.
- Continuous variables $d_e \geq 0,\ e = \{v, w\} \in E_{\text{out}} \subseteq E$, to represent the distance $d_{vw}(x_v^1, x_w^2)$ between the pairs of selected points of different elements (neighborhoods) and $d_v \geq 0,\ v \in V$, to represent the distance $d_v(x_v^1, x_v^2)$ between the pairs of points of the same element.

Let

$$\mathcal{D}_e = \{d \in \mathbb{R}_+^{|E_{\text{out}}|} : d_e \geq d_{vw}(x_v^1, x_w^2), \forall e = (v, w) \in E_{\text{out}},\ x \in \mathcal{X}\},$$

$$\mathcal{D}_v = \{d \in \mathbb{R}_+^{|E_{\text{in}}|} : d_v \geq d_v = (x_v^1, x_v^2), \forall v \in V,\ x \in \mathcal{X}\},$$

denote the domains for the feasibility of the d variables. The reader can see that these sets, namely $\mathcal{D}_e$ and $\mathcal{D}_v$, can be alternatively described using the constraints

$$\|x_v^1 - x_w^2\|_2 \leq d_e, \quad \forall e = \{v, w\} \in E_{\text{out}}, \tag{D$_1$}$$

$$\|x_v^1 - x_v^2\|_2 \leq d_v, \quad \forall v \in V, \tag{D$_2$}$$

$$x \in \mathcal{X}, \tag{D$_3$}$$

which set the distance values and impose that x belongs to its suitable neighborhood.

Then, a generic bilinear formulation for XPPN is

$$\begin{aligned} \min \quad & \sum_{e \in E_{\text{out}}} d_e z_e + \sum_{e \in E_{\text{in}}} f_v d_v \\ \text{s.t.} \quad & z \in \mathcal{T}_G, \\ & \text{(D1)}, \text{(D2)} \\ & (\mathcal{C} - \mathrm{C}), (\mathcal{P} - \mathrm{C}), (\alpha - \mathrm{C}) \end{aligned} \tag{Pdz}$$

The reader should observe that, as already mentioned, the above formulation is bilinear since the first term of the objective function contains products of variables of the form $d_e z_e$, for $e \in E_{\text{out}}$.

Next, we use McCormick's envelopes (McCormick, 1976) for the linearization of those bilinear terms of the objective function. We define additional variables $p_e \geq 0,\ e \in E_{\text{out}}$ that stand for that product.

Replacing the products by the new variables and introducing a new set of constraints enforcing the correct representation, we obtain the following formulation:

$$\min \quad P = \sum_{e \in E_{\text{out}}} p_e + \sum_{v \in V} f_v d_v \tag{RL-XPPN}$$

$$\begin{aligned} \text{s.t.} \quad & p_e \geq d_e - M_e(1 - z_e) \quad \forall e \in E_{\text{out}} \qquad \text{(LIN-Mc)} \\ & p_e \geq 0,\ \forall e \in E_{\text{out}} \\ & z \in \mathcal{T}_G \\ & \text{(D1)}, \text{(D2)} \\ & (\mathcal{C} - \mathrm{C}), (\mathcal{P} - \mathrm{C}), (\alpha - \mathrm{C}) \end{aligned}$$

Here M_e denotes an upper bound of the distance between the sets that are joined by e.

Furthermore, this formulation can be reinforced by adding some valid inequalities: $p_e \geq m_e z_e, \forall e \in E_{\text{out}}$ and $d_v \leq M_v, \forall v \in V$, where m_e and M_v are bounds that are adjusted in Section 5. The first family of valid inequalities sets lower bounds on the values for p_e

whereas the second ones sets upper bounds on the distances traveled by the drone within each neighborhood.

The above discussion leads us to strengthen a generic formulation for XPPN. This formulation will be particularized once the connectivity condition of the solutions is specifically introduced in the model.

$$\begin{array}{lll} \min & P = \sum_{e\in E_{\text{out}}} p_e + \sum_{v\in V} f_v d_v & \\ \text{s.t.} & p_e \geq d_e - M_e(1-z_e) & \forall e \in E_{\text{out}} \quad \text{(LIN-Mc)} \\ & p_e \geq m_e z_e & \forall e \in E_{\text{out}} \quad \text{(VI-1)} \\ & d_v \leq M_v & \forall v \in V \quad \text{(VI-2)} \\ & z \in \mathcal{T}_G & \\ & \text{(D1), (D2)} & \\ & (\mathcal{C}-\text{C}), (\mathcal{P}-\text{C}), (\alpha-\text{C}) & \end{array}$$

The two formulations that we present below differ from one another in the family of constraints used to enforce connectivity. One of them is by the family of subtour elimination constraints (SEC), Edmonds (2003). The other one relies on a compact formulation based on the well-known Miller-Tucker-Zemlin (MTZ) constraints, Miller, Tucker, & Zemlin (1960).

3.2.1. A valid formulation for XPPN based on SECs

The family of SEC is well-known in combinatorial optimization. It enforces connectivity by imposing that the number of edges among any subset of vertices can not exceed its cardinality minus one. Augmenting these constraints into the generic formulation presented above we obtain the following valid formulation for XPPN:

$$\begin{array}{lll} \min & P = \sum_{e\in E_{\text{out}}} p_e + \sum_{v\in V} f_v d_v & \text{(SEC-XPPN)} \\ & \text{(LIN-Mc), (VI-1), (VI-2)} & \\ & \sum_{w\in V\setminus\{v\}} z_{vw} = 1, & \forall v \in V \quad \text{(C}_1\text{)} \\ & \sum_{w\in V\setminus\{v\}} z_{wv} = 1, & \forall v \in V \quad \text{(C}_2\text{)} \\ & \sum_{e=(v,w):v,w\in S} z_e \leq |S|-1, & \forall S \subsetneq V \quad \text{(SEC)} \\ & \text{(D1), (D2)} & \\ & (\mathcal{C}-\text{C}), (\mathcal{P}-\text{C}), (\alpha-\text{C}) & \end{array}$$

Assignment Constraints (C_1) and (C_2) ensure that the drone enters and exits each component of the problem exactly once. Constraint (SEC) prevents the existence of subtours. This constraint forces that in any subset S of nodes included in V there can not be more edges between nodes in S than its number of nodes minus one, thus avoiding the existence of cycles.

Since there is an exponential number of SEC constraints, when we implement this formulation we need to perform a row generation procedure including constraints, whenever they are required, by a separation oracle. To find SEC inequalities, as usual, we search for disconnected components in the current solution. Among them, we choose the shortest subtour found in the solution to be added as a lazy constraint to the model.

If the considered distance between components is symmetric, we obtain the symmetric formulation based on SECs, denoted by (sSEC-XPPN). In this formulation, we can halve the number of binary variables and replace constraints (C_1) and (C_2) in (SEC-XPPN) by the following connectivity restrictions:

$$\sum_{w\in V\setminus\{v\}} z_{wv} = 2, \qquad \forall v \in V.$$

3.2.2. XPPN formulation based on the Miller-Tucker-Zemlin inequalities

This section addresses an alternative formulation that results replacing SEC inequalities by the so called Miller-Tucker-Zemlin constraints (see Miller et al., 1960). In this formulation, we introduce the integer variable s_v to generate an alternative formulation that eliminates the subtours and the exponential number of inequalities of (SEC-XPPN).

$$\begin{array}{lll} \min & P = \sum_{e\in E_{\text{out}}} p_e + \sum_{v\in V} f_v d_v & \text{(MTZ-XPPN)} \\ \text{s.t.} & \text{(LIN-Mc), (VI-1), (VI-2)} & \\ & \sum_{w\in V\setminus\{v\}} z_{vw} = 1, & \forall v \in V \quad \text{(C}_1\text{)} \\ & \sum_{w\in V\setminus\{v\}} z_{wv} = 1, & \forall v \in V \quad \text{(C}_2\text{)} \\ & |V| z_{vw} + s_v - s_w \leq |V|-1, & \forall e=(v,w) \in E_{\text{out}} \quad \text{(MTZ}_1\text{)} \\ & s_1 = 1 & \text{(MTZ}_2\text{)} \\ & 2 \leq s_v \leq |V|, & \forall v \in V \quad \text{(MTZ}_3\text{)} \\ & s_v - s_w + |V| z_{wv} \leq |V|-1, & \forall e=(v,w) \in E_{\text{out}}, w>1 \quad \text{(MTZ}_4\text{)} \\ & s_v - s_w + (|V|-2) z_{wv} \leq |V|-1, & \forall e=(v,w) \in E_{\text{out}}, v>1 \quad \text{(MTZ}_5\text{)} \\ & \text{(D1), (D2)} & \\ & (\mathcal{C}-\text{C}), (\mathcal{P}-\text{C}), (\alpha-\text{C}) & \end{array}$$

Again constraints (C_1) and (C_2) require that in each feasible solution only one edge departs from node v and only one edge enters at node v for any $v \in V$, respectively. It is well-known that constraints (MTZ_1)-(MTZ_3) (see Miller et al., 1960) model the elimination of subtours. The constraints (MTZ_1)-(MTZ_3) enforce connectivity, i.e., that there is only a single tour covering all vertices. The constraints (MTZ_4) and (MTZ_5) define the intermediate conditions for the tour that may improve the performance of this formulation over the formulation based on subtour elimination constraints (see Sawik (2016) for more details).

Now we state a result related to the relationship between the SEC and MTZ polytopes of our formulations of the XPPN, that is, the feasible regions of the respective LP relaxations of these models.

Theorem 2. *The SEC polytope is contained in the MTZ polytope for the XPPN.*

Proof. Observe that the only difference between these two polytopes is the family of constraints that ensures the elimination of subtours. Therefore, it is enough to see that the (SEC) constraints are stronger than those given in (MTZ_1)-(MTZ_3) which is proved in Velednitsky (2017). □

4. A heuristic algorithm for XPPN

In this section we present a heuristic algorithm for solving XPPN. This algorithm has two different applications. On the one hand, it provides good quality feasible solutions for XPPN that become a promising alternative to exact methods whenever the size of the problems is large. On the other hand, it also helps in solving exactly XPPN by feeding the exact formulations with a good initial solution which in turns speeds up the branch and bound search. The considered algorithm is composed by two phases: the Clustering Phase and the Variable Neighborhood Search (VNS) Phase. The so called clustering phase determines some points in each dimensional element (polygonal chain or neighborhood) and then the VNS phase finds a heuristic tour on the complete graph spanned by the previously obtained points.

The clustering phase

The first phase of the heuristic algorithm is based on solving a relatively easy single facility location problem: the Weber or median Problem. The solution of this problem looks for a prototype point (a representative) x_v, $v \in V$ of the dimensional elements in the problem (neighborhoods and polygonal chains) and another

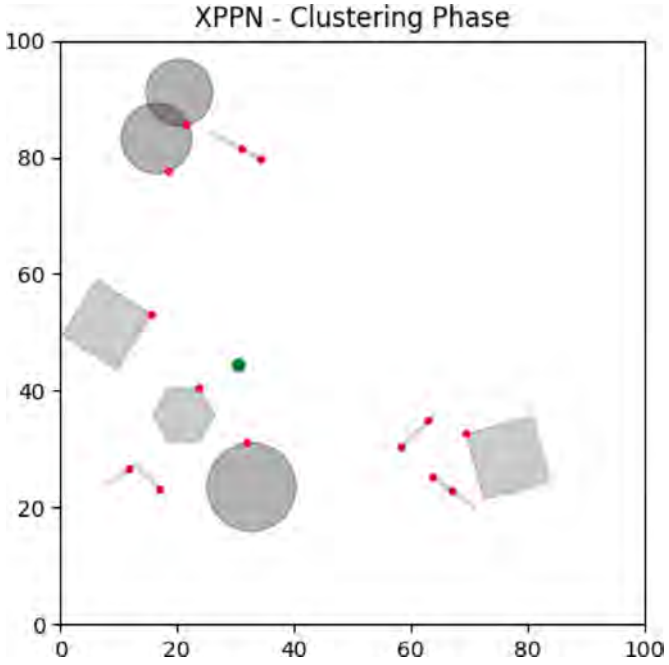


Fig. 3. Illustration of the first phase of the heuristic algorithm.

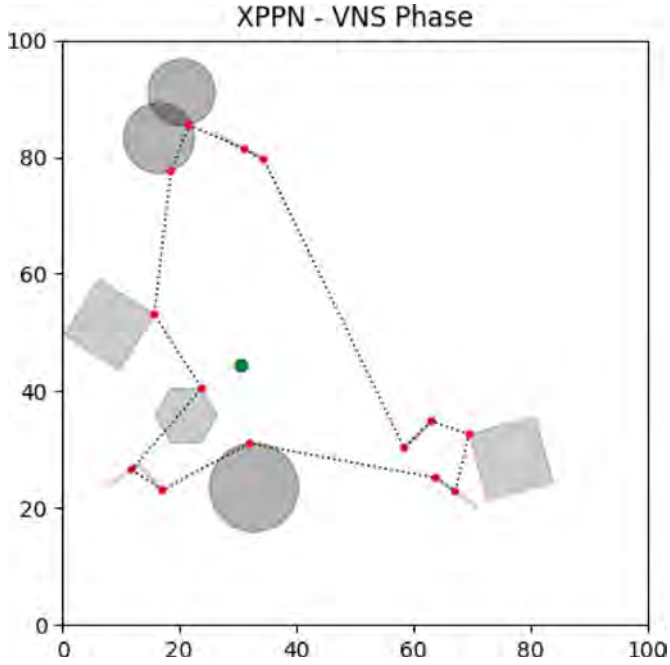


Fig. 4. Application of the VNS phase to the example of Fig. 3.

point, Med, so that the sum of the Euclidean distances from x_v to Med is minimized.

$$\min \quad \sum_{v \in V} \|x_v - \text{Med}\| \qquad \text{(Weber)}$$
$$\text{s.t.} \quad (\mathcal{C} - \text{C}), (\mathcal{P} - \text{C}), (\alpha - \text{C})$$

The idea of this approach is to find some points that are likely to be close to the true chosen points in each element in the final optimal drone route. Fig. 3 shows an example that combines six neighborhoods and four polygonal chains. Red points represent the points of each set and the green point is the proposed 1-median obtained after solving the corresponding Weber problem described above.

The variable neighborhood search phase

Once the points of each set have been chosen, the idea is to find the minimal cost drone route that joins these points. To obtain this route, we have used the well-known and general Variable Neighborhood Search metaheuristic developed in Mladenović & Hansen (1997). The Python implementation code has been taken from Pereira (2018). In that implementation, the distance matrix is computed by taking the Euclidean distances between each pair of points. In our case, we had to modify it because our distance matrix requires also distances computed along the different considered polygonals.

Using the example depicted in Fig. 3, we generate a tour considering this VNS approach with a maximum number of 25 attempts, a neighborhood of size 5 and 10 iterations. The final result is shown in Fig. 4.

Finally, in order to build a feasible solution for XPPN we take into account the position of the points (represented by x_v^1 and x_v^2) and the order in which they are visited in the tour obtained by the VNS phase of our heuristic (represented by z_{vw}). Once the solution is built, it can also be taken as an initial solution for any of the exact formulations presented above. In the following, we present the pseudo-code of this heuristic:

Algorithm 1: Heuristic for solving XPPN.

Let $\{\mathcal{N}_v : v \in V\}$ be the neighborhood set.Set $attempts = 25$, $neigh\_size = 5$, $iter = 10$.

1. Solve the Weber problem for $\mathcal{N}_v$ to get $\bar{x}$.
2. Consider the VNS approach with parameters *attempts*, *neigh_size* and *iter* and points $\bar{x}$ to obtain the order of visit to the neighborhoods $\bar{z}$.

5. Strengthening the formulation of XPPN

5.1. Pre-processing

In this section we explore the geometry of the neighborhoods that appear in the problem to fix a priori some variables and to increase the efficiency of the model.

First of all, we consider two special cases that relate the position of the entry and exit points of each neighborhood with the coefficient f_v of the objective function.

Remark. If the problem verifies that $f_v = 0$ for all $v \in V_{\mathcal{C}}$, then the entry and exit points x_v^1 and x_v^2 selected in each neighborhood are the same that the ones obtained by minimizing the distance between the neighborhoods.

Remark. If $f_v \geq 1$ for some $v \in V_{\mathcal{C}}$, then, there exists an optimal solution verifying $x_v^1 = x_v^2$.

Proof. Let us consider an optimal route and let p be the path in that route that visits $\mathcal{C}_v$. Assume without loss of generality that to visit $\mathcal{C}_v$, the route departs of the previous element $\mathcal{C}_u$ from x_u^2, enters $\mathcal{C}_v$ through x_v^1 and exits from x_v^2 where $x_v^1 \neq x_v^2$ and $f_v \geq 1$. Assume again without loss of generality that after visiting $\mathcal{C}_v$ the route goes to $\mathcal{C}_w$ entering by x_w^1. Let us consider the alternative path p' formed by x_u^2, the midpoint x_v' between x_v^1 and x_v^2 and x_w^1. The contribution of visiting $\mathcal{C}_v$ in the objective function of the problem will be

$$\begin{aligned} length(p') &= d(x_u^2, x'_v) + d(x'_v, x_w^1) \\ &\leq d(x_u^2, x_v^1) + d(x_v^1, x'_v) + d(x'_v, x_v^2) + d(x_v^2, x_w^1) \\ &= d(x_u^2, x_v^1) + d(x_v^1, x_v^2) + d(x_v^2, x_w^1) \\ &\leq d(x_u^2, x_v^1) + f_v d(x_v^1, x_v^2) + d(x_v^2, x_w^1) \\ &= length(p), \end{aligned}$$

but p is an optimal path to visit those elements in this optimal solution which turns all the above inequalities into equalities. Therefore, the path p' is also an optimal path within that optimal solution. However, by construction on $\mathcal{C}_v$, p' has the same entry and exit point which proves the claim. □

From now on, we assume in the rest of this section that $f_v \geq 1$ for all $v \in V$. The following outcome restricts the domain where the selected points can be located.

Proposition 1. *There exists always an optimal solution of the XPPN whose selected points are placed in the boundary of the neighborhoods.*

Proof. If the number of elements of the problem is two, the problem consists of calculating the minimum distance between two convex sets and it is known that the selected points are clearly located in the boundary of the sets if they do not overlap and can be chosen in the border in case of overlapping. If the number of neighborhoods is more than two, we can reduce the proof to analyze three consecutive elements. Let T be the triangle spanned by the point $x_u \in \mathcal{C}_u$ of the previously visited neighborhood, the point x_v in the neighborhood $\mathcal{C}_v$ and the point $x_w \in \mathcal{C}_w$ of the next neighborhood to be visited in an optimal sequence. We could have three possible cases depending on the number of points allocated in the boundary. If x_u, x_v, x_w are aligned, for each point that is not in the boundary, namely $\mathcal{C}_v$, we can consider the closest point obtained by the intersection of the line generated by x_u and x_v with the boundary of C_v, $\partial\mathcal{C}_v$. This point is also aligned with the others and its contribution to the objective function is the same as the one given by x_v. Therefore, let assume that these points are not aligned. We have three cases:

- Case 1: Suppose that only $x_v \in \mathcal{C}_v$ is not in the boundary. Let $\overline{x_u x_w}$ be the line segment that joins x_u and x_w. Let suppose, for the sake of contradiction, that there exists a neighborhood $\mathcal{C}_v$ whose selected point in the optimal sequence is in the topological interior of $\mathcal{C}_v$, i.e., $x_v \in int(\mathcal{C}_v)$. The idea of the proof is to find another point in the boundary of $\mathcal{C}_v$ closer to x_u and x_w. We consider the point $x'_v = r^{\perp} \cap \partial\mathcal{C}_v$, where $r^{\perp}$ is the perpendicular line to the line segment $\overline{x_u x_w}$ and $\partial\mathcal{C}_v$ is the boundary of $\mathcal{C}_v$. Observe that this intersection produces two points in $\partial\mathcal{C}_v$: among them we take the closest one to $\overline{x_u x_w}$. If we call T' the triangle generated by x_u, x'_v and x_w, then the height of T' to $\overline{x_u x_w}$ is smaller than the one of T. Hence by the Pythagoras Theorem, $\overline{x_u x'_v} < \overline{x_u x_v}$ and $\overline{x'_v x_w} < \overline{x_v x_w}$, which is a contradiction.
- Case 2: Assume that $x_u \in \mathcal{C}_u$ and $x_v \in \mathcal{C}_v$ are not in the boundary. We can take $x'_u = \overline{x_u x_v} \cap \partial\mathcal{C}_u$. This point is closer to x_v than x_u and it is in the boundary of $\mathcal{C}_u$. Therefore, we have two points in the boundary and we can apply the previous case to conclude that x_v must be in ∂C_v too.
- Case 3: Finally, suppose that no point is in the boundary of each neighborhood. Again, we can construct $x'_u = \overline{x_u x_v} \cap \partial\mathcal{C}_u$ that is closer to x_v and x_w. Then, we have a point in the boundary and Case 2 can be applied to the rest of points. □

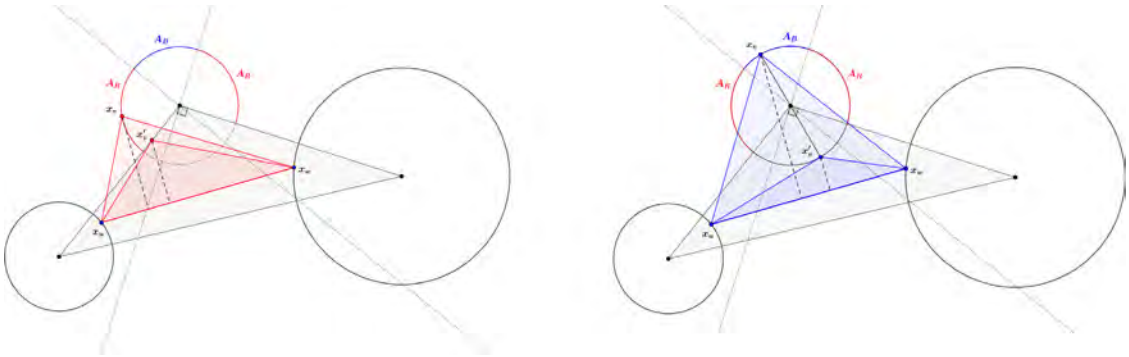
The special case in which all the neighborhoods are circles, allows us to limit even more the location of the points based on the construction given in the Proposition 1.

Corollary 2.1. *Any point selected in an optimal solution of the XPPN when all the neighborhoods are circles is placed in some arc of one of the circumferences inside of the convex hull generated by the center of the circles.*

Proof. If we have two neighborhoods, the selected points are located in the line segment that joins the center of the circles and the result follows. If the number of neighborhoods is more than two, we can reduce the proof to analyze three consecutive elements. Let T be the triangle spanned by the point $x_u \in \mathcal{C}_u$ of the previously visited neighborhood, the point x_v in the neighborhood $\mathcal{C}_v$ and the point $x_w \in \mathcal{C}_w$ of the next neighborhood in an optimal sequence. Let assume that we have two points inside the convex hull and $x_v \in C_v$ does not satisfy this property. Let also $\overline{x_u x_w}$ be the line segment that joins x_u and x_w. We distinguish two cases depending on the location of x_v in the neighborhood:

- If x_u, x_v, x_w are aligned, it is straightforward to conclude that x_v is in the convex hull of the centers.
- If x_u, x_v, x_w are not aligned, let assume that in N_v its selected point is not in the convex hull $\mathcal{C}_v$ of the centers of the neighborhoods. The idea of the proof is to find another point in the boundary of the convex hull $\mathcal{C}_v$ whose distance to x_u and x_w is smaller than the distance from x_v. We split the boundary of C_v (circumference) in two arcs A_R and A_B. These arcs are built by taking the perpendicular line to the edge of the convex hull $\mathcal{C}$:
  - If $x_v \in A_R$, we take x'_v the projection to the convex hull and it produces a triangle T' with lower height to $\overline{x_u x_w}$. Then, we can use the Proposition 1 to construct a point in the boundary of $\mathcal{C}_v$ that lies in the convex hull. (See Fig. 5.1).
  - If $x_v \in A_B$, we construct x'_v the diametrically opposite point of x_v in N_v. This point also produces a smaller height that contradicts the assumption that x_v gives the shortest tour. (See Fig. 5.2)

If the number of points outside the convex hull is more than one, we can apply this procedure iteratively to include these points in the convex hull generated by the center of the circles. □



Finally, we conclude this section giving another result that allows one to eliminate some neighborhoods and thus simplify the problem without modifying the objective value of the problem.

Proposition 2. *Given two neighborhoods A and B, if $B \supset A$, then B can be removed in the problem.*

Proof. Starting from the optimal solution of problem without B, we are going to build an optimal solution including B that is essentially the same. Let z^* the optimal tour by deleting the neighborhood B in the problem. By connectivity, there exist two neighborhoods A_{-1} and A_{+1} that are connected with A, i.e., such that $z^*_{A_{-1}A} = z^*_{AA_{+1}} = 1$. In addition, let x^*_A be the point chosen to visit the neighborhood A. If we include $B \supset A$ in the problem and we fix $x_B = x^*_A$ and $z_{AB} = z_{BA_{+1}} = 1$. This solution is also a simple path whose objective value is the same because $d(A, B) = 0$ and $d(B, A_{+1}) = d^*(A, A_{+1})$. □

5.2. Valid inequalities

The different models that we have proposed include in one way or another big-M constants. In order to strengthen the formulations we provide good upper bounds for those constants. In this section we present some results that adjust them for each kind of set considered in our models.

The first big-M constant we need to adjust is M_e that denotes an upper bound of the distance between the sets joined by an edge $e \in E_{\text{out}}$. We have three cases that depend on the shape of the sets A and B:

- If A and B are both ellipsoids, we cannot easily compute the maximum distance between A and B, but we can generate an upper bound of this distance by taking diametrically opposite points of minimum radius circles containing each ellipsoid. When both ellipsoids are circles, this bound coincides with the maximum distance.
- If A is an ellipsoid and B is a polygon or a polygonal chain, we can set this bound by the maximum of the distances of each vertex of B to the center of A plus the radius of the minimum circle that contains the ellipsoid A.
- If A and B are both polygons or polygonal chains, this bound can be computed exactly by taking the maximum of the distances between vertices of A and B.

The second bound to be adjusted is m_e. It denotes a lower bound of the distance of the sets joined by the edge $e \in E_{\text{out}}$. In this case, we can compute this distance exactly by solving a convex program that minimizes the distance between the sets A and B.

In the Figs. 5–7 we show the selected maximal (red) and minimal (blue) bounds depending on the shape of the sets.

In addition, the third bound represents the maximal distance between two points within a given neighborhood. We can compute this upper bound according to the shape of this set (see Fig. 8):

- If the set is an ellipsoid, we can take diametrically opposite points of the minimum radius circle that contains this ellipsoid.

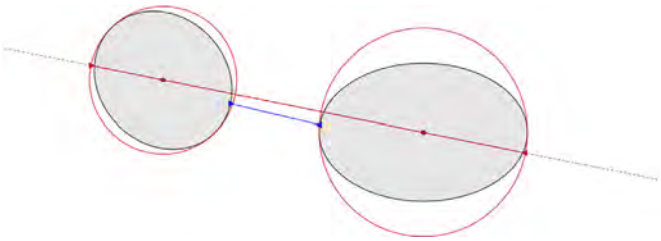


Fig. 5. Upper and lower bound when both sets are ellipsoids.

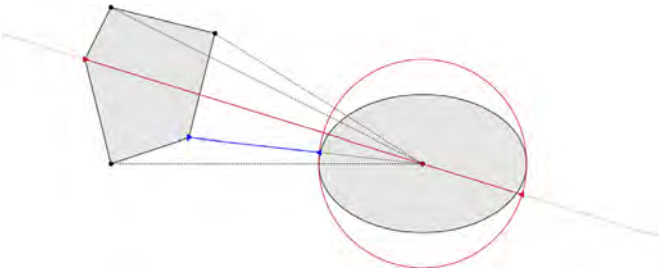


Fig. 6. Upper and lower bound when a set is a polygon and the other is an ellipsoid.

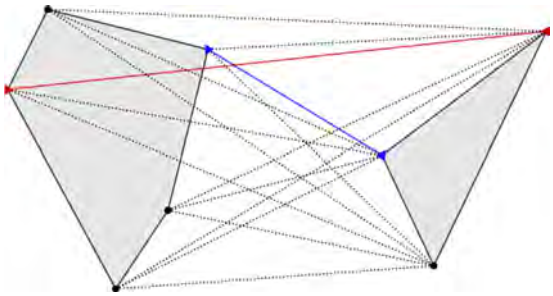


Fig. 7. Upper and lower bound when both sets are polygons.

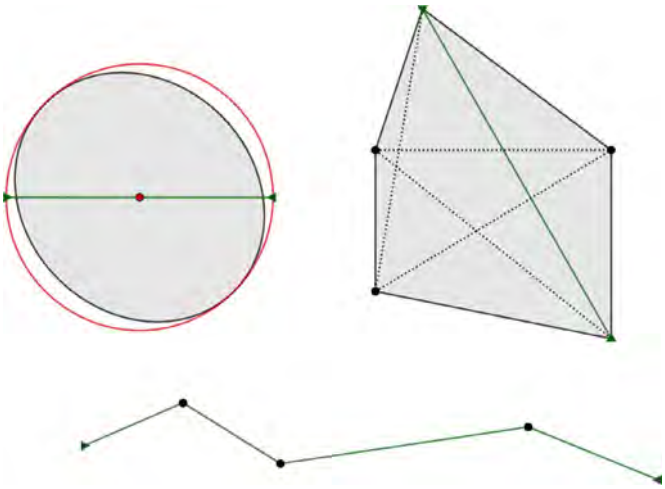


Fig. 8. Upper bound on the maximal distance within a set.

- If the set is a polygon, we can compute the maximum of the distances between each pair of vertices.
- If the set is a polygonal chain, this bound equals the length of the polygonal.

6. A decomposition algorithm

In this section we present an alternative row generation approach to solve the XPPN based on a Benders decomposition of the problem. The general method is based on the following observation: If we fix $z \in \mathcal{T}_G$ in the generic formulation of XPPN, we obtain a continuous SOC problem, which is well-known to be convex. On the other hand, the objective function that we are considering is bilinear. Hence, we can use a Benders-like decomposition approach (see Benders, 1962) to generate an iterative algorithm that solves this problem.

For a given $\bar{z} \in \mathcal{T}_G$, the "optimal" vertices and distances of its associated XPPN can be computed by solving the following sub-

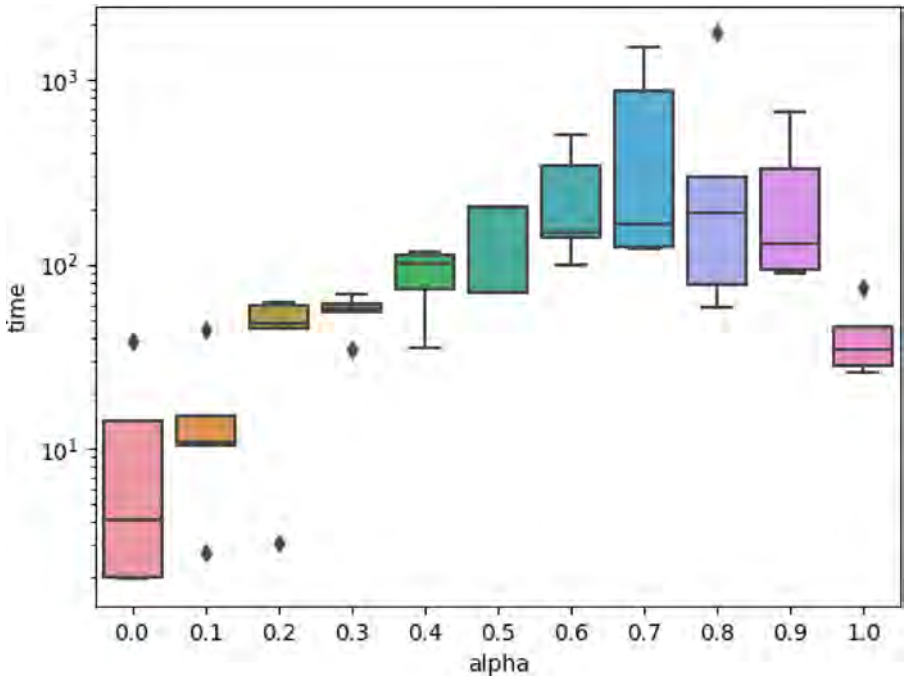


Fig. 9. Comparison of the times of the MTZ formulation varying the α parameter.

problem:

$$\min \quad d(\bar{z}) = \sum_{e\in E_{\text{out}}} d_e \bar{z}_e + \sum_{v\in V} f_v d_v \qquad (\text{Pd}\bar{z})$$
$$\text{s.t.} \quad d_e \in \mathcal{D}_e,\ d_v \in \mathcal{D}_v.$$

Note that the number of d variables in (Pd$\bar{z}$) is $2|V|$, because only distances with nonzero $\bar{z}_e$ variables need to be calculated. Thus, Benders decomposition is a good approach for solving the XPPN problem based on our formulations (see Blanco et al., 2017). The explicit form of the Benders cuts is the following:

$$P \geq d(\bar{z}) + \sum_{e:\bar{z}_e=1} M_e(z_e - 1) + \sum_{e:\bar{z}_e=0} m_e z_e \qquad (4)$$

where $P = \sum_{e\in E_{\text{out}}} p_e + \sum_{v\in V} f_v d_v$ with $p_e \geq 0$ and M_e and m_e are the upper and lower bounds estimated in the above section.

Therefore, the relaxed master problem at the kth iteration of the row-generation algorithm can be stated as:

$$P^* = \min \quad P$$
$$P \geq d(\bar{z}^k) + \sum_{e:\bar{z}^k_e=1} M_e(z^k_e - 1) + \sum_{e:\bar{z}^k_e=0} m_e z^k_e,\ k = 1,\dots,K, \quad (5)$$

$$P = \sum_{e\in E_{\text{out}}} p_e + \sum_{v\in V} f_v d_v \qquad (6)$$
$$z \in \mathcal{T}_G.$$

Adding the above cuts sequentially gives rise to the solution scheme described in Algorithm 2:

Observe that in the while loop we set the stopping criterion as the maximum allowed gap between the upper and lower bound: this gap cannot exceed the fixed threshold value ε.

Theorem 2.4 in Geoffrion (1972) states the finite convergence of the decomposition approach under the following assumptions: convexity and finiteness of the feasible domains, closeness of the "linking" constraints between the sets, and convexity of the objective functions. In our case, the finiteness of the number of underlying tours of $\mathcal{T}_G$, the convexity of (Pd$\bar{z}$) for any $\bar{z} \in \mathcal{T}_G$, and the linear separability of the problem allows us to apply the above result, which assures that Algorithm 2 terminates in a finite number of steps (for any given $\varepsilon \geq 0$).

Algorithm 2: Decomposition Algorithm for solving XPPN.

Initialization: Let $z^0 \in \mathcal{T}_G$ be an initial solution and ε a given threshold value.
Set $LB = 0$, $UB = +\infty$, $\bar{z} = z^0$.

while $|UB - LB| > \varepsilon$ **do**

1. Solve (Pd$\bar{z}$) for $\bar{z}$ to get $d(\bar{z})$.
2. Add the cut $P \geq d(\bar{z}) + \sum_{e:\bar{z}_e=1} M_e(z_e - 1) + \sum_{e:\bar{z}_e=0} m_e z_e$ to the current master problem.
3. Obtain the optimal value $\bar{P}$ to the current master problem, and its associated solution $\bar{z}$.
4. Update $LB = \max\{LB, \bar{P}\}$ and $UB = \min\{UB, \sum_{e\in E} d_e(\bar{z})\bar{z}_e + \sum_{v\in V} f_v d_v\}$

end

To avoid the enumeration of all tours of $\mathcal{T}_G$, we have initialized the algorithm with a non-empty set of randomly generated cuts which give a suitable initial representation of the lower envelope of P.

Given that the master problem exhibits a combinatorial nature, we have embedded the cut generation mechanism within a branch-and-cut scheme.

7. Computational experiments

7.1. Data generation

In this section we have performed a series of experiments to compare the formulations presented in Sections 3 and 6. Since no benchmark instances are available in the literature for this problem, based on the work of Blanco et al. (2017), we have generated five instances with a number $|V| \in \{5, 10, 15, 20\}$ of neighborhoods

Table 1
Computational comparison between MTZ formulation with and without initial solution.

Size	Radii	Mode	Final Gap (Init)	Final Gap (NoInit)	Opt. Time (Init)	Opt. Time (NoInit)
5	1	1	0.0	0.0	0.61	0.45
5	1	2	0.0	0.0	0.12	0.08
5	1	3	0.0	0.0	0.21	0.13
5	1	4	0.0	0.0	0.25	0.27
5	2	1	0.0	0.01	0.27	0.4
5	2	2	0.0	0.0	0.13	0.11
5	2	3	0.0	0.0	0.17	0.14
5	2	4	0.0	0.0	0.17	0.19
5	3	1	0.0	0.0	0.44	0.42
5	3	2	0.0	0.01	0.16	0.12
5	3	3	0.0	0.0	0.17	0.16
5	3	4	0.0	0.0	0.3	0.35
5	4	1	0.0	0.01	0.34	0.4
5	4	2	0.0	0.0	0.14	0.34
5	4	3	0.0	0.0	0.16	0.16
5	4	4	0.0	0.0	0.28	0.3
10	1	1	0.0	0.0	1.93	4.53
10	1	2	0.0	0.0	0.75	0.84
10	1	3	0.0	0.0	0.72	1.77
10	1	4	0.0	0.0	1.52	2.95
10	2	1	0.0	0.0	38.83	61.53
10	2	2	0.0	0.0	14.14	44.93
10	2	3	0.0	0.0	2.23	4.65
10	2	4	0.0	0.0	2.52	7.5
10	3	1	0.0	1.09	487.94	1049.86
10	3	2	0.0	0.0	35.81	153.37
10	3	3	0.0	0.0	13.28	29.43
10	3	4	0.0	0.0	133.81	510.58
10	4	1	19.28	10.0	3513.38	4134.31
10	4	2	0.0	0.0	238.98	1253.21
10	4	3	0.0	0.0	20.25	82.39
10	4	4	0.0	11.75	1142.17	3490.88
15	1	1	0.0	0.0	18.58	196.71
15	1	2	0.0	0.0	3.38	23.37
15	1	3	0.0	0.0	314.57	44.56
15	1	4	0.0	0.0	10.94	90.1
15	2	1	6.69	23.17	3135.29	7200.72
15	2	2	0.0	14.06	2460.78	7200.49
15	2	3	0.0	0.0	14.58	49.51
15	2	4	0.0	5.88	1052.16	4660.88
15	3	1	46.33	59.74	5760.56	7200.28
15	3	2	20.79	31.2	5760.84	7200.8
15	3	3	0.0	0.0	322.77	599.25
15	3	4	14.07	19.17	5865.82	6896.78
15	4	1	100.0	100.0	7200.47	7200.98
15	4	2	20.2	36.9	4421.25	7200.51
15	4	3	0.19	0.72	2195.2	3566.82
15	4	4	21.6	27.71	7200.42	7200.5

Table 2
Computational comparison between MTZ formulation and Benders algorithm for problems with up to 10 neighborhoods.

Size	Radii	Mode	Final Gap (Benders)	Time (Benders)	#Cuts	Final Gap (MTZ)	Time (MTZ)
10	1	1	0.0	15.72	19.0	0.0	1.93
10	1	2	0.0	23.64	55.4	0.0	0.75
10	1	3	0.0	13.22	25.8	0.0	0.72
10	1	4	0.0	33.96	29.8	0.0	1.52
10	2	1	76.1	6430.08	1209.0	0.0	38.83
10	2	2	56.14	4777.58	1009.6	0.0	14.14
10	2	3	0.0	1766.06	380.6	0.0	2.23
10	2	4	10.57	5993.57	804.6	0.0	2.52
10	3	1	96.21	7208.56	1481.2	0.0	487.94
10	3	2	92.16	7203.99	1352.2	0.0	35.81
10	3	3	9.29	5832.51	520.4	0.0	13.28
10	3	4	84.41	7214.86	921.8	0.0	133.81
10	4	1	98.79	7205.35	2283.0	19.28	3513.38
10	4	2	95.53	7207.51	1343.4	0.0	238.98
10	4	3	19.55	7220.14	499.0	0.0	20.25
10	4	4	82.69	7211.46	789.8	0.0	1142.17

Table 3
Asymmetric SEC results with initial solution.

Size	Radii	Mode	Final Gap	Exact Time	Heur. Time	% Improved Gap
5	1	1	0.0	0.11	2.29	0.83
5	1	2	0.0	0.06	2.2	0.57
5	1	3	0.0	0.14	3.82	0.37
5	1	4	0.0	0.12	2.8	0.92
5	2	1	0.0	0.09	2.35	2.51
5	2	2	0.0	0.06	2.37	2.25
5	2	3	0.0	0.15	3.51	1.37
5	2	4	0.0	0.09	2.68	2.53
5	3	1	0.0	0.11	2.26	3.79
5	3	2	0.0	0.08	2.34	3.85
5	3	3	0.0	0.16	3.72	2.05
5	3	4	0.0	0.15	2.82	2.29
5	4	1	0.0	0.13	2.31	4.42
5	4	2	0.0	0.26	2.18	5.18
5	4	3	0.0	0.16	3.48	3.69
5	4	4	0.0	0.14	2.89	8.05
10	1	1	0.0	0.95	4.31	3.25
10	1	2	0.0	0.47	4.37	2.56
10	1	3	0.0	1.72	7.72	1.28
10	1	4	0.0	4.81	5.6	1.53
10	2	1	0.0	21.74	4.73	6.48
10	2	2	0.0	8.45	4.36	6.37
10	2	3	3.23	2949.61	9.25	2.21
10	2	4	1.38	2188.88	6.14	3.22
10	3	1	0.0	522.23	5.08	10.0
10	3	2	0.0	84.64	4.9	9.26
10	3	3	3.47	4324.06	10.54	1.69
10	3	4	0.0	404.51	5.29	7.48
10	4	1	5.32	2484.47	5.17	7.76
10	4	2	0.0	539.03	4.85	9.21
10	4	3	3.57	3656.07	10.11	5.47
10	4	4	16.69	6548.93	6.18	7.86
15	1	1	0.0	14.12	5.63	2.74
15	1	2	0.0	4.63	5.58	4.59
15	1	3	12.12	7200.55	12.9	0.6
15	1	4	0.46	1597.94	7.44	4.2
15	2	1	29.42	7200.43	5.7	11.07
15	2	2	17.89	6178.28	5.6	8.74
15	2	3	13.91	7200.95	12.55	0.1
15	2	4	23.44	7200.6	7.25	3.11
15	3	1	70.59	7200.52	5.77	12.59
15	3	2	35.67	7200.65	5.78	12.85
15	3	3	9.7	5828.43	12.44	2.16
15	3	4	45.94	7200.79	9.95	0.49
15	4	1	100.0	7200.38	99.95	81.0
15	4	2	43.12	7200.7	5.67	7.02
15	4	3	7.6	5789.95	12.67	3.58
15	4	4	41.1	7200.6	8.48	6.57
20	1	1	2.58	3189.58	6.9	2.72
20	1	2	1.78	2894.34	6.47	4.59
20	1	3	10.17	5797.93	13.57	1.78
20	1	4	11.01	6434.25	11.34	1.47
20	2	1	63.7	7200.95	6.41	10.96
20	2	2	39.3	7201.34	6.77	10.83
20	2	3	11.82	6050.34	14.24	4.17
20	2	4	37.99	7200.84	10.26	2.74
20	3	1	95.47	7200.71	6.7	15.29
20	3	2	55.89	7201.07	6.62	16.12
20	3	3	17.75	7201.0	14.43	2.0
20	3	4	45.88	7200.79	11.44	11.6
20	4	1	100.0	7200.55	5.11	11.71
20	4	2	60.12	7201.0	6.43	1.85
20	4	3	10.06	7201.0	14.34	4.05
20	4	4	23.76	7200.58	7.05	4.16

and we report the average results. We have considered three different types of neighborhoods to be visited:

- Circles of radii r.
- Regular polygons of radii r with a random number of vertices in the interval $[3, 10]$.
- Polygonal chains parameterized by its breakpoints that are at a distance of in r from one another and some random percentage $\alpha \in [0, 1]$ of their length to be visited.

In addition, the centers or breakpoints of these elements have been generated uniformly in the square [0, 100]. On the one hand,

Table 4
Symmetric SEC results with initial solution.

Size	Radii	Mode	Final Gap	Exact Time	Heur. Time	% Improved Gap
5	1	1	0.0	0.08	2.38	0.0
5	1	2	0.0	0.04	2.29	0.0
5	1	3	0.0	0.08	3.74	0.51
5	1	4	0.0	0.05	2.88	0.09
5	2	1	0.0	0.06	2.35	0.01
5	2	2	0.0	0.06	2.38	0.0
5	2	3	0.0	0.08	3.59	1.5
5	2	4	0.0	0.06	2.72	1.26
5	3	1	0.0	0.05	2.28	0.01
5	3	2	0.0	0.05	2.36	2.47
5	3	3	0.0	0.09	3.67	0.81
5	3	4	0.0	0.07	2.83	3.36
5	4	1	0.0	0.06	2.32	0.01
5	4	2	0.0	0.05	2.31	0.13
5	4	3	0.0	0.07	3.62	5.95
5	4	4	0.0	0.08	2.95	1.9
10	1	1	0.0	0.29	4.52	0.01
10	1	2	0.0	0.14	4.66	0.01
10	1	3	0.0	0.2	7.8	0.0
10	1	4	0.0	0.23	5.51	0.04
10	2	1	0.0	3.72	5.41	0.05
10	2	2	0.0	1.88	5.69	2.28
10	2	3	0.0	1.34	9.34	0.0
10	2	4	0.0	1.36	9.68	1.17
10	3	1	0.0	45.54	5.27	0.15
10	3	2	0.0	9.37	4.9	0.53
10	3	3	0.0	490.19	10.53	0.67
10	3	4	0.0	7.78	5.33	4.51
10	4	1	0.0	520.74	5.22	0.26
10	4	2	0.0	36.72	4.99	8.1
10	4	3	0.54	1474.78	11.41	0.0
10	4	4	0.0	1461.93	6.84	2.91
15	1	1	0.0	2.1	6.83	0.0
15	1	2	0.0	0.86	6.46	1.11
15	1	3	3.14	2881.8	13.26	0.0
15	1	4	0.0	1.17	7.92	0.0
15	2	1	12.93	5840.0	7.19	0.0
15	2	2	8.9	4560.43	7.0	0.75
15	2	3	11.18	5761.15	15.14	0.11
15	2	4	8.3	4642.84	8.36	0.0
15	3	1	64.34	7200.42	7.39	0.0
15	3	2	28.89	7200.44	7.45	0.0
15	3	3	5.59	5765.79	16.7	0.42
15	3	4	24.3	7200.44	10.87	2.94
15	4	1	99.69	7200.18	99.96	92.84
15	4	2	35.34	7200.52	7.41	3.21
15	4	3	12.84	7200.49	248.17	0.96
15	4	4	35.59	7200.53	10.82	0.8
20	1	1	0.0	175.65	11.47	0.81
20	1	2	0.95	1632.15	10.98	0.03
20	1	3	0.0	466.74	18.07	1.28
20	1	4	8.59	2887.48	16.38	1.36
20	2	1	43.77	7200.49	11.95	0.0
20	2	2	26.72	7200.65	11.51	0.0
20	2	3	4.65	4354.1	27.26	0.0
20	2	4	26.11	7200.5	16.42	0.37
20	3	1	81.51	7200.5	12.52	0.0
20	3	2	47.27	7200.95	12.13	0.0
20	3	3	16.84	7200.81	37.43	0.26
20	3	4	44.18	7200.59	19.15	0.0
20	4	1	100.0	7200.61	4.72	12.06
20	4	2	55.27	7200.73	12.43	0.0
20	4	3	15.84	7200.84	3191.77	0.28
20	4	4	40.08	7200.79	20.47	0.0

we have studied four different scenarios to generate the radii to define the elements:

- **Small size Neighborhoods** ($r = 1$): Radii randomly generated in $[0, 5]$.
- **Small-Medium Neighborhoods** ($r = 2$): Radii randomly generated in $[5, 10]$.
- **Medium-Large size Neighborhoods** ($r = 3$): Radii randomly generated in $[10, 15]$.
- **Large size Neighborhoods** ($r = 4$): Radii randomly generated in $[15, 20]$.

Finally, we have also considered four modes depending on the nature of the neighborhoods:

- Mode 1: All neighborhoods are circles.
- Mode 2: All neighborhoods are regular polygons.
- Mode 3: All neighborhoods are polygonal chains.

Table 5
MTZ results with initial solution.

Size	Radii	Mode	Final Gap	Exact Time	Heur. Time	% Improved Gap
5	1	1	0.0	0.61	1.47	0.02
5	1	2	0.0	0.12	1.3	0.0
5	1	3	0.0	0.21	2.01	0.08
5	1	4	0.0	0.25	1.64	0.06
5	2	1	0.0	0.27	1.33	2.38
5	2	2	0.0	0.13	1.25	0.01
5	2	3	0.0	0.17	1.9	0.64
5	2	4	0.0	0.17	1.49	2.22
5	3	1	0.0	0.44	1.28	1.01
5	3	2	0.0	0.16	1.28	4.02
5	3	3	0.0	0.17	1.95	0.73
5	3	4	0.0	0.3	1.68	0.44
5	4	1	0.0	0.34	1.28	15.36
5	4	2	0.0	0.14	1.26	8.52
5	4	3	0.0	0.16	1.98	1.55
5	4	4	0.0	0.28	1.7	4.39
10	1	1	0.0	1.93	2.63	0.09
10	1	2	0.0	0.75	2.3	0.01
10	1	3	0.0	0.72	4.49	0.3
10	1	4	0.0	1.52	5.11	0.05
10	2	1	0.0	38.83	2.33	0.01
10	2	2	0.0	14.14	2.11	2.29
10	2	3	0.0	2.23	15.6	0.97
10	2	4	0.0	2.52	3.28	0.77
10	3	1	0.0	487.94	2.44	0.16
10	3	2	0.0	35.81	2.22	1.8
10	3	3	0.0	13.28	14.76	2.94
10	3	4	0.0	133.81	3.1	1.16
10	4	1	19.28	3513.38	2.39	0.47
10	4	2	0.0	238.98	2.28	13.46
10	4	3	0.0	20.25	21.99	5.45
10	4	4	0.0	1142.17	3.57	5.52
15	1	1	0.0	18.58	5.98	0.15
15	1	2	0.0	3.38	2.98	0.35
15	1	3	0.0	314.57	9.09	0.2
15	1	4	0.0	10.94	8.58	2.26
15	2	1	6.69	3135.29	3.7	0.67
15	2	2	0.0	2460.78	3.8	4.81
15	2	3	0.0	14.58	87.78	3.68
15	2	4	0.0	1052.16	8.27	2.28
15	3	1	46.33	5760.56	4.12	4.04
15	3	2	20.79	5760.84	4.51	3.57
15	3	3	0.0	322.77	420.37	4.26
15	3	4	14.07	5865.82	7.74	2.37
15	4	1	100.0	7200.47	5.23	3.13
15	4	2	20.2	4421.25	4.35	9.72
15	4	3	0.19	2195.2	237.9	6.28
15	4	4	21.6	7200.42	6.55	11.68
20	1	1	0.0	743.32	13.95	2.78
20	1	2	0.0	110.91	62.75	9.0
20	1	3	1.6	2896.62	17.67	0.13
20	1	4	1.16	3112.33	20.05	1.19
20	2	1	37.26	7200.45	90.1	9.42
20	2	2	19.43	7200.68	5.23	4.37
20	2	3	0.0	1051.22	254.06	5.93
20	2	4	17.15	7200.48	19.33	2.06
20	3	1	78.16	7200.35	6.05	4.07
20	3	2	34.44	5763.72	5.63	6.17
20	3	3	0.73	4530.27	299.19	5.04
20	3	4	30.71	7200.52	22.32	7.34
20	4	1	100.0	7200.63	8.71	6.0
20	4	2	41.32	7200.67	35.68	25.64
20	4	3	1.83	7200.52	307.92	7.45
20	4	4	29.84	7200.52	33.81	9.7

- Mode 4: Mixture of the three previously considered neighborhoods.

The reader should observe that all our instances are symmetric since they are embedded in $\mathbb{R}^2$ and distances are measured with Euclidean norm. All the formulations were coded in Python 3.7, and solved using Gurobi Optimization (2019) in a Intel(R) Xeon(R) E-2146G CPU @ 3.50 GHz and 64GB of RAM. A time limit of 2 hours was set in all the experiments. The interested reader can download some examples of the XPPN formulations in.lp format for several instances in the GitHub link cited in Puerto & Valverde (2021).

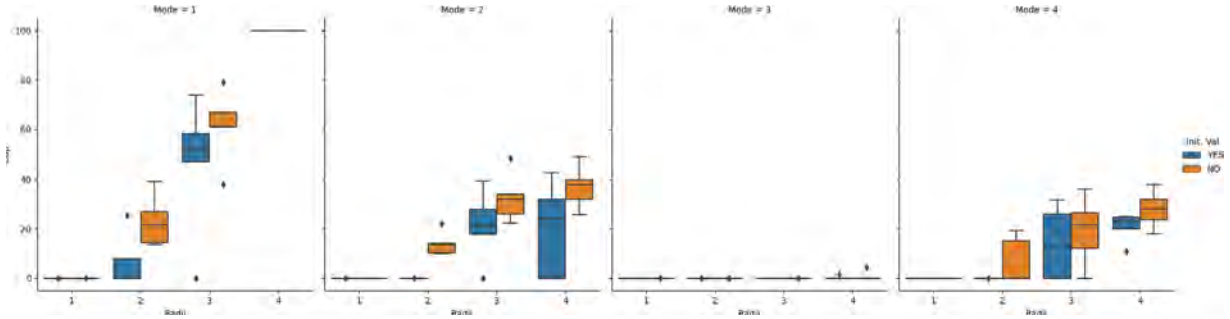


Fig. 10. Comparison of the final gap between MTZ formulation with and without initial solution after 7200 seconds for instances with 15 neighborhoods.

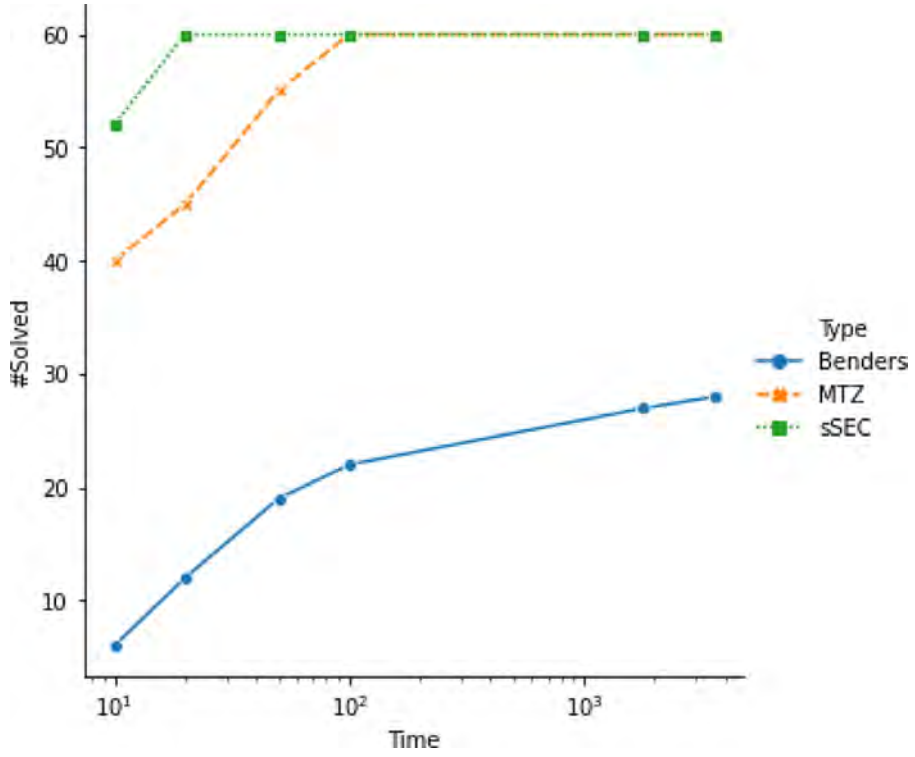


Fig. 11. Performance profile: Time vs #Solved.

7.2. Comparing time and non-time dependent formulations

We analyzed in a preliminary study the time dependent formulation in Section 3.1. For this formulation we have solved a number of instances of the easiest configuration corresponding to neighborhoods given by circles of small radius (Mode 1 and $r = 1$). For a size of $n = 10$ neighborhoods, these instances are always solved to optimality on average in 1800 seconds. However, already for a size of $n = 12$ neighborhoods we could not solve to optimality any of the considered instances within 7200 seconds. Moreover, for 20% of the instances the solver could not even find a feasible solution, and for the rest the average gap was above 80%. For this reason, in the rest of the section, we have restricted ourselves to the comparisons of the non-time dependent formulations presented in Section 3.2 where larger instances can be solved.

7.3. Assessing the difficulty of the problems depending on the α parameter

In order to assess the difficulty of the problem as a function of α we report in this section the following experiment. A batch of 5 instances involving only polygonal chains have been created and solved by fixing the location of the polygonal chains and varying only the α parameter in $\{0, 0.1, 0.2, \ldots, 0.9, 1\}$. To simplify the analysis, we have set the same α for all the polygonals in each instance, although the model has the flexibility to assign a different α for each one. In Fig. 9 we report a boxplot of the execution time that the solver needs to compute the optimal solution of the five instances for each α. As the reader can see, the difficulty of the problem is not monotone on α. It increases monotonically from zero (the simplest) to some maximum difficulty around 0.7 or 0.8 and then, it decreases until $\alpha = 1$ which is, approximately, of the same difficulty as $\alpha = 0.5$. These results make sense according to the complexity of choosing the entering and exiting points of the polygonals: for $\alpha = 0$ these points reduce to only one, the closest point; whereas for $\alpha = 1$ these points are also fixed since they must be the initial and final points. These facts simplifies the problem as explained above.

7.4. Initializing the solver with a heuristic solution

Our next preliminary test was devoted to decide whether initializing the proposed formulations with an initial solution helps in solving the problem or not. In this regard, we have performed

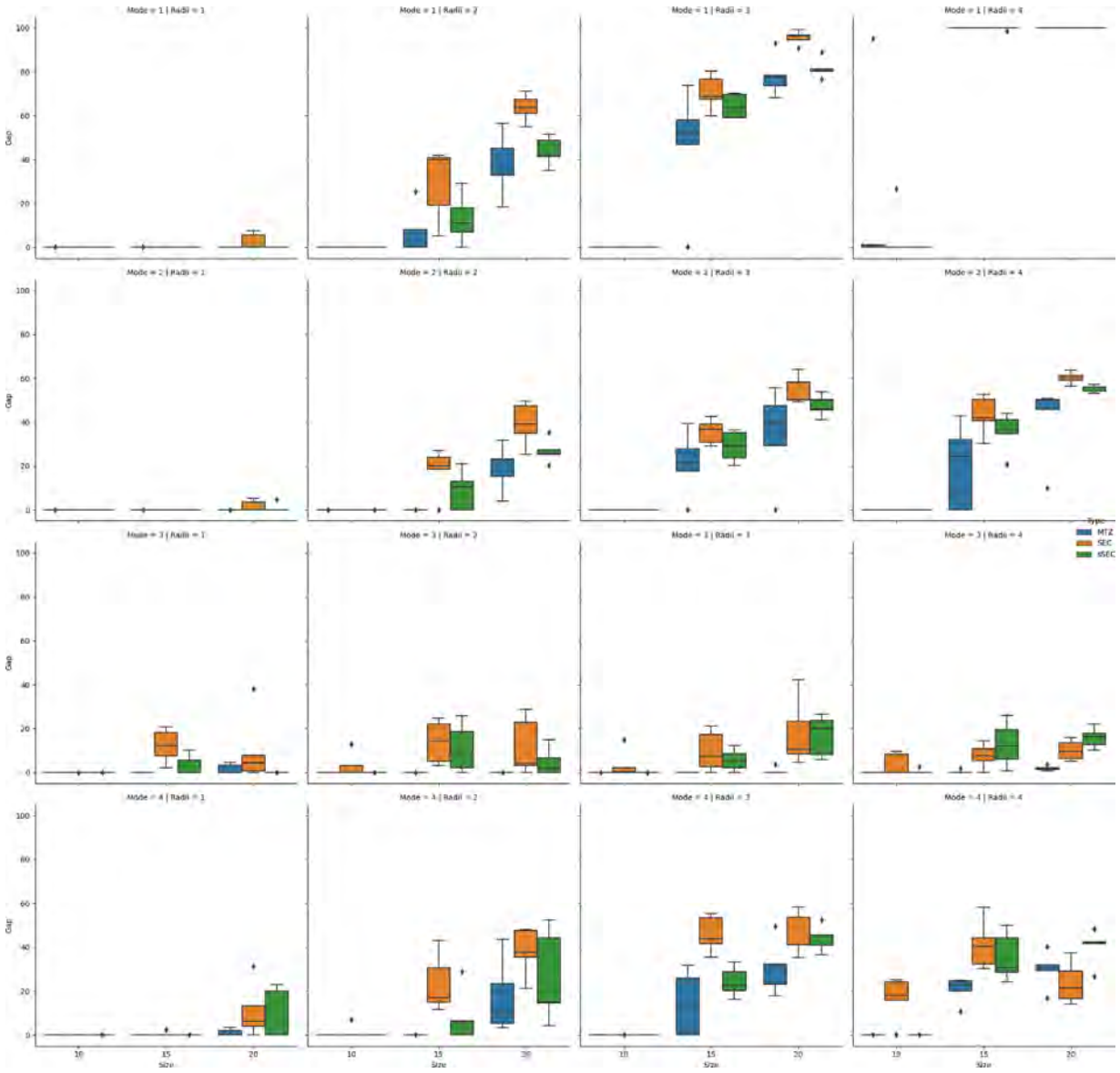


Fig. 12. Final gap after 7200 seconds.

a comparison between formulation MTZ with and without the initial solution provided by the VNS heuristic. The results are summarized in Table 1. This table reports average results for instances of sizes 5,10 and 15 neighborhoods with all combinations of radii and modes. It compares the final gap and running times for the formulation with and without initial solution (**Final Gap (Init), Opt. Time (Init)**) (resp. **Final Gap (NoInit), Opt. Time (NoInit)**). The results are also depicted in the boxplox diagrams in Fig. 10. Both, table and figure, clearly show that loading an initial solution helps in reducing the gap and the cpu time: all the instances up to 10 neighborhoods are solved to optimality with and without initial solution but for 15 neighborhoods the final gap in the first case is always better than in the latter (blue boxes are always below orange ones). Based on this results, in the following, we have always solved the instances loading an initial solution.

7.5. Comparing benders cuts with the MTZ formulation

Here, the decomposition algorithm described in Algorithm 2 is compared with the MTZ formulation without initialization. The computational results obtained by our implementation are included in Table 2. This table compares the results of the Final Gap, cpu time and number of cuts added applying the decomposition algorithm versus those obtained with formulation MTZ. From these results we conclude that the decomposition algorithm performs worse than formulation MTZ even for small size problems. The Benders optimality cuts involve big-Ms, which in turns implies that a lot of cuts are needed (if not all) to certify optimality. The big-M constraints comes from the linearization of bilinear terms which do to allow to apply the Benders approach because the lack of convexity. Thus, this may be one of the reasons why its performance is worse than MTZ. To reinforce our observation, we have also included a performance profile of number of solved instances versus time for formulations sSEC, MTZ and the decomposition algorithm (see Fig. 11). The reader can observe that the number of solved instances within the time limit is approximately one half comparing Benders decomposition with MTZ and sSEC. These results lead us to not include this algorithm in the final computational experience for larger problem sizes presented in the last subsection.

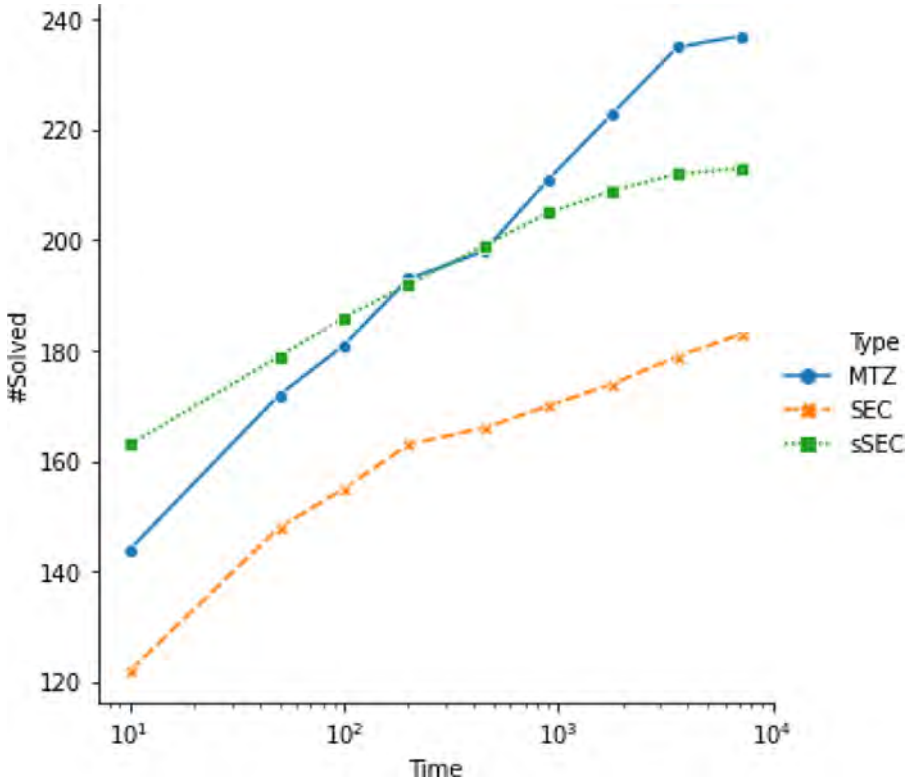


Fig. 13. Performance profile: Time vs #Solved.

7.6. Comparing MTZ, SEC and sSEC with initialization from a heuristic solution

The remaining information of our computational experiments can be found in Tables 3–5. The first one reports our results for formulation SEC, the second one for sSEC (symmetric version of SEC) and the third one for MTZ. Information in all the three tables is organized in the same way. Each row shows averages of five instances for different combinations of factors (**Size**, **Radii** and **Mode**) each one with four different levels. Our tables have 9 columns. The first three (Size, Radii and Mode) describe the parameters of the problem. Then, we report the final gap (**% Final Gap**), time required by the exact method (**Exact Time**), time required by the heuristic (**Heur. Time**) and the % improvement of the gap with respect to the initial solution (**% Improved Gap**). Overall, we have solved 320 instances.

To have a clearer view of the results we also present some comparative boxplots obtained from the tables above. First of all, we report the final gap after two hours of running time. We have gathered all the information in Fig. 12. It is organized in four rows corresponding with the different modes: row i shows results for Mode i, $i = 1, \ldots, 4$. Within each row, there are four columns one per radius size. Then, each graph within this 4×4 grid contains comparative diagrams for the four different problem sizes considered, namely $|V| = 5, 10, 15, 20$ neighborhoods. Finally, for each problem size we compare the results obtained for our three different formulations Miller-Tucker-Zemlin (MTZ), Subtour elimination (SEC) and the symmetric version of SEC (sSEC). For instance, looking at the second row, third column (Mode 2, Radius 3) one can see that for $|V| = 5$ and 10, which correspond to the first two boxes the gap of the three formulations is zero in all the instances (actually the boxes are collapsed to lines). However, for $|V| = 15$ and 20 MTZ seems to outperforms SEC and sSEC, and moreover, sSEC is also better than SEC since the green boxes lie below the orange ones. As a general comment, one can observe that for all combinations of factors MTZ (the blue boxes) outperforms SEC (orange) and sSEC (green) and also sSEC reports smaller gaps than SEC, with the only exception of Mode 3 where SEC seems to work better than sSEC.

Finally, we have included in Fig. 13 a performance profile graph of number of instances solved versus time. This figure shows that SEC formulation is the weakest since it solves less number of instances in the same cpu time. The comparison between MTZ and sSEC is not that clear although in the long run MTZ seems to outperforms sSEC since the former solves more instances than the latter.

We also compare next, the behaviour of SEC and sSEC in number of cuts required by these two formulations to solve the corresponding problems. As before, we have organized the information in a 4×4 grid of boxplox graphs. The reader can easily observe that sSEC always requires less number of cuts (blue boxes corresponding to SEC are always above orange ones corresponding to sSEC) showing that this formulation is more accurate than SEC: it reports smaller gaps (see Fig. 14) and needs less number of cuts.

8. Concluding remarks

This paper has analyzed a novel version of the crossing postman problem with neighborhoods. We have shown that the problem can be cast within the framework of the family of mixed integer second order cone programming and several exact formulations are presented and computationally tested on an extensive testbed of instances. Additionally, we have presented a heuristic algorithm providing good quality solutions. It can be considered for large scale problems and also as a procedure to obtain initial solutions to initialize exact solvers handling our formulations. Computational results show that the problem is very hard and already for problems with 20 neighborhoods exact approaches fail to find an optimal solution within two hours of cpu time.

This research opens up several research lines and extensions of the basic problem that can be included in the model. Among them

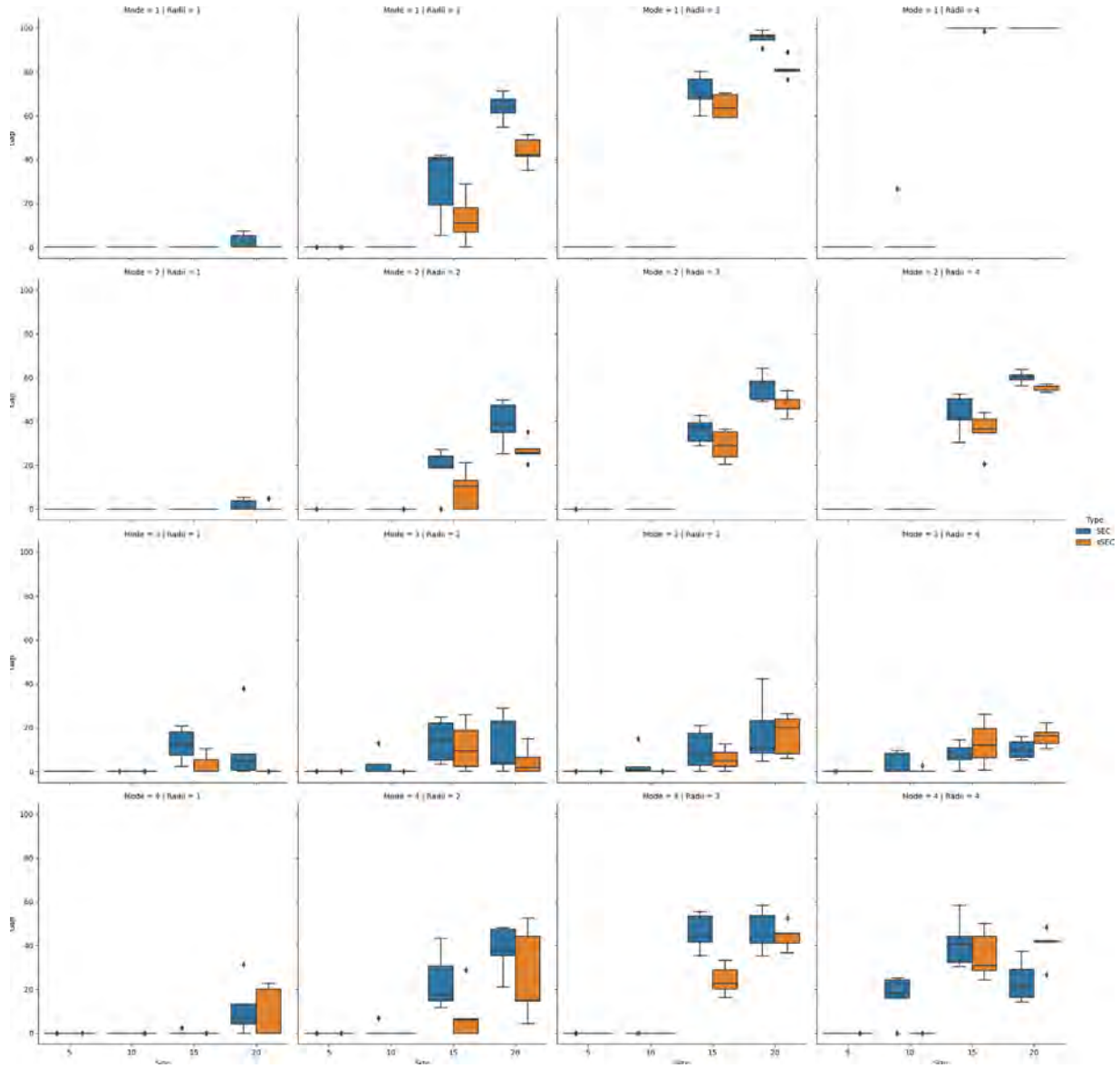


Fig. 14. Number of SEC added in the execution time.

we mention finding better formulations or decomposition schemes that help in solving exactly larger instance sizes; and alternative heuristic algorithms that allow tackling large scale problems. Other extensions of the proposed models considered in this paper are the consideration of barriers that represent some buildings that the drone tour can not cross, conditions that control the displacement on the border of nonlinear neighborhoods like circles or problems that take into account the limited autonomy of drones requiring that the drone comes back to a depot to be recharged before to complete the route. Some of these topics will be the subject of a follow up paper.

Acknowledgments

This research has been partially supported by Spanish Ministry of Education and Science/FEDER grant number MTM2016-74983-C02-(01–02), and projects FEDER-US-1256951, Junta de Andalucía P18-FR-1422, CEI-3-FQM331 and NetmeetData: Ayudas Fundación BBVA a equipos de investigación científica 2019.

References

Amorosi, L., Chiaraviglio, L., D'Andreagiovanni, F., & Blefari-Melazzi, N. (2018). Energy-efficient mission planning of UAVs for 5G coverage in rural zones. In *Proceedings of the IEEE international conference on environmental engineering (ee)* (pp. 1–9). https://doi.org/10.1109/EE1.2018.8385250.

Arkin, E. M., & Hassin, R. (1994). Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics, 55*(3), 197–218. https://doi.org/10.1016/0166-218x(94)90008-6.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik, 4*, 238–252. https://doi.org/10.1007/BF01386316.

Blanco, V. (2019). Ordered p-median problems with neighbourhoods. *Computational Optimization and Applications, 73*, 603–645. https://doi.org/10.1007/s10589-019-00077-x.

Blanco, V., Fernández, E., & Puerto, J. (2017). Minimum spanning trees with neighborhoods: Mathematical programming formulations and solution methods. *European Journal of Operational Research, 262*(3), 863–878. https://doi.org/10.1016/j.ejor.2017.04.023.

Blanco, V., & Puerto, J. (2021). On hub location problems in geographically flexible networks. *International Transactions in Operational Research, n/a*(n/a). https://doi.org/10.1111/itor.12993.

(2015). In Á. Corberán, & G. Laporte (Eds.), *Arc routing: Problems, methods, and applications*. Philadelphia (USA): Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611973679.

Díaz-Báñez, J. M., Mesa, J. A., & Schöbel, A. (2004). Continuous location of dimensional structures. *European Journal of Operational Research, 152*(1), 22–44. https://doi.org/10.1016/S0377-2217(02)00647-1.

Disser, Y., Mihalák, M., Montanar, S., & Widmayer, P. (2014). *Rectilinear shortest path and rectilinear minimum spanning tree with neighborhoods: 8596 LNCS* (pp. 208–220). Springer Verlag. https://doi.org/10.1007/978-3-319-09174-7_18.

Edmonds, J. (2003). *Combinatorial optimization eureka, you shrink!* (pp. 11–26)). Springer.

Garfinkel, R., & Webb, J. (1999). On crossings, the crossing postman problem, and the rural postman problem. *Networks: An International Journal, 34*(3), 173–180. 10.1002/(SICI)1097-0037(199910)34:3%3C173::AID-NET1%3E3.0.CO;2-W

Gentilini, I., Margot, F., & Shimada, K. (2013). The travelling salesman problem with neighbourhoods: Minlp solution. *Optimization Methods and Software, 28*(2), 364–378. https://doi.org/10.1080/10556788.2011.648932.

Geoffrion, A. M. (1972). Generalized Benders decomposition. *Journal of Optimization Theory and Applications, 10*(4), 237–260. https://doi.org/10.1007/BF00934810.

Gurobi Optimization, L. (2019). Gurobi optimizer reference manual, version 8.1.0.

Knight, R. (2016). Drones deliver healthcare. http://insideunmannedsystems.com/drones-deliver-healthcare/.

Lavars, N. (2015). Amazon to begin testing new delivery drones in the US. *New Atlas, 13*.

Lobo, M. S., Vandenberghe, L., Boyd, S., & Lebret, H. (1998). Applications of second-order cone programming. *Linear Algebra and its Applications, 284*(1), 193–228. https://doi.org/10.1016/S0024-3795(98)10032-0. International Linear Algebra Society (ILAS) Symposium on Fast Algorithms for Control, Signals and Image Processing

Mallozzi, L., Puerto, J., & Rodríguez-Madrena, M. (2019). On location-allocation problems for dimensional facilities. *Journal of Optimization Theory and Applications, 182*(2), 730–767. https://doi.org/10.1007/s10957-018-01470-y.

McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part I convex underestimating problems. *Mathematical Programming, 10*, 147–175. https://doi.org/10.1007/BF01580665.

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM, 7*(4), 326329. https://doi.org/10.1145/321043.321046.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*(11), 1097–1100. https://doi.org/10.1016/s0305-0548(97)00031-2.

Orloff, C. S. (1974). A fundamental problem in vehicle routing. *Networks, 4*(1), 35–64. https://doi.org/10.1002/net.3230040105.

Otto, A., Agatz, N., Campbell, J., Golden, B., & Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks, 72*(4), 411–458. https://doi.org/10.1002/net.21818.

Pereira, V. (2018). Project: Metaheuristic-Local_Search-Variable_Neighborhood_Search. https://github.com/Valdecy/Metaheuristic-Local_Search-Variable_Neighborhood_Search.

Puerto, J., & Valverde, C. (2021). Project: Instances for the crossing postman problem with neighborhoods (XPPN). https://github.com/z72vamac/examples_xppn.

Sawik, T. (2016). A note on the miller-Tucker-Zemlin model for the asymmetric traveling salesman problem. *Bulletin of the Polish Academy of Sciences Technical Sciences, 64*(3), 517–520. https://doi.org/10.1515/bpasts-2016-0057.

Schöbel, A. (2015). Location of dimensional facilities in a continuous space. In *Location science* (pp. 135–175). Springer International Publishing. https://doi.org/10.1007/978-3-319-13111-5_7.

Velednitsky, M. (2017). Short combinatorial proof that the DFJ polytope is contained in the MTZ polytope for the asymmetric traveling salesman problem. *Operations Research Letters, 45*(4), 323–324. https://doi.org/10.1016/j.orl.2017.04.010.

Yang, Y., Lin, M., Xu, J., & Xie, Y. (2007). *Minimum spanning tree with neighborhoods: 4508 LNCS* (pp. 306–316). Springer Verlag. https://doi.org/10.1007/978-3-540-72870-2_29.

Yuan, B., & Zhang, T. (2017). Towards solving TSPN with arbitrary neighborhoods: A hybrid solution. In *Acalci* (pp. 204–215). https://doi.org/10.1007/978-3-319-51691-2.

Chapter 7

The Hampered Travelling Salesman Problem with Neighbourhoods

The Hampered Travelling Salesman Problem with Neighbourhoods*

Justo Puerto[a,1], Carlos Valverde[b,1,*]

[a] *Department of Statistics and Operations Research, University of Seville, Seville, 41012, Spain*
[b] *Department of Statistics and Operations Research, University of Seville, Seville, 41012, Spain*

Abstract

This paper deals with two different route design problems in a continuous space with neighbours and barriers: the shortest path and the travelling salesman problems with neighbours and barriers. Each one of these two elements, neighbours and barriers, makes the problems harder than their standard counterparts. Therefore, mixing both together results in a new challenging problem that, as far as we know, has not been addressed before but that has applications for inspection and surveillance activities and the delivery industry assuming uniformly distributed demand in some regions. We provide exact mathematical programming formulations for both problems assuming polygonal barriers and neighbours that are second-order cone (SOC) representable. These hypotheses give rise to mixed integer SOC formulations that we preprocess and strengthen with valid inequalities. The paper also reports computational experiments showing that our exact method can solve instances with 75 neighbourhoods and a range between 125-145 barriers.

Keywords: Routing, Travelling salesman, Networks, Conic programming and interior point methods

1. Introduction

Routing problems are considered classical problems at the core of combinatorial optimisation. Among them, the Travelling Salesman Problem (TSP) is one of its most important representations, and it has been studied extensively in many different forms. The problem is well-known to be NP-hard. Its analysis has led in the last decades to methodological advances and new optimisation techniques that have allowed solving real life instances that few years ago were beyond solvable limits. The TSP has been studied in its geometric and pure combinatorial forms because of its algorithmic and practical implications. One very interesting extension arises when we require the tour to visit a set of regions rather than points. This version of the problem is called the TSP with Neighbours (see Arkin and Hassin (1994)) which is APX-hard to approximate even for regions that are (intersecting) line segments. They can represent regions that the drone must reach and where customers are willing to pick up the orders (they can be seen as uniform probability densities) in the delivery industry. Moreover, they can also be used for modelling some areas that must be inspected by the drone (whenever visiting a point of these areas, it suffices to consider them as inspected). On the other hand, the graphic TSP (see T. Moemke and O. Svensson

*This research has been partially supported by the Agencia Estatal de Investigación (AEI) and the European Regional Development Fund (ERDF): PID2020-114594GB-C21; and Regional Government of Andalusia: project P18-FR-1422.
*Corresponding author
Email addresses: puerto@us.es (Justo Puerto), cvalverde@us.es (Carlos Valverde)
[1] Equally contributing authors

(22)), where distances are measures by geodesics (shortest paths in the respective metric space), has also attracted the attention of many researchers since it models, in a natural way, TSP routes among forbidden barriers, robot motion planning and automatic navigation in computer games.

For the above mentioned reasons, it is very important to analyse the mathematical implications of barriers to the geometry and computation of routes in a continuous space. Beyond the mathematical relevance due to the intrinsic difficulty of the resulting models (non-convexities), the importance of this topic also comes from its many real-life applications and, more specifically, drone delivery with uniformly distributed demand and drone inspection and surveillance in urban areas with buildings or similar barriers. Most of the classical methods in route design are based on an underlying network structure. Our approach is essentially different since, on top of assuming the existence of barriers, we also assume that the locations of targets to be visited are uniformly distributed in neighbourhoods, giving rise to challenging problems in the continuous space. Nevertheless, the resulting problems still retain geometric elements that must be exploited to partially overcome the difficulties of solution approaches and algorithms in the design of routes among neighbourhoods with barriers.

The well-known Travelling Salesman Problem with Neighbourhoods (TSPN) was introduced by Arkin and Hassin (1994). These authors addressed this problem by proposing heuristic procedures that construct tours and prove that the length of these tours is within a constant factor of the length of an optimal tour. In Gentilini et al. (2013), authors formulate this problem as a non-convex mixed integer non-linear program (MINLP) that yields a convex nonlinear program when the discrete variables are fixed. Moreover, Yuan and Zhang (2017) combine strategically metaheuristics and classical TSP solvers to produce high quality solutions for the TSPN with arbitrary neighbourhoods. In Glock and Meyer (2023), the concept of spatial coverage in routing and path planning is unified and defined, and a categorisation scheme of related problems is introduced. Other classical combinatorial problems, such as the Minimum Spanning Tree (MST) or the Crossing Postman Problem (XPP), which have been extended to deal with neighbourhoods, are developed in Blanco et al. (2017) and Puerto and Valverde (2022), respectively.

In Mennell (2009), the Close-Enough Travelling Salesman problem was introduced. This problem is a particular case of the TSPN where the neighbourhoods are represented by circles. In that dissertation, the author also performed extensive computational experiments by solving instances developed by them. The authors of Coutinho et al. (2016) proposed an exact algorithm based on branch-and-bound and second-order cone programming. Finally, a metaheuristic approach was developed that computes the upper and lower bounds of the optimal solution value. This metaheuristic used a discretization scheme and the Carousel Greedy algorithm to obtain high-quality solutions (see Carrabs et al. (2020) for more details).

As far as we are concerned, the use of barriers in location problems has been widely considered (see Klamroth (2002)) but the corresponding barrier routing problem has attracted less attention in the field of Operations Research. In spite of that, one can find connections with some problems in the area of computational geometry, such as the shortest path problem in polygons and the touring polytopes problem (see Mitchell (2017)), in navigation games competition as, for instance, the Physical Travelling

Salesman Problem (see D. Perez et al. (2014)), or in robot motion planning (see Y. K. Hwang and N. Ahuja (1992) and J. . -P. Laumond et al. (1994)). Different complexity results are known, and many effective heuristic algorithms have been developed for specific classes of problems. However, as far as we know, in all cases targets are points and no exact algorithms are provided. Moreover, at times, these problems can appear as subproblems of some other combinatorial problems, so that one would like to embed them into some mathematical programming formulation. Nevertheless, we are not aware of any mathematical programming formulation for these problems that permits it to be considered as a building block of more complex/integrated problems in real life applications, even without requiring target to be neighbourhoods.

Our goal in this paper is to deal with the TSP with Neighbourhoods and barriers that we call the Hampered Traveling Salesman with Neighbourhoods (H-TSPN). As commented above, each one of these two elements (neighbourhoods and barriers) makes the problem difficult. Thus, mixing both together results in a new problem that, as far as we know, has not been addressed before, but that has a lot of applications in the delivery industry and inspection and surveillance activities, as justified previously. Moreover, it also has implications from the methodological point of view because introduces non-fixed elements in network representation and the issue of obtacles (barriers) in the solution space in the same problem.

Our contribution is to provide exact mathematical programming formulations for the problem assuming polygonal barriers and neighbourhoods that are second-order cone (SOC) representable. These hypotheses give rise to mixed integer SOC formulations that we preprocess and strengthen with valid inequalities. In our way to the formulations we deal with the associated Hampered Shortest Path Problem with Neighbourhoods (H-SPPN) which is used as a building block for the more difficult H-TSPN. Although we prove that the H-SPPN is polynomially solvable, we also give a formulation for this problem that highlights the constraints necessary for H-TSPN. The respective H-TSPN is NP-hard. We give exact formulations using a geodesic shortest-path representation that allows us to solve medium-sized instances of this class of problems. Our computational tests show the difficulty of handling barriers and neighbourhoods together, but also that our formulation is useful for problems with 75 neighbourhoods and a range between 125-145 barriers.

The paper is organised into 6 sections. The first section is the introduction. In the second section, the problems to be dealt with are described, and the notation followed in the rest of the paper is set. Section 3 develops formulations for the problems defined in the manuscript. These initial formulations are later reformulated using finite dominating sets described in the paper. Section 4 strengthens these formulations by preprocessing variables, developing families of valid inequalities to be added to the formulations and proposing some variable-fixing strategies. A computational experience is reported in Section 5. The paper ends with a section devoted to conclusions and extensions, where some interesting further research connected with the problems addressed in this paper is discussed.

2. Preliminaries and Problems' Description

This section is devoted to setting the hypotheses that describe the framework where we analyse the considered problems. We will deal with two problems in this paper: the Hampered Shortest Path Problem with Neighbourhoods H-SPPN and the Hampered Travelling Salesman Problem with Neighbourhoods H-TSPN. Since we have in mind their applications to the drone delivery problem with uniformly distributed demand in regions and inspection problems, at times, we will refer to the moving object as *drone*.

2.1. Assumptions and notation

In both problems considered, we denote by $\mathcal{B}$ the set of line segments that model the barriers and adopt the assumptions listed below.

A1 The line segments of $\mathcal{B}$ are located in a general position, i.e., the endpoints of these segments are not aligned. Although it is possible to model the aligned case, one can always slightly modify one of the endpoints so that the segments are not aligned.

A2 The line segments of $\mathcal{B}$ are open sets, that is, it is possible that the drone visits the endpoints of the segments, but entering into its interior is not allowed. Observe that without loss of generality, we can always slightly enlarge these segments to make them open.

A3 If there are two aligned overlapping barriers, we assume that there is only one barrier containing both of them.

In this work, we also analyse a special case of the H-TSPN that assumes, besides the previous hypotheses, that:

A4 There is no rectilinear path connecting two neighbourhoods without crossing an obstacle.

This problem is called the Hampered Traveling Salesman Problem with Hidden Neighbourhoods H-TSPHN. The consideration of this more constrained subproblem appears as a natural extension of the H-SPPN. Indeed, in H-SPPN, the shortest path that joins two neighbourhoods is sought by assuming **A4**. Otherwise, it would be trivially solved by finding the standard shortest path between both neighbourhoods since no obstacles would block its use.

Figures 1, 2 and 3 show an example of each one of the three problems that are being considered, respectively. Figure 1 shows an instance of the H-SPPN, where the blue neighbourhood represents the source, the green one represents the target, and the red line segments show the barriers that the drone cannot cross. In Figure 2, an instance of the H-TSPHN is shown, where the neighbourhoods are balls and the barriers are, again, the red line segments. Finally, Figure 3 illustrates an example of the H-TSPN, where the orange and blue balls can be joined by a rectilinear path.

To simplify the presentation, we introduce some general notation used throughout the paper.

Figure 1: H-SPPN instance

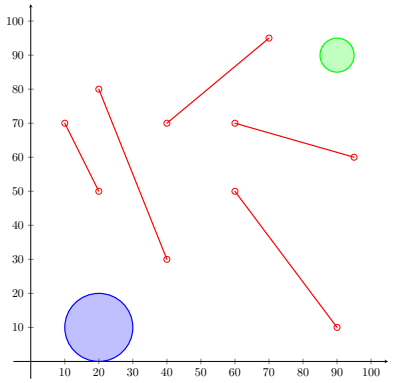


Figure 2: H-TSPHN instance

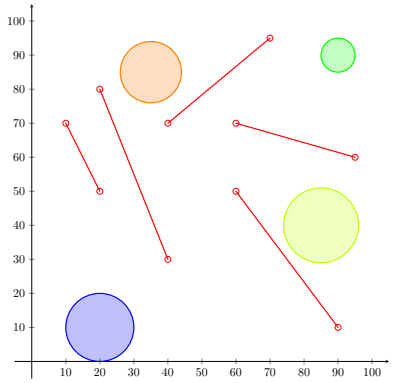
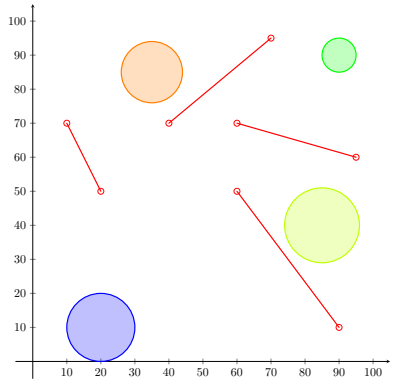


Figure 3: H-TSPN instance



Notation

The following terminology clarifies the notation applied throughout the paper:

- P and Q are referred to as generic points that, at the same time, are identified with their coordinates $P = (P_x, P_y)$ and $Q = (Q_x, Q_y)$, respectively.
- Given two points P^1 and P^2, the line segment that joins P^1 and P^2 is denoted by $\overline{P^1P^2}$. It can be parameterized as follows:

$$\overline{P^1P^2} = \{P \in \mathbb{R}^2 : P = \lambda P^1 + (1-\lambda)P^2,\ \lambda \in [0,1]\}.$$

- Given two points P^1 and P^2, the edge whose vertices are P^1 and P^2 is denoted by (P^1, P^2).
- Given two points P^1 and P^2, the vector pointing from P^1 to P^2 is denoted by $\overrightarrow{P^1P^2}$. It is computed as $\overrightarrow{P^1P^2} = P^2 - P^1$.
- Given three points P^1, P^2 and P^3, $\det(P^1|P^2P^3)$ denotes the following determinant:

$$\det(P^1|P^2P^3) = \det\left(\overrightarrow{P^1P^2} \;\middle|\; \overrightarrow{P^1P^3} \right) := \det\begin{pmatrix} P^2_x - P^1_x & P^3_x - P^1_x \\ P^2_y - P^1_y & P^3_y - P^1_y \end{pmatrix}.$$

  The sign of $\det(P^1|P^2P^3)$ gives the orientation of the point P^1 with respect to the line segment $\overline{P^2P^3}$. Note that $\det(P^1|P^2P^3) \neq 0$ by **A1**.

Once defined the problems treated in this work, the following subsections describe the construction of the visibility graphs that are used to develop some mathematical programming formulations that solve the problem exactly. These objects represent the set of possible rectilinear paths the drone can follow without crossing any barrier (see, e.g., O'Rourke (2017), for more details). For these problems, the visibility graphs are not fixed. They depend on the points selected in the neighbourhoods, as shown in Figures 4 and 5.

2.2. Description of the Hampered Shortest Path Problem with Neighbourhoods

In H-SPPN, we have a source neighbourhood $N_S \subset \mathbb{R}^2$ and a target neighbourhood $N_T \subset \mathbb{R}^2$, which we assume to be second-order cone-representable sets and a set $\mathcal{B}$ of line segments that play the role of barriers that the drone cannot cross.

The goal of the H-SPPN is to find the best pair of points $(P_S, P_T) \in N_S \times N_T$ in the source and target neighbourhoods that minimise the length of the path that joins both points without crossing any barrier of $\mathcal{B}$ and assuming **A1-A4**. To state the model, we define the following sets:

- $V_S = \{P_S\}$. Set composed by the point selected in the source neighbourhood N_S.
- $V_{\mathcal{B}} = \{P_B^1, P_B^2 : B = \overline{P_B^1 P_B^2} \in \mathcal{B}\}$. Set of endpoints of the barriers in the problem.
- $V_T = \{P_T\}$. Set composed by the point selected in the target neighbourhood N_T.
- $E_S = \{(P_S, P_B^i) : P_B^i \in V_{\mathcal{B}} \text{ and } \overline{P_S P_B^i} \cap B'' = \emptyset, \forall B'' \in \mathcal{B},\ i = 1, 2\}$. Set of edges formed by the lines that join the selected point in the source neighbourhood N_S and each endpoint on the barriers that do not cross any other barrier in $\mathcal{B}$.
- $E_{\mathcal{B}} = \{(P_B^i, P_{B'}^j) : P_B^i, P_{B'}^j \in V_{\mathcal{B}} \text{ and } \overline{P_B^i P_{B'}^j} \cap B'' = \emptyset,\ \forall B'' \in \mathcal{B},\ i, j = 1, 2\}$. Set of edges formed by the line segments that join two endnodes of $V_{\mathcal{B}}$ and do not cross any other barrier in $\mathcal{B}$.
- $E_T = \{(P_B^i, P_T) : P_B^i \in V_{\mathcal{B}} \text{ and } \overline{P_B^i P_T} \cap B'' = \emptyset, \forall B'' \in \mathcal{B},\ i = 1, 2\}$. Set of edges formed by the line segments that join the selected point in the target neighbourhood N_T and every endpoint on the barriers that do not cross any other barrier in $\mathcal{B}$.

The above sets allow us to define the visibility graph $G_{\text{SPP}} = (V_{\text{SPP}}, E_{\text{SPP}})$ induced by barriers and neighbourhoods, where $V_{\text{SPP}} = V_S \cup V_{\mathcal{B}} \cup V_T$ and $E_{\text{SPP}} = E_S \cup E_{\mathcal{B}} \cup E_T$.

Figures 4 and 5 show how the visibility graph G_{SPP} is generated for an instance of the H-SPPN. The blue dashed line segments represent the edges of E_S, the green dashed lines, the edges of E_T and the red dashed lines represent the edges of $E_{\mathcal{B}}$.

Figure 4: Generation of the visibility graph G_{SPP}. Case 1

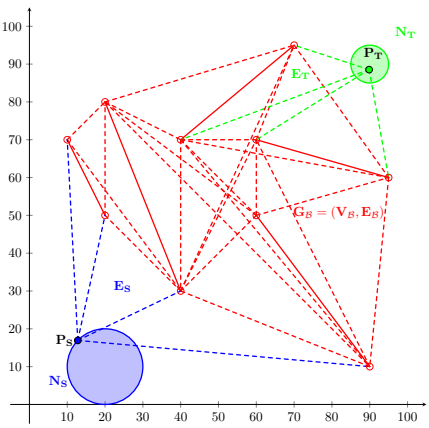
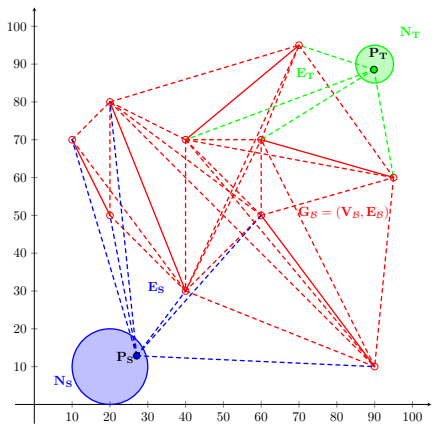


Figure 5: Generation of the visibility graph G_{SPP}. Case 2



A special case that can be highlighted occurs when the neighbourhoods, N_S and N_T, are points. In that case, the induced graph is completely fixed and it is only necessary to find which edges are included

by keeping in mind that the graph must be planar, i.e., without crossings. This idea is later exploited in Subsection 3.3.1.

2.3. Description of the Hampered Travelling Salesman Problem with Hidden Neighbourhoods

The H-TSPHN is an extension of the H-SPPN where the neighbourhood set $\mathcal{N}$ is considered to play the role of source and target in the H-SPPN and, moreover, a set of given targets must be visited. The aim of the H-TSPHN is to find the shortest tour that visits each neighbourhood $N \in \mathcal{N}$ exactly once without crossing any barrier $B \in \mathcal{B}$ and assuming again **A1**-**A4**.

To present our formulation for the H-TSPHN, the graph induced by the endpoints of the barriers and the neighbourhoods is different from the previous one for the H-SPPN. For its description, we introduce the following sets:

- $V_{\mathcal{N}} = \{P_N : N \in \mathcal{N}\}$. Set of points selected in the neighbourhoods $\mathcal{N}$ to be visited.
- $V_{\mathcal{B}} = \{P_B^1, P_B^2 : B = \overline{P_B^1 P_B^2} \in \mathcal{B}\}$. Set of endpoints of the barriers of the problem.
- $E_{\mathcal{N}} = \{(P_N, P_B^i) : P_N \in V_{\mathcal{N}}, P_B^i \in V_{\mathcal{B}} \text{ and } \overline{P_N P_B^i} \cap B'' = \emptyset, \forall B'' \in \mathcal{B}, i = 1, 2\}$. Set of edges formed by line segments that join each point selected in the neighbourhoods of $\mathcal{N}$ with each endpoint on the barriers and do not cross any barrier in $\mathcal{B}$.
- $E_{\mathcal{B}} = \{(P_B^i, P_{B'}^j) : P_B^i, P_{B'}^j \in V_{\mathcal{B}} \text{ and } \overline{P_B^i P_{B'}^j} \cap B'' = \emptyset,\ \forall B'' \in \mathcal{B},\ i, j = 1, 2\}$. Set of edges formed by line segments that join two vertices of $V_{\mathcal{B}}$ and do not cross any barrier in $\mathcal{B}$.

Following the same idea as before, we consider the visibility graph $G_{\text{TSPH}} = (V_{\text{TSPH}}, E_{\text{TSPH}})$ induced by barriers and neighbourhoods, where $V_{\text{TSPH}} = V_{\mathcal{N}} \cup V_{\mathcal{B}}$ and $E_{\text{TSPH}} = E_{\mathcal{N}} \cup E_{\mathcal{B}}$.

2.4. Description of the Hampered Travelling Salesman Problem with Neighbourhoods

For the general case, barriers are not required to completely separate all neighbourhoods, i.e., when moving from one neighbourhood to another, sometimes it is possible to follow a straight line without crossing any barrier. The main difference lies in the description of the edges of the graph induced by the neighbourhoods and endpoints of the barriers.

By following the same approach that before, the sets that describe the visibility graph in this case are:

- $V_{\mathcal{N}} = \{P_N : N \in \mathcal{N}\}$. Set of points in the neighbourhoods $\mathcal{N}$ that must be visited.
- $V_{\mathcal{B}} = \{P_B^1, P_B^2 : B = \overline{P_B^1 P_B^2} \in \mathcal{B}\}$. Set of endpoints of the barriers of the problem.
- $V_{\text{TSP}} = V_{\mathcal{N}} \cup V_{\mathcal{B}}$.
- $E_{\text{TSP}} = \{(P, P') : P, P' \in V_{TSP} \text{ and } \overline{PP'} \cap B'' = \emptyset, \forall B'' \in \mathcal{B}\}$. Set of edges formed by the line segments that join every pair of points in V_{TSP} that do not cross any barrier.

3. MINLP Formulations

This section proposes a mixed integer nonlinear programming formulation for the problems described in Section 2. First of all, we present the conic programming representation of the neighbourhoods and distance. Then, we set the constraints that check if a segment is included in the set of edges E_X with $X \in \{SPP, TSPH, TSP\}$. Finally, the formulations for the H-SPPN, H-TSPHN and H-TSPN are described.

We would like to remark that having a mathematical programming representation for the H-SPPN is very important, even though we will prove in the following that this problem can be solved with a polynomial-time combinatorial algorithm. Indeed, we will need this type of representation to address some NP-hard problems that require computing shortest paths as building blocks, such as, for example, the location problem with barriers and neighbours and the H-TSPN. Therefore, we will start providing two different mathematical programming formulations for the H-SPPN capable of being embedded in more complex problems.

First, we introduce the decision variables that represent the problem. These are summarised in Table 1.

Table 1: Summary of decision variables used in the mathematical programming model

Binary Decision Variables	
Name	**Description**
$\alpha(P\|QQ')$	1, if the determinant $\det(P\|QQ')$ is positive, 0, otherwise.
$\beta(PP'\|QQ')$	1, if the determinants $\det(P\|QQ')$ and $\det(P'\|QQ')$ have the same sign, 0, otherwise.
$\gamma(PP'\|QQ')$	1, if the determinants $\det(P\|QQ')$ and $\det(P'\|QQ')$ are both positive, 0, otherwise.
$\delta(PP'\|QQ')$	1, if the line segments $\overline{PP'}$ and $\overline{QQ'}$ intersect, 0, otherwise.
$\varepsilon(PP')$	1, if the line segment $\overline{PP'}$ does not cross any barrier, 0, otherwise.
$y(PQ)$	1, if the edge (P, Q) is selected in the solution of the model, 0, otherwise.
Continuous Decision Variables	
Name	**Description**
P_N	Coordinates representing the point selected in the neighbourhood N.
$d(PQ)$	Euclidean distance between the points P and Q.
$g(PQ)$	Amount of commodity passing through the edge (P, Q).

3.1. Conic programming constraints in the models

Let $\|\cdot\|_2$ denote the standard Euclidean norm derived from the dot product in $\mathbb{R}^n$, i.e., $\|u\|_2 = (u^t u)^{1/2}$. The second-order cone (or Lorentz cone) of dimension $k+1$ is defined as:

$$\mathcal{L}^{k+1} = \{(x, y) \in \mathbb{R}^k \times \mathbb{R} : \|x\|_2 \leq y\}.$$

Then, a second-order cone constraint is defined as

$$\|A_i x - b_i\|_2 \leq c_i^t x - d_i \Leftrightarrow (A_i x - b_i, c_i^t x - d_i) \in \mathcal{L}^{k_i+1}, \quad i = 1, \ldots, m,$$

where $A_i \in \mathbb{R}^{k_i \times n}, b_i \in \mathbb{R}^{k_i}, c_i \in \mathbb{R}^n$ and $d_i \in \mathbb{R}$ are the problem parameters.

Second-order cone constraints can be used to represent common convex constraints. If $A_i \equiv 0$ and $b_i \equiv 0$, for some $i \in 1, \ldots, m$, the corresponding second-order cone constraint is reduced to a linear one. On the other hand, if $c_i \equiv 0$ and $d_i \leq 0$, the constraint is reduced to a convex quadratic constraint that can represent ellipsoids or hyperbolic constraints (see Lobo et al. (1998) and Boyd and Vandenberghe (2004) for more information). The advantages of second-order constraints are the fully symmetric duality, heavily utilised by solution algorithms, the existence of polynomial-time interior point methods and its extremely powerful modeling possibilities (Nesterov and Nemirovski, 1994).

In the two problems considered, namely H-SPPN and H-TSPN, there exist two second-order cone constraints that model the distance between pairs of points P and Q, as well as the representation of neighbourhoods where the points can be chosen.

For modelling the distance between pairs of points, we introduce the nonnegative continuous variable $d(PQ)$ that represents the distance between P and Q:

$$\|P - Q\| \leq d(PQ), \quad \forall (P, Q) \in E_X, \tag{d-C}$$

where E_X is the set of edges E_{SPP}, E_{TSPH} or E_{TSP}.

For the representation of neighbourhoods where points can be chosen, since we are assuming that the neighbourhoods are second-order cone (SOC) representable, they can be expressed by means of the constraints:

$$P_N \in N \Longleftrightarrow \|A_N^i P_N + b_N^i\| \leq (c_N^i)^T P_N + d_N^i, \quad i = 1, \ldots, nc_N, \tag{N-C}$$

where A_N^i, b_N^i, c_N^i and d_N^i are the constraints parameters, i and nc_N denotes the number of constraints that appear in the block associated with the neighbourhood N.

These type of elements could be extended further to unions of SOC representable sets by introducing binary variables. Each one determines the set where the point chosen to visit the union of these SOC representable sets is located.

Let $\mathcal{U}_N = \{C_N^1, \ldots, C_N^{m_N}\}$ be the second-order cone representable sets that define the neighbourhood N. Consider the binary variable χ_N^j that is one if $P_N \in C_N^j$, and zero otherwise. Therefore, for each $N \in \mathcal{N}$,

$$P_N \in \mathcal{U}_N \Longleftrightarrow \begin{cases} \|A_N^{ij} P_N + b_N^{ij}\| & \leq & (c_N^{ij})^T P_N + d_N^{ij} + U_N^j (1 - \chi_N^j), \quad i = 1, \ldots, nc_N, \quad j = 1, \ldots, m_N, \\ \sum_{j=1}^{m_N} \chi_N^j & = & 1, \end{cases} \tag{U-C}$$

where U_N^j is a big M constant on the maximal distance between two points in the union of sets. The

reader may observe that (N-C) can be replaced by (U-C) without altering the validity of the formulations. However, for the sake of simplicity, only the SOC-representable sets are considered in the paper.

3.2. Checking whether a segment is an edge of the induced graph

The goal of this subsection is to provide a test to check whether, given two arbitrary vertices $P, Q \in V_X$, the edge $(P, Q) \in E_X$, with $X \in \{SPP, TSPH, TSP\}$, that is, whether the line segment $\overline{PQ}$ does not intersect any barrier of $\mathcal{B}$. The following well-known computational geometry result described, among others, in Boissonnat and Snoeyink (2000), can be used to check if two line segments intersect.

Remark 1. *Let $\overline{PQ}$ and $B = \overline{P_B^1 P_B^2} \in \mathcal{B}$ be two different line segments. If*

$$\operatorname{sign}\left(\det(P|P_B^1 P_B^2)\right) = \operatorname{sign}\left(\det(Q|P_B^1 P_B^2)\right) \quad \text{or} \quad \operatorname{sign}\left(\det(P_B^1|PQ)\right) = \operatorname{sign}\left(\det(P_B^2|PQ)\right),$$

then $\overline{PQ}$ and B do not intersect.

If the sign of $\det(P|P_B^1 P_B^2)$ is the same than $\det(Q|P_B^1 P_B^2)$, then, the orientation of P and Q with respect to the segment $\overline{P_B^1 P_B^2}$ is the same. It means that both points lie in the same half-plane generated by the line that contains this segment.

Let $P, Q \in V_X$, where V_X denotes the set of vertices V_{SPP}, V_{TSPH} or V_{TSP}. It is essential to model the conditions of the Remark 1 by using binary variables that check the sign of determinants, the equality of signs, and the disjunctive condition, since these determinants depend on the location of P and Q.

To model the sign of each determinant in Remark 1, we introduce the binary variable α, which assumes the value one if the determinant is positive and zero, otherwise. Note that the case where the determinants are null does not need to be considered because the segments are located in a general position.

It is possible to represent the sign condition by including the following constraints:

$$\begin{aligned}
\left[1 - \alpha(P|P_B^1 P_B^2)\right] L(P|P_B^1 P_B^2) \le \det(P|P_B^1 P_B^2) \le U(P|P_B^1 P_B^2)\, \alpha(P|P_B^1 P_B^2), \qquad (\alpha\text{-C})\\
\left[1 - \alpha(Q|P_B^1 P_B^2)\right] L(Q|P_B^1 P_B^2) \le \det(Q|P_B^1 P_B^2) \le U(Q|P_B^1 P_B^2)\, \alpha(Q|P_B^1 P_B^2),\\
\left[1 - \alpha(P_B^1|PQ)\right] L(P_B^1|PQ) \le \det(P_B^1|PQ) \le U(P_B^1|PQ)\, \alpha(P_B^1|PQ),\\
\left[1 - \alpha(P_B^2|PQ)\right] L(P_B^2|PQ) \le \det(P_B^2|PQ) \le U(P_B^2|PQ)\, \alpha(P_B^2|PQ),
\end{aligned}$$

where L and U are the lower and upper bounds of the value of the corresponding determinants, respectively. If a determinant is positive, then α must be one to make the second inequality feasible. Analogously, if the determinant is not positive, α must be zero to enforce the correct condition.

Assuming the possibility of going directly between neighbourhoods leads us to include product of continuous variables in the determinants of constraints (α-C). These products make our formulation for the H-TSPN to become non-convex. However, in the H-TSPHN, since two of the three arguments of each determinant are fixed, the α constraints become linear. This important difference motivates the computational comparison between the two formulations in Section 5.

Now, we check whether the pairs of determinants

$$\det(P|P_B^1P_B^2),\ \det(Q|P_B^1P_B^2) \quad \text{and} \quad \det(P_B^1|PQ),\ \det(P_B^2|PQ) \tag{1}$$

have the same sign, we introduce the binary variable β, that is one if the corresponding pair has the same sign, and zero otherwise.

Therefore, the correct value of the variable β can be enforced by the following constraint of the variables α.

$$\begin{aligned}
\beta(PQ|P_B^1P_B^2) &= \alpha(P|P_B^1P_B^2)\alpha(Q|P_B^1P_B^2) + \left[1-\alpha(P|P_B^1P_B^2)\right]\left[1-\alpha(Q|P_B^1P_B^2)\right], \\
\beta(P_B^1P_B^2|PQ) &= \alpha(P_B^1|PQ)\alpha(P_B^2|PQ) + \left[1-\alpha(P_B^1|PQ)\right]\left[1-\alpha(P_B^2|PQ)\right].
\end{aligned}$$

This condition can be equivalently written using an auxiliary binary variable γ that models the product of the α variables:

$$\begin{aligned}
\beta(PQ|P_B^1P_B^2) &= 2\gamma(PQ|P_B^1P_B^2) - \alpha(P|P_B^1P_B^2) - \alpha(Q|P_B^1P_B^2) + 1, \\
\beta(P_B^1P_B^2|PQ) &= 2\gamma(P_B^1P_B^2|PQ) - \alpha(P_B^1|PQ) - \alpha(P_B^2|PQ) + 1,
\end{aligned} \tag{β-C}$$

We observe that γ can be linearised using the following constraints:

$$\begin{aligned}
&\gamma(PQ|P_B^1P_B^2) \le \alpha(P|P_B^1P_B^2), && \gamma(P_B^1P_B^2|PQ) \le \alpha(P_B^1|PQ), \\
&\gamma(PQ|P_B^1P_B^2) \le \alpha(Q|P_B^1P_B^2), && \gamma(P_B^1P_B^2|PQ) \le \alpha(P_B^2|PQ), \\
&\gamma(PQ|P_B^1P_B^2) \ge \alpha(P|P_B^1P_B^2) + \alpha(Q|P_B^1P_B^2) - 1, && \gamma(P_B^1P_B^2|PQ) \ge \alpha(P_B^1|PQ) + \alpha(P_B^2|PQ) - 1.
\end{aligned} \tag{γ-C}$$

Later, we need to check whether there exists any coincidence of the sign of determinants, so we define the binary variable δ, which is one if the segments do not intersect and zero, otherwise. This condition can be modelled by using the following disjunctive constraints:

$$\frac{1}{2}\left[\beta(PQ|P_B^1P_B^2) + \beta(P_B^1P_B^2|PQ)\right] \le \delta(PQ|P_B^1P_B^2) \le \beta(PQ|P_B^1P_B^2) + \beta(P_B^1P_B^2|PQ). \tag{δ-C}$$

Indeed, the above constraints state that if there exists a sign coincidence in any of the two pairs of determinants in (1) , then δ is one to satisfy the left constraint, and the right one is always fulfilled. However, if none of the signs of the pairs of determinants is the same, then the second constraint is zero and δ is null.

Finally, we need to check that

$$\overline{PQ} \cap B'' = \emptyset, \quad \forall B'' \in \mathcal{B}, \quad \iff \quad \delta(PQ|P_{B''}^1P_{B''}^2) = 1, \quad \forall B'' \in \mathcal{B}.$$

Hence, if we denote by $\varepsilon(PQ)$ the binary variable that is one if the previous condition is satisfied for

all $B'' \in \mathcal{B}$ and zero otherwise, this variable can be represented by means of the following inequalities:

$$\left[\sum_{B'' \in \mathcal{B}} \delta(PQ|P^1_{B''}P^2_{B''}) - |\mathcal{B}|\right] + 1 \leq \varepsilon(PQ) \leq \frac{1}{|\mathcal{B}|} \sum_{B'' \in \mathcal{B}} \delta(PQ|P^1_{B''}P^2_{B''}). \qquad (\varepsilon\text{-C})$$

If there is, at least, a barrier $B'' \in \mathcal{B}$ that intersects the segment $\overline{PQ}$, then $\delta(PQ|P^1_{B''}P^2_{B''})$ is zero and the second inequality forces ε to be zero because the right hand side is fractional and the first inequality is nonpositive. Nevertheless, if no barrier intersects the segment $\overline{PQ}$, then ε is equal to one, because the left-hand side of the first inequality is one and the right-hand side of the second inequality is also one.

Based on the description above, we can identify the set of actual edges of the graph G using the variables ε as follows:

$$E_X = \{(P,Q) : P,Q \in V_X, \varepsilon(PQ) = 1, P \neq Q\}, \quad X \in \{SPP, TSPH, TSP\}.$$

This representation of E_X with $X \in \{SPP, TSPH, TSP\}$ will be exploited in the formulations described in the following subsections.

It is interesting to note that $E_{\mathcal{B}}$ is a fixed set whose edges can be computed using Remark 1. Then, the variables ε can be prefixed in advance. However, the edges in $E_X \setminus E_{\mathcal{B}}$ depend on the points selected in the neighbourhoods, as explained before.

3.3. A formulation for the H-SPPN

The idea of the formulation of the H-SPPN is to extend the classical formulation of the Shortest Path Problem by taking into account the description of the graph G_{SPP} in Subsection 2.2.

The formulation describes the path that the drone can follow by taking into account the edges of the induced graph. Let $P, Q \in V_{\text{SPP}}$ and let $y(PQ)$ be the binary variable that is one if the drone goes from P to Q. Then, the inequalities

$$y(PQ) \leq \varepsilon(PQ), \qquad (\text{y-C})$$

assure that the drone can go from P to Q only if the segment $\overline{PQ}$ does not cross any barrier.

Taking into account the constraints explained in the subsections above, the following MINLP formulation is valid for H-SPPN.

$$\text{minimize} \quad \sum_{(P,Q)\in E_{\text{SPP}}} d(PQ)y(PQ) \tag{H-SPPN}$$

$$\text{subject to} \quad \sum_{\{Q:(P,Q)\in E_{\text{SPP}}\}} y(PQ) - \sum_{\{Q:(Q,P)\in E_{\text{SPP}}\}} y(QP) = \begin{cases} 1, & \text{if } P \in V_S, \\ 0, & \text{if } P \in V_{\mathcal{B}}, \\ -1, & \text{if } P \in V_T. \end{cases},$$

$$(\alpha\text{-C}), (\beta\text{-C}), (\gamma\text{-C}), (\delta\text{-C}) \quad \forall P, Q \in V_{\text{SPP}}, \quad \forall P_B^1, P_B^2 \in V_{\mathcal{B}},$$

$$(\varepsilon\text{-C}), (\text{y-C}), (\text{d-C}) \quad \forall P, Q \in V_{\text{SPP}},$$

$$(\text{N-C}) \quad \forall P \in V_S \cup V_T.$$

The objective function minimises the length of the path followed by the drone at the edges of the induced graph G_{SPP}. The first constraints are the flow conservation constraints that ensure connectivity (they will be also used in the formulations of the H-TSPN), the second constraints represent the sets E_S and E_T and the third ones state that the selected points must be in their respective neighbourhoods. Figure 7 reports the optimal solution for the example of the H-SPPN.

3.3.1. Reformulating the H-SPPN

The formulation for the H-SPPN presented above can be simplified taking into account the following observation.

Proposition 1. *There exist two finite dominating sets, N_S^* and N_T^*, of possible candidates to be starting points in N_S and terminal points in N_T, respectively. Moreover,*

$$N_S^* = \{P_S(P_B^i) : P_S(P_B^i) = \arg\min_{P_S \in N_S} \|P_S - P_B^i\|, \, (P_S, P_B^i) \in E_S \text{ and } P_B^i \in V_{\mathcal{B}}\},$$
$$N_T^* = \{P_T(P_B^i) : P_T(P_B^i) = \arg\min_{P_T \in N_T} \|P_B^i - P_T\|, \, (P_B^i, P_T) \in E_T \text{ and } P_B^i \in V_{\mathcal{B}}\}.$$

Proof. Note that the points chosen in N_S and N_T in an optimal solution for H-SPPN must be those that give the minimum distances to the points of the first and last barriers visited in any optimal solution, respectively. Therefore, N_S^* and N_T^* must contain, at most, the points in the sets

$$\{P_S(P_B^i) : P_S(P_B^i) = \arg\min_{P_S \in N_S} \|P_S - P_B^i\|, \, (P_S, P_B^i) \in E_S \text{ and } P_B^i \in V_{\mathcal{B}}\},$$
$$\{P_T(P_B^i) : P_T(P_B^i) = \arg\min_{P_T \in N_T} \|P_B^i - P_T\|, \, (P_B^i, P_T) \in E_T \text{ and } P_B^i \in V_{\mathcal{B}}\},$$

as claimed. □

Therefore, we can compute, *'a priori'*, the sets N_X^*, $X \in \{S, T\}$. For each $P_B^i \in V_{\mathcal{B}}$, the corresponding point in the source neighbourhood $P_X(P_B^i)$ is computed by solving the following convex problem:

$$P_X(P_B^i) = \begin{array}{ll} \underset{P_X \in N_X}{\arg\min} & d(P_X P_B^i) \\ \text{subject to} & (\alpha\text{-C}), (\beta\text{-C}), (\gamma\text{-C}), (\delta\text{-C}), \quad \forall P_B^1, P_B^2 \in V_{\mathcal{B}}, \\ & (\varepsilon\text{-C}), \\ & \varepsilon(P_X P_B^i) = 1, \\ & \|P_X - P_B^i\| \leq d(P_X P_B^i). \end{array} \tag{N_X^*}$$

Then, the visibility graph is constructed by means of these dominating sets. This graph, required to address the new formulation, uses the previous graph G_{SPP} in which we increase the new edges that connect the initial and terminal points $P_S \in N_S$ and $P_T \in N_T$, respectively, with the corresponding points in N_S^* and N_T^*. Moreover, we also augment the edges connecting the points in the dominating sets with the extreme points of the segments where the minimal distances are attained. The reader may note that since the optimisation model minimises the overall distance, in an optimal solution P_S must belong to N_S^* and P_T to N_T^*.

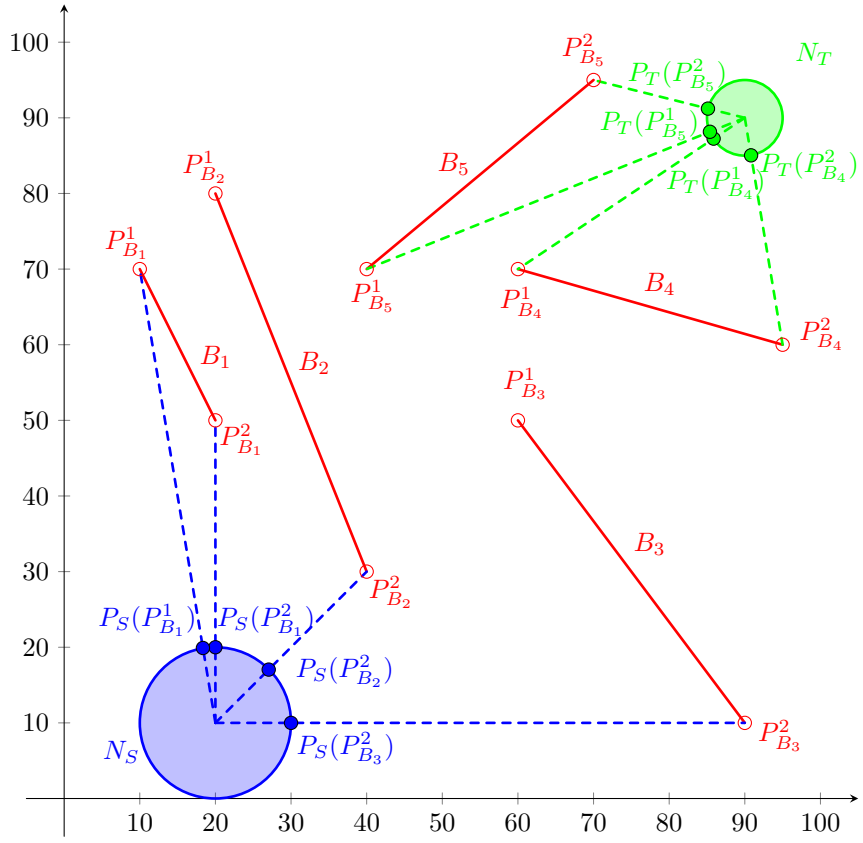
Therefore, the following sets are needed to build the new visibility graph induced by the dominating sets.

- $V_S^* = N_S^*$. Dominating set of points associated with the neighbourhood N_S.
- $V_T^* = N_T^*$. Dominating set of points associated with the neighbourhood N_T.
- $E_S^{\text{int}} = \{(P_S, P_S(P_B^i)) : P_S \in V_S \text{ and } P_B^i \in V_{\mathcal{B}}\}$. Set of edges that join the selected point $P_S \in N_S$ with each point $P_S(P_B^i)$ in the dominating set N_S^*.
- $E_S^{\text{ext}} = \{(P_S(P_B^i), P_B^i) : P_B^i \in V_{\mathcal{B}}\}$. Set of edges joining the point $P_S(P_B^i)$ with its respective P_B^i in $V_{\mathcal{B}}$.
- $E_S^* = E_S^{\text{int}} \cup E_S^{\text{ext}}$.
- $E_T^{\text{int}} = \{(P_T(P_B^i), P_T) : P_B^i \in V_{\mathcal{B}} \text{ and } P_T \in V_T\}$. Set of edges that join each point $P_S(P_B^i)$ in the dominating set N_S^* with the point selected $P_T \in N_T$.
- $E_T^{\text{ext}} = \{(P_B^i, P_T(P_B^i)) : P_B^i \in V_{\mathcal{B}}$. Set of edges joining the point P_B^i in $V_{\mathcal{B}}$ with its respective $P_T(P_B^i)$.
- $E_T^* = E_T^{\text{int}} \cup E_T^{\text{ext}}$.

We define the graph $G_{\text{SPP}}^* = (V_{\text{SPP}}^*, E_{\text{SPP}}^*)$, where $V_{\text{SPP}}^* = V_S^* \cup V_{\text{SPP}} \cup V_T^*$ and $E_{\text{SPP}}^* = E_S^* \cup E_{\mathcal{B}} \cup E_T^*$.

The major advantage of this approach is that, once the sets N_S^* and N_T^* are calculated beforehand, the entire graph G_{SPP}^* is fixed because the incident edges can be computed for each point $P_S(P_B^i)$ and $P_T(P_B^i)$, $P_B^i \in V_{\mathcal{B}}$, separately. Figure 6 shows how the dominating sets N_S^* and N_T^* are calculated for an instance of the H-SPPN.

Figure 6: Illustration of the dominating sets associated with an instance of the H-SPPN.



Defining again the variables d and y for the edges in E^*_{SPP}, the new formulation for the H-SPPN can be expressed as the following simplified form:

$$\text{minimize} \quad \sum_{(P,Q)\in E^*_{\text{SPP}}} d(PQ)y(PQ) \qquad \text{(H-SPPN}^*\text{)}$$

$$\text{subject to} \quad \sum_{\{Q:(P,Q)\in E^*_{\text{SPP}}\}} y(PQ) - \sum_{\{Q:(Q,P)\in E^*_{\text{SPP}}\}} y(QP) = \begin{cases} 1, & \text{if } P \in V_S, \\ 0, & \text{if } P \in V^*_S \cup V_{\mathcal{B}} \cup V^*_T, \\ -1, & \text{if } P \in V_T. \end{cases},$$

$$\text{(d-C)} \quad \forall P, Q \in V^*_{\text{SPP}},$$

$$\text{(N-C)} \quad \forall P \in V_S \cup V_T.$$

Let us assume that the finite dominating sets N^*_S and N^*_T are given.

Proposition 2. *Let $n = |\mathcal{B}|$. The* H-SPPN *can be solved in polynomial time $O(n^3)$.*

Proof. The proof follows using the finite dominating sets N^*_S and N^*_T. First, it is clear that the cardinality of these finite dominating sets is $O(n)$ since there is one point in each set for each endpoint of a segment in $\mathcal{B}$. Next, we recall that given a set of polygonal obstacles with $O(n)$ vertices, the shortest linear size map from any given point can be computed in $O(n \log n)$ time. Once you have that map, finding the shortest path to any other point in the region is done in $O(n)$ time (Mitchell, 2017). Therefore, the H-SPPN can be solved as follows.

For each point $P_S(P_B^i) \in N_S^*$ we compute the shortest path map of linear size and solve the problem of the shortest path with respect to all $O(n)$ points in N_T^*. In general, this operation takes $O(n \log n + n^2)$. The same operation has to be repeated for all $O(n)$ points in N_S^*. The best among all those paths is the shortest path from S to T. Therefore, the overall complexity to solve H-SPPN is $O(n^3)$.

□

3.4. A formulation for the H-TSPHN

The rationale of the formulation for the H-TSPHN is to consider the variant called Steiner TSP (STSP) (see Letchford et al. (2013)). In this problem, the task is to find a minimal cost route that visits a set of required nodes ($V_{\mathcal{N}}$). During the route, vertices (V_{TSPH}) can be visited more than once, and edges (E_{TSPH}) may be traversed more than once.

It is well-known that it is possible to convert any instance of the STSP into an instance of the standard TSP, by computing the shortest paths between every pair of required nodes, when these nodes are fixed. However, in our problem, since the points in the neighbourhoods are not fixed, the H-SPPN cannot be used directly to derive an optimal solution for the H-TSPHN, although it may produce a good approximation to generate lower bounds for the H-TSPHN that allows to solve larger instances.

Similarly as in the H-SPPN, we use a single-commodity flow formulation to ensure connectivity. We can assume that the neighbourhood N_1 is required and that the drone departs from that depot (assuming that it is N_1) with $|\mathcal{N}| - 1$ units of commodity and one unit of commodity must be delivered to each neighbourhood. Then, for each edge $(P, Q) \in E_{\text{TSPH}}$, we define the following variables:

- $y(PQ)$, binary variable equal to one if the drone goes from P to Q.
- $g(PQ)$, non-negative continuous variable that represents the amount of commodity passing through the edge (P, Q).

Hence, we can adjust the single-commodity flow formulation to the induced graph G_{TSPH} as follows:

$$\text{minimize} \quad \sum_{(P,Q)\in E_{\text{TSPH}}} d(PQ)y(PQ) \tag{H-TSPHN}$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{\{Q:(P_N,Q)\in E_{\mathcal{N}}\}} y(P_NQ) \geq 1, && \forall P_N \in V_{\mathcal{N}}, \\
& \sum_{\{Q:(P,Q)\in E_{\text{TSPH}}\}} y(PQ) = \sum_{\{Q:(Q,P)\in E_{\text{TSPH}}\}} y(QP), && \forall P \in V_{\text{TSPH}}, \\
& \sum_{\{Q:(Q,P_N)\in E_{\mathcal{N}}\}} g(QP_N) - \sum_{\{Q:(P_N,Q)\in E_{\mathcal{N}}\}} g(P_NQ) = 1, && \forall P_N \in V_{\mathcal{N}} \setminus \{P_{N_1}\}, \\
& \sum_{\{Q:(Q,P_B^i)\in E_{\text{TSPH}}\}} g(QP_B^i) - \sum_{\{Q:(P_B^i,Q)\in E_{\text{TSPH}}\}} g(P_B^iQ) = 0, && \forall P_B^i \in V_{\mathcal{B}}, \\
& g(PQ) \leq (|\mathcal{N}|-1)y(PQ), && \forall (P,Q) \in E_{\text{TSPH}}, \\
& (\alpha\text{-C}), (\beta\text{-C}), (\gamma\text{-C}), (\delta\text{-C}) && \forall P,Q \in V_{\text{TSPH}}, \quad \forall P_B^1, P_B^2 \in V_{\mathcal{B}}, \\
& (\varepsilon\text{-C}), (\text{y-C}), (\text{d-C}) && \forall P,Q \in V_{\text{TSPH}}, \\
& (\text{N-C}) && \forall P_N \in V_{\mathcal{N}}.
\end{aligned}$$

The first group of constraints imposes that the drone departs from each neighbourhood. The second block of constraints is the flow-conservation constraints. The third inequalities ensure that one unit of commodity is delivered to each of the required neighbourhood. The fourth ensures that the fictitious nodes at the end of the barriers do not consume commodity. Finally, the last inequalities enforce that some commodity goes throughout an edge only if this edge is used in the tour. The inequalities (α-C), (β-C), (γ-C), (δ-C), (ε-C), (y-C), (d-C), (N-C) require the variables of the problem to be well defined. Figure 8 shows the optimal solution for the example of the H-TSPHN.

Proposition 3. *The* H-TSPHN *is NP-complete.*

Note that, once a point is fixed in each neighbourhood, the problem that results in the induced graph G_{TSPH} is the Steiner TSP. This problem is an extension of the TSP, that is NP-complete (Papadimitriou and Steiglitz, 1977).

3.4.1. Reformulating the H-TSPHN

The idea of computing a dominating set that represents each of the neighbourhoods in the H-SPPN can be adopted to reformulate the H-TSPHN in the same way. The dominating sets are stated in the next proposition.

Proposition 4. *Given a neighbourhood $N \in \mathcal{N}$, there exists a finite dominating set, N^* of possible candidates to be in N. Moreover,*

$$N^* = \{P_N(P_B^i, P_{B'}^j) : P_N(P_B^i, P_{B'}^j) = \underset{P_N \in N}{\arg\min} \|P_B^i - P_N\| + \|P_N - P_{B'}^j\| \text{ and } (P_B^i, P_N), (P_N, P_{B'}^j) \in E_{\mathcal{N}}\}.$$

Proof. The way a drone visits a neighbourhood N is

$$P_B^i \longrightarrow P_N \longrightarrow P_{B'}^j,$$

for some points P_B^i, $P_{B'}^j \in V_{\mathcal{B}}$, since, by **A4**, there does not exist a rectilinear path that joins any pair of neighbourhoods. Therefore, the points chosen in an optimal solution for H-TSPHN must be those that produce the minimum distances to the points of the barriers visited previously and next visited in the optimal solution. Therefore, N^* must be composed, at most, of the points in the set

$$\{P_N(P_B^i, P_{B'}^j) = \arg\min_{P_N \in N} \|P_B^i - P_N\| + \|P_N - P_{B'}^j\| : (P_B^i, P_N), (P_N, P_{B'}^j) \in E_{\mathcal{N}}\},$$

which completes the proof. □

Again, we can compute the respective dominating set N^* of a neighbourhood $N \in \mathcal{N}$ by solving a convex problem for each pair of barrier endpoints:

$$N^* = \{P_N(P_B^i, P_{B'}^j) : P_N(P_B^i, P_{B'}^j) = \arg\min_{P_N \in N} \|P_B^i - P_N\| + \|P_N - P_{B'}^j\|, \varepsilon(P_B^i P_N) = 1 \text{ and } \varepsilon(P_N P_{B'}^j) = 1\}.$$

The graph induced by the precomputed dominating sets is described in terms of the following sets:

- $V_{\mathcal{N}}^* = \{N^* : N \in \mathcal{N}\}$. Union of the dominating sets associated with each neighbourhood.
- $E_N^{\text{int}} = \{(P_N(P_B^i, P_{B'}^j), P_N), (P_N, P_N(P_B^i, P_{B'}^j)) : P_B^i \in V_{\mathcal{B}}, P_N \in N \text{ and } P_{B'}^j \in V_{\mathcal{B}}\}$. Set of edges joining the point selected $P_N \in N$ with each point $P_N(P_B^i, P_{B'}^j)$ in the dominating set N^*.
- $E_N^{\text{ext}} = \{(P_B^i, P_N(P_B^i, P_{B'}^j)), (P_N(P_B^i, P_{B'}^j), P_{B'}^j) : P_B^i \in V_{\mathcal{B}} \text{ and } P_{B'}^j \in V_{\mathcal{B}}\}$. Set of edges joining the point $P_N(P_B^i, P_{B'}^j)$ with its respective P_B^i and $P_{B'}^j$ in $V_{\mathcal{B}}$.
- $E_N^* = E_N^{\text{int}} \cup E_N^{\text{ext}}$. Set of edges associated with the neighbourhood N.
- $E_{\mathcal{N}}^* = \{E_N^* : N \in \mathcal{N}\}$. Union of the edges of every neighbourhood.

We define the graph $G_{\text{TSPH}}^* = (V_{\text{TSPH}}^*, E_{\text{TSPH}}^*)$, where $V_{\text{TSPH}}^* = V_{\text{TSPH}} \cup V_{\mathcal{N}}^*$ and $E_{\text{TSPH}}^* = E_{\mathcal{N}}^* \cup E_{\mathcal{B}}$.

Again, the sets N^* for each $N \in \mathcal{N}$ can be computed in advance so that the whole graph G_{TSPH}^* is fixed. The new formulation for the H-TSPHN can be represented as the next simplified program:

$$\text{minimize} \quad \sum_{(P,Q)\in E^*_{\text{TSPH}}} d(PQ)y(PQ) \tag{H-TSPHN*}$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{\{Q:(P_N,Q)\in E_N^{\text{int}}\}} y(P_NQ) \geq 1, && \forall P_N \in V_{\mathcal{N}},\\
& \sum_{\{Q:(P,Q)\in E^*_{\text{TSPH}}\}} y(PQ) = \sum_{\{Q:(Q,P)\in E^*_{\text{TSPH}}\}} y(QP), && \forall P \in V^*_{\text{TSPH}},\\
& \sum_{\{Q:(Q,P_N)\in E_N^{\text{int}}\}} g(QP_N) - \sum_{\{Q:(P_N,Q)\in E_N^{\text{int}}\}} g(P_NQ) = 1, && \forall P_N \in V_{\mathcal{N}} \setminus \{P_{N_1}\},\\
& \sum_{\{Q:(Q,P)\in E^*_{\text{TSPH}}\}} g(QP) - \sum_{\{Q:(P,Q)\in E^*_{\text{TSPH}}\}} g(PQ) = 0, && \forall P \in V_{\mathcal{B}} \cup V^*_{\mathcal{N}},\\
& g(PQ) \leq (|\mathcal{N}|-1)y(PQ), && \forall (P,Q) \in E^*_{\text{TSPH}},\\
& \text{(d-C)} \quad \forall P,Q \in V^*_{\text{TSPH}},\\
& \text{(N-C)} \quad \forall P_N \in V_{\mathcal{N}}.
\end{aligned}$$

The reader may note that the maximum number of dominating points associated with an instance $(\mathcal{N},\mathcal{B})$ of the H-TSPHN is $\frac{1}{2}|\mathcal{N}||\mathcal{B}|(|\mathcal{B}|-1)$. Therefore, to obtain all sets N^* for each $N \in \mathcal{N}$, it is required to solve, at most, $\frac{1}{2}|\mathcal{N}||\mathcal{B}|(|\mathcal{B}|-1)$ convex programs which may be computationally expensive but polynomial.

3.5. Relaxing the assumptions of the problem: The H-TSPN

In this subsection, we analyse the differences between H-TSPHN and H-TSPN, where it is possible to move from one neighbourhood to another one without crossing any barrier. The main difference lies in the description of the edges of the graph induced by the neighbourhoods and the endpoints of the barriers.

By taking the same approach as before, the sets that describe the graph in the new case are $V_{\mathcal{N}}$, $V_{\mathcal{B}}$, V_{TSP} and E_{TSP}, as described in Subsection 2.3.

$$\text{minimize} \quad \sum_{(P,Q)\in E_{\text{TSP}}} d(PQ)y(PQ) \tag{H-TSPN}$$

subject to

$$\sum_{\{Q:(P_N,Q)\in E_{\mathcal{N}}\}} y(P_NQ) \geq 1, \qquad \forall P_N \in V_{\mathcal{N}},$$

$$\sum_{\{Q:(P,Q)\in E_{\text{TSP}}\}} y(PQ) = \sum_{\{Q:(Q,P)\in E_{\text{TSP}}\}} y(QP), \quad \forall P \in V_{\text{TSP}},$$

$$\sum_{\{Q:(Q,P_N)\in E_{\mathcal{N}}\}} g(QP_N) - \sum_{\{Q:(P_N,Q)\in E_{\mathcal{N}}\}} g(P_NQ) = 1, \qquad \forall P_N \in V_{\mathcal{N}} \setminus \{P_{N_1}\},$$

$$\sum_{\{Q:(Q,P_B^i)\in E_{\text{TSP}}\}} g(QP_B^i) - \sum_{\{Q:(P_B^i,Q)\in E_{\text{TSP}}\}} g(P_B^iQ) = 0, \qquad \forall P_B^i \in V_{\mathcal{B}},$$

$$g(PQ) \leq (|\mathcal{N}|-1)y(PQ), \qquad \forall (P,Q) \in E_{\text{TSP}},$$

$$(\alpha\text{-C}), (\beta\text{-C}), (\gamma\text{-C}), (\delta\text{-C}) \quad \forall P,Q \in V_{\text{TSP}}, \quad \forall P_B^1, P_B^2 \in V_{\mathcal{B}},$$

$$(\varepsilon\text{-C}), (\text{y-C}), (\text{d-C}) \quad \forall P,Q \in V_{\text{TSP}},$$

$$(\text{N-C}) \quad \forall P_N \in V_{\mathcal{N}}.$$

The formulation described above is analogous to those detailed in (H-TSPHN). The difference between the set of edges in the H-TSPN with respect to the graph in H-TSPHN is that, in the former case, the edges that join each pair of neighbourhoods must be considered. This fact leads to including the product of continuous variables in the constraints α of the model that represent the determinants that determine whether the segment joining the two variable points in the neighbourhoods crosses any barrier or not. These products make the problem a non-convex, quadratically constrained program. In Figure 9, the optimal solution for the example of the H-TSPN is represented. It shows a direct path joining the blue and orange neighbourhoods.

Figure 7: Optimal solution for the instance of the H-SPPN

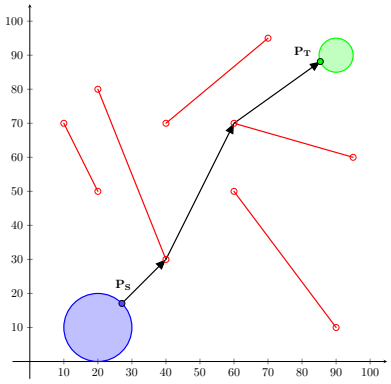


Figure 8: Optimal solution for the instance of the H-TSPHN

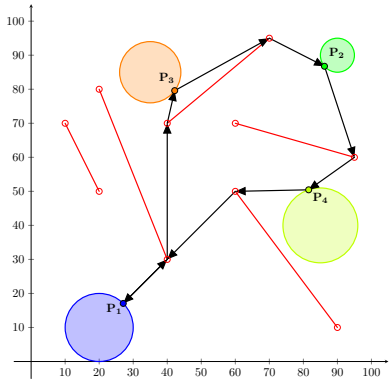
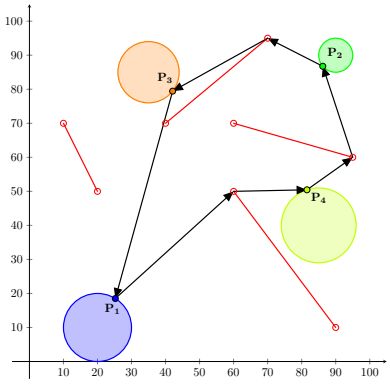


Figure 9: Optimal solution for the instance of the H-TSPN



3.5.1. Real-world scenario

In this subsection, we include an example from a real neighbourhood of the city map of Cordoba in the south of Spain. In order to make surveillance drone activities in some parks of the district, the

Figure 10: Satellite view for the real-world scenario

Figure 11: Block view for the real-world scenario

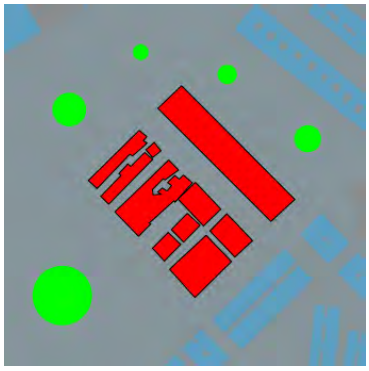
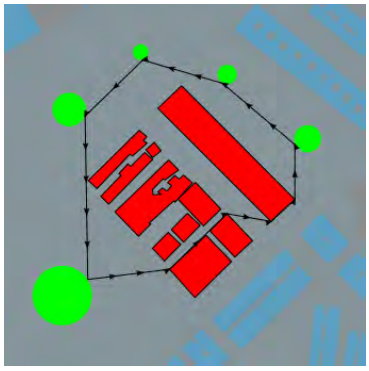


Figure 12: Optimal solution for the real-world scenario



neighbourhoods, represented as circles, must be visited by the drone by taking into account buildings that can not be crossed. Figure 10 shows a part of the district that contains three parks that must be inspected and the buildings the drone cannot cross. Figure 11 represents a block view that includes, in a schematic way, the barriers and neighbourhoods that must be visited. Finally, the optimal solution for the H-TSPN for this real scenario is given by the tour included in Figure 12.

The data describing this case study can be downloaded from Puerto and Valverde (2023).

4. Strengthening the formulations

4.1. Preprocessing

In this subsection, a pre-processing result is proposed that allows one to fix some variables. It is based on analysing the relative position between the neighbourhoods and the barriers. Specifically, we present a sufficient condition that ensures that there are some barriers whose endpoints cannot be incident at the edges of $E_{\mathcal{N}}$ so that it is not necessary to include them in $E_{\mathcal{N}}$.

Let us denote

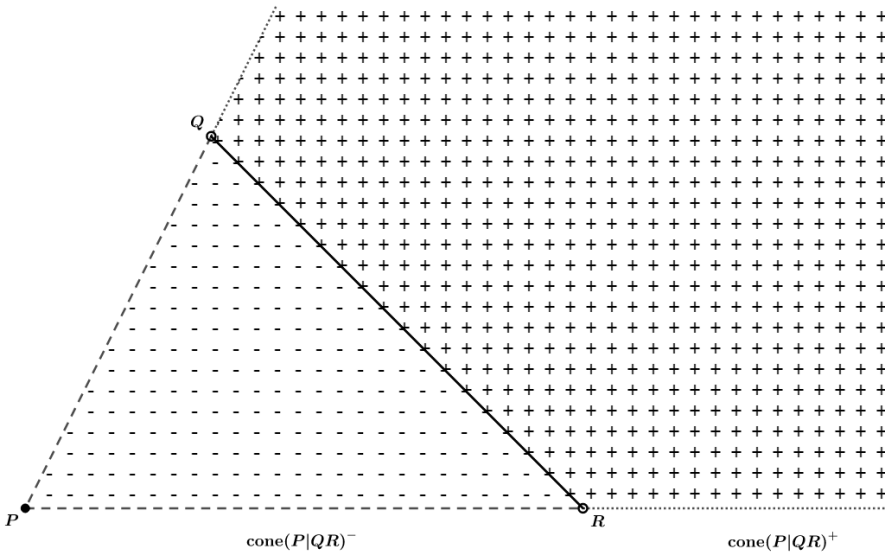
$$\begin{aligned}
\text{cone}(P|QR) &:= \{\mu_1\overrightarrow{PQ} + \mu_2\overrightarrow{PR} : \mu_1, \mu_2 \geq 0\},\\
\text{cone}(P|QR)^- &:= \{\mu_1\overrightarrow{PQ} + \mu_2\overrightarrow{PR} : \mu_1, \mu_2 \geq 0,\ \mu_1 + \mu_2 \leq 1\},\\
\text{cone}(P|QR)^+ &:= \{\mu_1\overrightarrow{PQ} + \mu_2\overrightarrow{PR} : \mu_1, \mu_2 \geq 0,\ \mu_1 + \mu_2 \geq 1\}.
\end{aligned}$$

Note that $\text{cone}(P|QR)$ is the union of $\text{cone}(P|QR)^-$ and $\text{cone}(P|QR)^+$. It is also important to note that $\text{cone}(P|QR)^+$ represents the subset of points P' in the plane whose segments $\overline{PP'}$ cross the barrier $\overline{QR}$ (see Figure 13), i.e.

$$\text{cone}(P|QR)^+ = \{P' : \overline{PP'} \cap \overline{QR} \neq \emptyset\}.$$

Let $B = \overline{P_B^1 P_B^2} \in \mathcal{B}$ a barrier. In the following proposition, we give a sufficient condition to not include the edge (P_N, P_B^i) in $E_{\mathcal{N}}$.

Figure 13: Representation of the cone generated by the point P and the line segment $\overline{QR}$.



Proposition 5. *Let* $B' = \overline{P^1_{B'}P^2_{B'}} \in \mathcal{B}$ *and* $\text{cone}(P^i_B|P^1_{B'}P^2_{B'})^+$ *the conical hull generated by these points. If*

$$N \subset \bigcup_{B' \in \mathcal{B}} \text{cone}(P^i_B|P^1_{B'}P^2_{B'})^+,$$

then $(P_N, P^i_B) \notin E_{\mathcal{N}}$.

Proof. If $P_N \in N$, then there exists a $B' \in \mathcal{B}$ such that $P_N \in \text{cone}(P^i_B|P^1_{B'}P^2_{B'})^+$. Therefore, $\overline{P^i_B P_N} \cap B' \neq \emptyset$ and $(P_N, P^i_B) \notin E_{\mathcal{N}}$. □

One can easily check computationally the condition of the previous proposition by using the following procedure. First, we deal with the case where neighbourhoods are segments. Let $N = \overline{P^1_N P^2_N}$ be a line segment and r_N be the straight line that contains the line segment N represented as:

$$r_N : P^1_N + \lambda \overrightarrow{P^1_N P^2_N}, \qquad \lambda \in \mathbb{R}.$$

In Figure 14 an example of the Algorithm 1 is described to check if there exists a rectilinear path that joins the solid point P^i_B in the red barrier defined by the extreme points P^i_B and P^j_B with the segment $\overline{P^1_N P^2_N}$. First, the dashed straight lines $r(P^i_B, P^j_{B''})$ generated by P^i_B and each point of the barrier are computed. Second, the straight line r_N is determined by P^1_N and P^2_N. Then, each of the straight lines $r(P^i_B, P^j_{B''})$ is intersected with r_N obtaining the points Q^1_1, Q^1_2, Q^2_1 and Q^2_2. Each of these points has two associated parameter values μ and λ with respect to the straight lines $r(P^i_B, P^j_{B''})$ and r_N, respectively. Finally, the points are ordered in nondecreasing sequence with respect to the λ values. Since

$$M = \lambda^1_1 \leq 0, 1 \leq \lambda^2_1 = 2,$$

the segment $\overline{P^1_N P^2_N}$ is fully included in $\text{cone}(P^i_B|P^1_1 P^2_1)^+$ and, therefore, (P^i_B, P_N) is not included in $E_{\mathcal{N}}$.

Note that this algorithm also allows us to decide whether the drone can access a barrier point from

Algorithm 1: Checking computationally whether $(P_N, P_B^i) \notin E_{\mathcal{N}}$ when N is a segment.

Initialization: Let P_B^i be the point of the edge (P_N, P_B^i) to check whether $(P_N, P_B^i) \notin E_{\mathcal{N}}$.
Set $points = \{P_N^1, P_N^2\}$, $lambdas = \{0, 1\}$.

for $B'' \in \mathcal{B}$ **do**
  for $j \in \{1, 2\}$ **do**
    Compute the straight line

$$r(P_B^i, P_{B''}^j) = P_B^i + \mu_{B''}^j \overrightarrow{P_B^i P_{B''}^j},$$

    that contains the points P_B^i and $P_{B''}^j$.
    Intersect $r(P_B^i, P_{B''}^j)$ and r_N at the point $Q_{B''}^j$ and compute $\overline{\mu}_{B''}^j$ such that

$$Q_{B''}^j = P_B^i + \overline{\mu}_{B''}^j \overrightarrow{P_B^i P_{B''}^j}.$$

    if $|\overline{\mu}_{B''}^j| \geq 1$ **then**
      Compute $\lambda_{B''}^j$ such that

$$Q_{B''}^j = P_N^1 + \lambda_{B''}^j \overrightarrow{P_N^1 P_N^2}.$$

      if $\overline{\mu}_{B''}^j \geq 1$ **then**
        Include $\lambda_{B''}^j$ in $lambdas$.
      else
        if $\lambda_{B''}^j \geq 0$ **then**
          Set $\lambda_{B''}^j = M << 0$ and include it in $lambdas$.
        else
          Set $\lambda_{B''}^j = M >> 0$ and include it in $lambdas$.

Order the set $lambdas$ in non-decreasing order.
If it is satisfied that

$$\begin{aligned}
\min\{\lambda_{B'}^1, \lambda_{B'}^2\} \leq 0 \leq \max\{\lambda_{B'}^1, \lambda_{B'}^2\}, &\quad \text{for some } B' \in \mathcal{B}, \\
\min\{\lambda_{B'}^1, \lambda_{B'}^2\} \leq 1 \leq \max\{\lambda_{B'}^1, \lambda_{B'}^2\}, &\quad \text{for some } B' \in \mathcal{B}, \\
\min\{\lambda_{B'}^1, \lambda_{B'}^2\} \leq \lambda_{B''}^j \leq \max\{\lambda_{B'}^1, \lambda_{B'}^2\}, &\quad \text{for some } B' \in \mathcal{B} \setminus \{B''\}, \quad \forall \lambda_{B''}^j \in lambdas \setminus \{M\},
\end{aligned}$$

or

$$\min\{\lambda_{B'}^1, \lambda_{B'}^2\} \leq 0, 1 \leq \max\{\lambda_{B'}^1, \lambda_{B'}^2\}, \quad \text{for some } B' \in \mathcal{B},$$

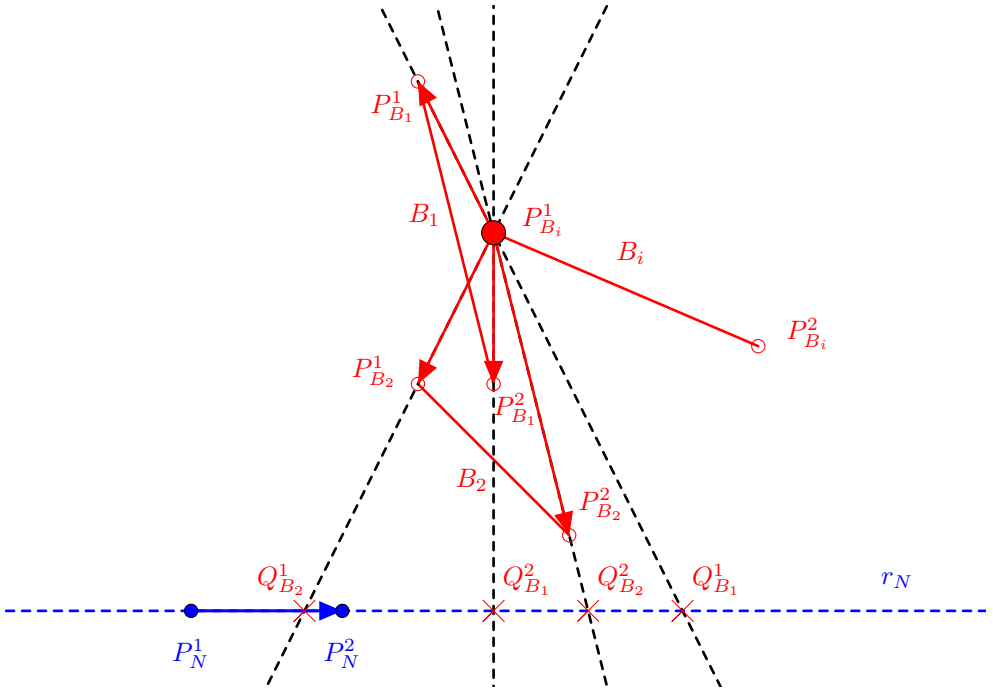
then $(P_N, P_B^i) \notin E_{\mathcal{N}}$.

any point in the neighbourhood N. It is enough to check in (15) that

$$0 \notin \left[\min\{\lambda_{B'}^1, \lambda_{B'}^2\}, \max\{\lambda_{B'}^1, \lambda_{B'}^2\}\right] \quad \text{and} \quad 1 \notin \left[\min\{\lambda_{B'}^1, \lambda_{B'}^2\}, \max\{\lambda_{B'}^1, \lambda_{B'}^2\}\right], \quad \forall B' \in \mathcal{B}.$$

For the case where N is an ellipse, the same rationale can be followed. The idea is to generate the largest line segment contained in the ellipse and to repeat the procedure in Algorithm 1. Let F_1 and F_2 be the focal points of N.

Figure 14: Example of the Algorithm 1



	$Q^1_{B_1}$	P^1_N	$Q^1_{B_2}$	P^2_N	$Q^2_{B_1}$	$Q^2_{B_2}$
μ	$-\frac{5}{2}$	$\times$	$\frac{5}{2}$	$\times$	$\frac{5}{2}$	$\frac{5}{4}$
λ	$M << 0$	0	$\frac{3}{4}$	1	2	$\frac{9}{4}$

Algorithm 2: Checking computationally whether $(P_N, P_B^i) \notin E_{\mathcal{N}}$ when N is an ellipse.

Initialization: Let P_B^i be the point whose edge (P_B^i, P_N) is going to check whether $(P_N, P_B^i) \notin E_{\mathcal{N}}$.

Set $points = \{\}$, $lambdas = \{\}$.

1 Compute the straight line $r(F^1, F^2)$.

2 Intersect $r(F^1, F^2)$ and the boundary of N, ∂N, in the points P_N^1 and P_N^2.

3 Include P_N^1 and P_N^2 in *points*.

4 Apply Algorithm 1.

4.2. Valid inequalities

This subsection is devoted to showing some results that adjust the big M constants that appear in the previous formulation, specifically, in constraints (α-C), where modelling of the sign requires computing the lower and upper bounds L and U, respectively. We are going to determine these bounds explicitly for the cases where the neighbourhoods are ellipses and segments.

Let $\overline{P^1_{B'}P^2_{B'}} = B' \in \mathcal{B}$ be a barrier, and $P_N \in N$. Let $\det(P_N|P^1_{B'}P^2_{B'})$ also be the determinant

whose value must be bounded. Clearly, the solution of the following problem gives a lower bound of the determinant:

$$\overline{L} = \min_{P_N=(x,y)\in N} F(x,y) := \det(P_N|P^1_{B'}P^2_{B'}) = \begin{vmatrix} P^1_{B'_x} - x & P^2_{B'_x} - x \\ P^1_{B'_y} - y & P^2_{B'_y} - y \end{vmatrix}. \qquad \text{(L-Problem)}$$

4.2.1. Lower and upper bounds when the neighbourhoods are line segments

In this case, the segment whose endpoints are P^1_N and P^2_N can be expressed as the following convex set:

$$N = \{(x,y) \in \mathbb{R}^2 : (x,y) = \mu P^1_N + (1-\mu)P^2_N,\ 0 \le \mu \le 1\}.$$

Since we optimise a linear function in a compact set, we can conclude that the objective function in (L-Problem) achieves its minimum and maximum at the extreme points of N, that is, in P^1_N and P^2_N.

4.2.2. Lower and upper bounds when the neighbourhoods are ellipses

The next case considered is that when N is an ellipse, that is, N is represented by the following inequality:

$$N = \{(x,y) \in \mathbb{R}^2 : ax^2 + by^2 + cxy + dx + ey + f \le 0\},$$

where a, b, c, d, e, f are coefficients of the ellipse. In extended form, we need to find:

$$\begin{aligned} \text{minimize} \quad & F(x,y) = \begin{vmatrix} P^1_{B'_x} - x & P^2_{B'_x} - x \\ P^1_{B'_y} - y & P^2_{B'_y} - y \end{vmatrix} = xP^1_{B'_y} - xP^2_{B'_y} + yP^2_{B'_x} - yP^1_{B'_x} + P^1_{B'_x}P^2_{B'_y} - P^1_{B'_y}P^2_{B'_x}, \\ \text{subject to} \quad & ax^2 + by^2 + cxy + dx + ey + f \le 0. \end{aligned} \qquad \text{(L-Ellipse)}$$

Since we minimise a linear function in a convex set, we can conclude that the extreme points are located in the frontier, so we can use the Lagrangian function to compute these points.

$$F(x,y;\lambda) = xP^1_{B'_y} - xP^2_{B'_y} + yP^2_{B'_x} - yP^1_{B'_x} + P^1_{B'_x}P^2_{B'_y} - P^1_{B'_y}P^2_{B'_x} + \lambda(ax^2 + by^2 + cxy + dx + ey + f).$$

$$\nabla F(x,y;\lambda) = 0 \iff \begin{cases} \frac{\partial F}{\partial x} &= P^1_{B'_y} - P^2_{B'_y} + 2ax\lambda + cy\lambda + d\lambda &= 0, \\ \frac{\partial F}{\partial y} &= P^2_{B'_x} - P^1_{B'_x} + 2by\lambda + cx\lambda + e\lambda &= 0, \\ \frac{\partial F}{\partial \lambda} &= ax^2 + by^2 + cxy + dx + ey + f &= 0. \end{cases}$$

From the first two equations, we obtain the following:

$$\lambda = \frac{P^2_{B'_y} - P^1_{B'_y}}{2ax + cy + d} = \frac{P^1_{B'_x} - P^2_{B'_x}}{2by + cx + e}.$$

From this equality, we obtain the following general equation of the straight line:

$$(P^2_{B'_y} - P^1_{B'_y})(2by + cx + e) - (P^1_{B'_x} - P^2_{B'_x})(2ax + cy + d) = 0,$$
$$\left[c(P^2_{B'_y} - P^1_{B'_y}) - 2a(P^1_{B'_x} - P^2_{B'_x})\right] x + \left[2b(P^2_{B'_y} - P^1_{B'_y}) - c(P^1_{B'_x} - P^2_{B'_x})\right] y + \left[e(P^2_{B'_y} - P^1_{B'_y}) - d(P^1_{B'_x} - P^2_{B'_x})\right] = 0,$$
$$\left[(2a, c) \cdot \overrightarrow{P^1_{B'} P^2_{B'}}\right] x + \left[(c, 2b) \cdot \overrightarrow{P^1_{B'} P^2_{B'}}\right] y + \left[(d, e) \cdot \overrightarrow{P^1_{B'} P^2_{B'}}\right] = 0,$$

where $\cdot$ denotes the scalar product of two vectors. Solving the quadratic system:

$$\begin{cases} \left[(2a, c) \cdot \overrightarrow{P^1_{B'} P^2_{B'}}\right] x + \left[(c, 2b) \cdot \overrightarrow{P^1_{B'} P^2_{B'}}\right] y + \left[(d, e) \cdot \overrightarrow{P^1_{B'} P^2_{B'}}\right] & = 0, \\ ax^2 + by^2 + cxy + dx + ey + f & = 0, \end{cases}$$

they arise two solutions $x^{\pm}$ and $y^{\pm}$ that are evaluated in the objective function to obtain the lowest and highest value, respectively, according to $L(P_N|P^1_{B'}P^2_{B'})$ and $U(P_N|P^1_{B'}P^2_{B'})$, respectively. The reader may note that the same approach can be adopted to obtain the bounds for the rest of the determinants that appear in (α-C).

4.2.3. Variable Fixing

In this subsection, the geometry of the problem is exploited to fix the variables. In particular, when a neighbourhood N is in the half-space generated by a barrier B, the sign of the determinant $\det(P_N|P^1_B P^2_B)$ does not change for any point $P_N \in N$. Therefore, a relevant number of variables α (hence β, γ, δ and ε) that model the sign of this determinant can be fixed 'a priori'. It is sufficient to check whether both bounds $L(P_N|P^1_{B'}P^2_{B'})$ and $U(P_N|P^1_{B'}P^2_{B'})$ computed in Subsection 4.2.2 have the same sign or not. Figure 15 shows an example where variables α can be fixed. Each of the blue neighbourhoods is completely contained in the half-spaces generated by the barrier, and α is fixed. However, the variable α corresponding to the orange neighbourhood depends on the half-space in which P_N is located.

5. Computational experiments

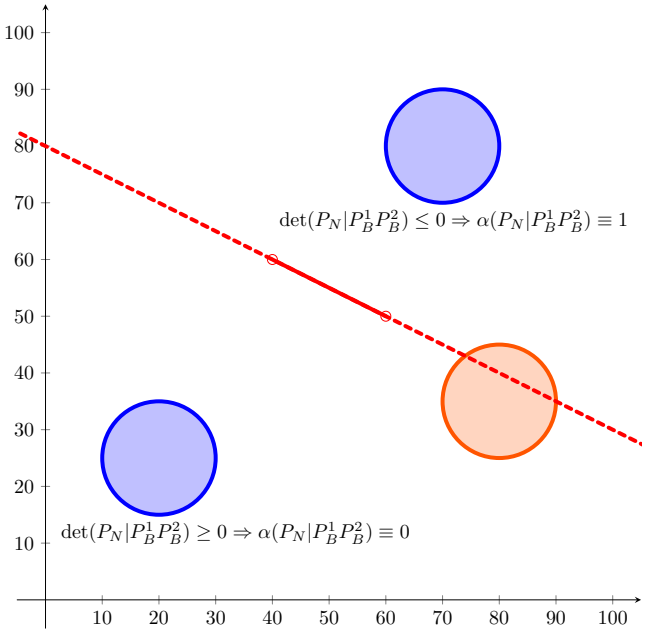
This section is devoted to studying the performance of (H-TSPHN) and (H-TSPN) formulations proposed in Section 3. In the first subsection, the procedure for generating the considered random instances is described. The second subsection details the experiments that have been conducted. The third subsection reports the results obtained in these experiments.

5.1. Data generation

To generate the instances of our experiments, we assume assumptions **A1-A4** stated in Section 2. The following proposition gives an upper bound for the number of balls that can be generated given an instance with $n = |\mathcal{B}|$ barriers:

Proposition 6. *Under assumptions **A1-A4**, the maximum number of balls that can be considered in* H-TSPHN *is* $O(n^2)$.

Figure 15: Fixing α variables when the whole neighborhood lies in one of the half-spaces generated by the barrier



Proof. Let $G_{\mathcal{B}} = (V_{\mathcal{B}}, E_{\mathcal{B}})$ the visibility graph of the barrier configuration in $\mathcal{B}$. The proof is based on the properties of this visibility graph described in Mitchell (2017), where it is proven that for a graph with n vertices $E_{\mathcal{B}} = O(n^2)$. It is clear that the maximum number of balls coincides with the number of faces, $f_{G_{\mathcal{B}}}$, of $G_{\mathcal{B}}$. Recall that the number of vertices in $\mathcal{B}$ is n. Next, by the Euler formula, the number of faces $f_{G_{\mathcal{B}}}$ is $2 + E_{G_{\mathcal{B}}} - n$. Since $E_{\mathcal{B}} = O(n^2)$, it follows that $f_{G_{\mathcal{B}}} = O(n^2)$. □

Algorithm 3 describes a general way to construct instances where neighbourhoods are balls.

Algorithm 3: General scheme of the instances generation

Initialization: Let $|\mathcal{N}|$ be the number of neighbourhoods to generate.

Let $R = [LB_x, UB_x] \times [LB_y, UB_y] \subseteq \mathbb{R}^2$ be the rectangle where centers will be generated.

Set $points = \{\}$; $\mathcal{B} = \{\}$; $\mathcal{N} = \{\}$.

1 Generate $|\mathcal{N}|$ points uniformly distributed in R and include them in $points$.

2 Generate pairwise disjoint barriers that separate the points and include them in $\mathcal{B}$.

3 Generate neighbourhoods around $points$ and include them in $\mathcal{N}$.

The two following subsections develop Steps 2 and 3 of the Algorithm 3, respectively. Specifically, the Algorithm 4, related with Step 2, details how barriers are generated assuming **A1-A4**, while the Algorithm 5, related with Step 3, describes the way neighbourhoods are designed. In practice, Algorithms 3, 4 and 5 are implemented to be run sequentially and generate the battery of instances used for testing the formulations.

Barriers generation

In this subsection, we focus on how to generate line segments located in general position without crossings. The idea is to build bisectors that separate each pair of points in the set $points$. The initial length of each bisector is r_{init}. This length is reduced until it does not intersect any of the previously

generated line segments. The Algorithm 4 reports the pseudocode to generate barriers.

Algorithm 4: Generation of the barriers

Initialization: Let *points* be the set already randomly uniformly generated in R.

Let r_{init} be one half of the initial length of the barriers.

Set $\mathcal{B} = \{\}$.

for $P, P' \in$ *points* **do**
  if $\overline{PP'} \cap B = \emptyset$, $\forall B \in \mathcal{B}$ **then**
    Compute $\overrightarrow{d} = \overrightarrow{PP'}$.
    Compute $M = P + \frac{1}{2}\overrightarrow{d}$.
    Compute the unitary vector $\overrightarrow{n_u}$ perpendicular to $\overrightarrow{d}$.
    Set $r = r_{\text{init}}$.
    Generate the barrier $B(r) = \overline{P_B^+ P_B^-}$ where $P_B^{\pm} = M \pm r\overrightarrow{n_u}$.
    while $B(r) \cap B' \neq \emptyset$ *for some* $B' \in \mathcal{B}$ **do**
      Set $r := r/2$.
      Generate the barrier $B(r)$.
    Include $B(r)$ in $\mathcal{B}$.

Neighbourhood generation

Once the set of points and barriers are generated, the neighbourhoods are created using two different sizes:

- **Randomly-sized neighbourhoods:** that do not intersect barriers and verify **A4**.
- **Fixed-sized neighbourhoods:** that can cross barriers and are not required to assume **A4**.

The first case is used to study the performance of the models H-TSPHN and H-TSPN proposed in the article when the neighbourhoods are circles or line segments. The following procedure describes the pseudocode to create circles.

Algorithm 5: Generation of randomly-sized circles

Initialization: Let *points* be the set randomly, uniformly generated in R.

Let $\mathcal{B}$ be the barriers already generated.

Set $\mathcal{N} = \{\}$.

for $P \in$ *points* **do**
  Set $r_{\max} = \min_{\{P_B \in B : B \in \mathcal{B}\}} d(P, P_B)$.
  Generate a random *radii* uniformly distributed in the interval $\left[\frac{1}{2}r_{\max}, r_{\max}\right]$.
  Set the ball N whose centre is P and radii is *radii*.
  Include N in $\mathcal{N}$.

The line segments instances are generated by randomly selecting two diametrically opposite points in the boundary of the balls instances obtained with Algorithm 5.

The second case focuses on the effectiveness of the exact model for the H-TSPN in terms of *overlapping ratio* of the circles. This ratio, introduced in Mennell (2009), is calculated by dividing the average radius

of the neighbourhood sets by the length of the longest side of R. Mennell (2009) shows that the higher the overlaping ratio, the higher the difficulty of the instance. Algorithm 5 is slightly modified by setting a fixed value for *radii*, based on instances considered in Behdani and Smith (2014).

The data of all the instances used in all our computational experiments can be downloaded from Puerto and Valverde (2023).

5.2. Configuration of the experiments

In this work, three experiments are designed to study the behaviour of the models H-TSPHN and H-TSPN.

In the first experiment, we construct five instances for each number of randomly-sized neighbourhoods in $|\mathcal{N}| \in \{5, 10, 20, 30, 50, 60, 65, 70, 75, 80, 100\}$ within $R = [0, 100] \times [0, 100]$. These neighbourhoods are balls and line segments that have been created using the Algorithm 5. In addition, barriers are generated according to Algorithm 4. $|\boldsymbol{\mathcal{B}}|$ reports the average number of barriers generated for each experiment. For each instance, we run the models with and without strengthening the formulations.

In the second experiment, we generate ten instances with a number $|\mathcal{N}| = 10$ of fixed circles of *radii* equals to 0.5. The centres are drawn in $R = [0, 0] \times [16, 10]$. We run H-TSPN considering removing edges, variable fixing and big M estimation in a separate way to know the marginal contribution of each of these strengthening methods to improve the behaviour of the solver.

In the third experiment, based on Behdani and Smith (2014), we generate ten instances with a number $|\mathcal{N}| \in \{6, 8, 10, 12, 14, 16, 18, 20\}$ of fixed circles. The centres are drawn in $R = [0, 0] \times [16, 10]$. The fixed *radii* considered are 0.25, 0.5, and 1. Therefore, the overlap ratios are 0.015625, 0.03125 and 0.0625, respectively. We run H-TSPN with strengthening to see the performance of our methods when neighbourhoods overlap.

Formulations are coded in Python 3.9.2 (Van Rossum and Drake (2009)) and solved in Gurobi 9.1.2 (Gurobi Optimization LLC (2022)) on an AMD® Epyc 7402p 8-core processor.

The values obtained by the solver that are reported in our tables are:

- $\boldsymbol{\#Found}$: number of instances in which the solver finds a feasible solution.
- $\boldsymbol{Gap}$: gap between the best incumbent solution with respect to the best bound found by the solver. It is computed as $Gap = (upper\,bound - lower\,bound)/lower\,bound$.
- $\boldsymbol{Time_{model}}$: time (in seconds) spent by the solver to obtain the best solution found.
- $\boldsymbol{Time_{marginal}}$: time (in seconds) spent by Python to set up the model, including only a strengthening subprocess.
- $\boldsymbol{Time_{prepro}}$: time (in seconds) spent by Python to set up the model, including all the strengthening process.
- $\boldsymbol{Time_{total}}$: overall time (in seconds) to solve the model.

For all the experiments, a time limit of 1 hour of solver time was set in the branch-and-bound procedure.

5.3. Results of the experiments

We report the results of the first experiment in Table 2. The layout is organised in 3 blocks of columns. The first block describes the parameters of the problem: number of neighbourhoods $|\mathcal{N}|$, if **A4** is assumed or not (H-TSPHN vs H-TSPN), if strengthening is considered, and the number of barriers. The second and third blocks describe the average results obtained with the solver for circles and segments, respectively.

Table 2: Computational results obtained with (H-TSPHN) and (H-TSPN)

				Circles					*Segments*				
$\|\mathcal{N}\|$	A_4	*Strengthening*	$\|\mathcal{B}\|$	*#Found*	*Gap*	$Time_{model}$	$Time_{prepro}$	$Time_{total}$	*#Found*	*Gap*	$Time_{model}$	$Time_{prepro}$	$Time_{total}$
5	no	no	4.8	5	0	157.24	0.29	157.53	5	0	42.59	0.91	43.5
		yes	4.8	5	0	1.24	0.44	1.68	5	0	0.42	1.47	1.89
	yes	no	10.4	5	0.1	819.98	1.3	821.28	5	0	22.03	1.58	23.61
		yes	10.4	5	0	0.61	1.16	1.77	5	0	0.38	1.57	1.95
10	no	no	9.2	5	0.17	2193.53	2.09	2195.62	5	0.42	2884.22	6.6	2890.82
		yes	9.2	5	0	9.1	2.58	11.68	5	0	1.69	9.61	11.3
	yes	no	19.2	5	0.17	933.67	9.59	943.26	5	0.3	1448.73	11.8	1460.53
		yes	19.2	5	0	2.56	6.36	8.92	5	0	1.53	9.24	10.77
20	no	no	17.6	5	0.17	3600	16.15	3616.15	5	0.2	3600	50.93	3650.93
		yes	17.6	5	0	68.67	17.43	86.1	5	0	11.66	74.89	86.55
	yes	no	35.8	5	0.21	2349.26	87.7	2436.96	5	0	145.06	111.34	256.4
		yes	35.8	5	0	42.45	43.49	85.94	5	0	8.05	64.01	72.06
30	no	no	28	4	0.5	3600	75.15	3675.15	5	0.4	3600	201.05	3801.05
		yes	28	5	0	2034.53	65.23	2099.76	5	0	54.35	246.96	301.31
	yes	no	56.4	5	0.23	3027.02	512.03	3539.05	5	0.18	1039.08	672.46	1711.54
		yes	56.4	5	0	147.98	179.95	327.93	5	0	96.72	270.56	367.28
50	no	no	44.2	3	0.87	3600	364.75	3964.75	4	0.3	3600	960.53	4560.53
		yes	44.2	3	0	2485.76	297.99	2783.75	5	0	311.49	1043.15	1354.64
	yes	no	89	5	0.37	3600	4650.3	8250.3	5	0	1445.39	4654.95	6100.34
		yes	89	5	0.1	3600	1213.85	4813.85	5	0	353.17	1292.12	1645.29
60	no	no	50.2	0	-	-	-	-	5	0.53	3600	1213.08	4813.08
		yes	50.2	3	0.22	3600	538.11	4138.11	5	0	1384.92	1103.43	2488.35
	yes	no	100.8	5	0.15	3600	8688.65	12288.65	5	0	1671.85	8726.63	10398.48
		yes	100.8	5	0.13	3600	2179.26	5779.26	5	0.01	2903.27	2339.58	5242.85
65	no	no	52.8	0	-	-	-	-	4	0.75	3600	1561.46	5161.46
		yes	52.8	2	0.35	3600	671.07	4271.07	5	0.02	3249.12	1343.12	4592.24
	yes	no	106	5	0.13	3600	11250.62	14850.62	5	0	1381.66	11269.25	12650.91
		yes	106	5	0.17	3600	2843.48	6443.48	5	0.01	2877.66	3003.47	5881.13
70	no	no	57.6	0	-	-	-	-	4	0.85	3600	1977.3	5577.3
		yes	57.6	3	0.12	3600	898.99	4498.99	5	0.04	3211.2	1754.51	4965.71
	yes	no	115.6	5	0.29	3600	17366.28	20966.28	5	0.04	2853.58	17433.28	20286.86
		yes	115.6	5	0.32	3600	4106.95	7706.95	5	0.02	3203.33	4311.14	7514.47
75	no	no	63.2	0	-	-	-	-	3	0.74	3600	2976.6	6576.6
		yes	63.2	0	-	-	-	-	5	0.24	3283.37	242407	5707.44
	yes	no	126.8	4	0.24	3600	26363.48	29963.48	5	0.01	1903.99	26153.87	28057.86
		yes	126.8	3	0.23	3600	5382.14	8982.14	5	0.02	2458.75	6140.02	8598.77
80	no	no	64	0	-	-	-	-	1	0.83	3600	4205.41	7805.41
		yes	64	0	-	-	-	-	5	0	1775.16	2858.51	4633.67
	yes	no	128.6	0	-	-	-	-	5	0.11	3471.06	29073.69	32544.75
		yes	128.6	0	-	-	-	-	5	0	2701.21	7031.87	9733.08
100	no	no	81.6	0	-	-	-	-	0	-	-	-	-
		yes	81.6	0	-	-	-	-	4	0	1761.08	6620.09	8381.17
	yes	no	163.2	0	-	-	-	-	5	0.48	3600	91153.55	94753.55
		yes	163.2	0	-	-	-	-	5	0	2720.76	19429.1	22149.86

Analysing the results in Table 2, we first observe that solving the problem considering balls as neighbourhoods is harder than solving with segments. Next, we also observe that this approach solves to optimality all instances for circles up to $|\mathcal{N}| = 30$ with strengthening for the H-TSPHN problem. However, if we do not strengthen the formulation, the solver always reports gap for all the instances with circles. The same behaviour can be seen for the H-TSPN up to $|\mathcal{N}| = 30$. Nevertheless, for sizes $|\mathcal{N}|$ in

50-75, both formulations start to differ in the number of instances in which the solver can find a feasible solution. For the H-TSPHN, the reader may observe that strengthening does not improve the gap within the time limit. However, in the H-TSPN, strengthening does increase the number of instances in which the solver finds a solution and the fraction of gaps certified after the execution time. Anyway, whenever a feasible solution is found, the maximum average gap that is reported with strengthening is 0.35. Finally, for sizes 80 and 100, the solver cannot find any solution for any of the models, regardless of whether strengthening is considered or not. In terms of execution time, we can conclude that strengthening always improves both the time to obtain the optimal solution ($\boldsymbol{Time_{model}}$) and the time the computer takes to load all the variables and constraints of the model ($\boldsymbol{Time_{prepro}}$). This difference is more appreciable in the H-TSPHN, as the reader can notice in the aggregation of these two columns in ($\boldsymbol{Time_{total}}$). This fact can be explained in terms of the number of barriers: the higher the number of barriers, the larger the number of variables that can be fixed beforehand. On the other hand, we can observe that Gurobi can report the optimal solution for almost all segment instances generated by solving the strengthened versions of the H-TSPHN and H-TSPN. In addition, the time spent to solve all these instances is lower than the time limit. However, the time to load and strengthen the model is very similar to that for circle instances.

In Figures 16 and 17, the reader can compare, at a glance, the time that the solver spent to obtain the best solution found and the gap between this solution and the best bound found by the solver. We can conclude that strengthening improves significantly the time spent by Gurobi to get the optimal solution and instances with circles are harder to be solved than those with segments, in terms of time and final gap.

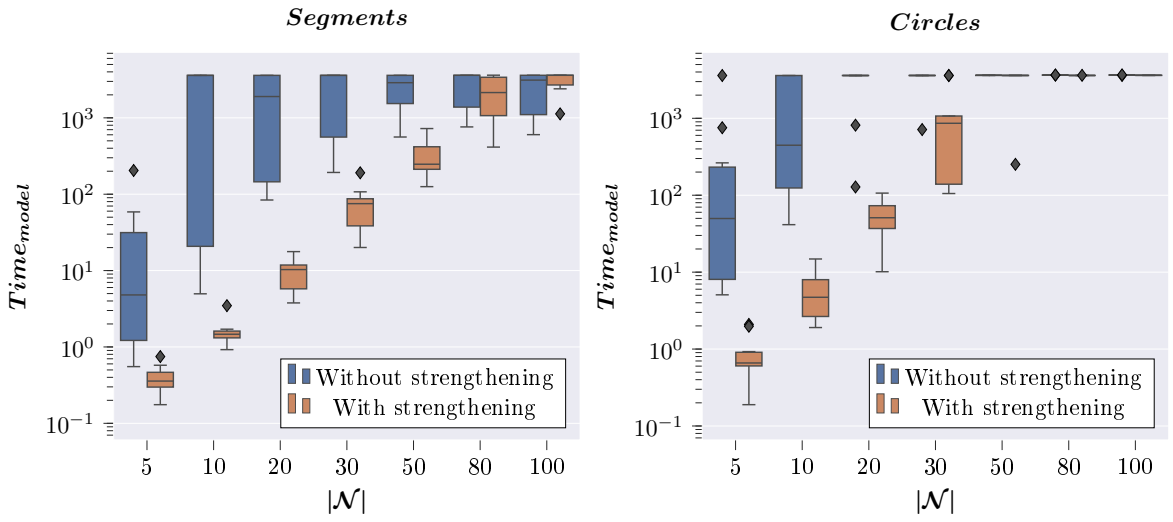


Figure 16: Runtime of the model (H-TSPN) without and with strengthening when the neighborhoods are segments and balls.

Once we have validated in our first set of experiments that the proposed strengthening techniques are useful, we want to test their individual contribution to that improvement. For this reason, we have designed a second experiment to assess the usefulness of each individual strengthening: edge removing, variable fixing and Big-M estimation. In this case, we have considered a representative set of instances

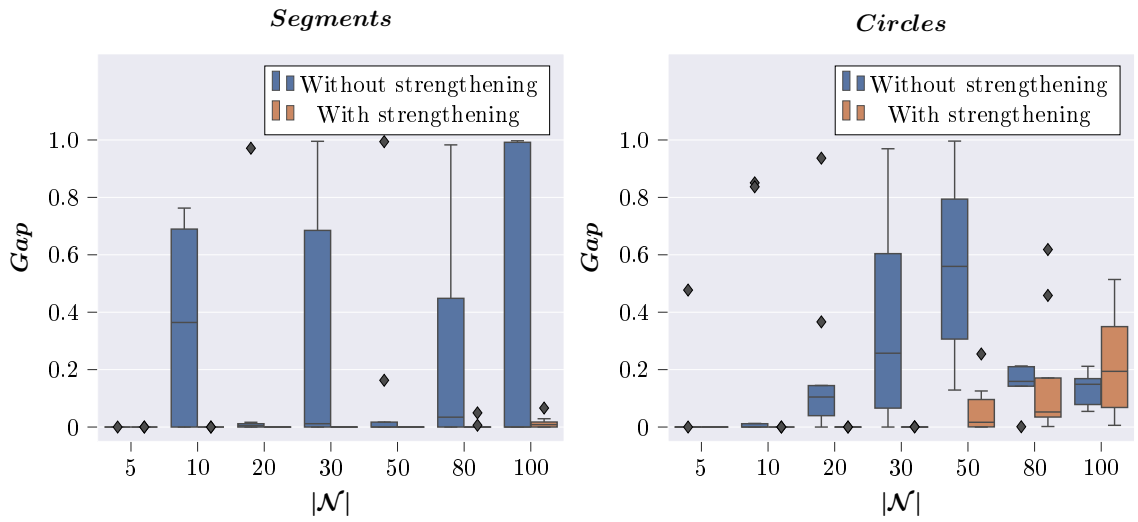


Figure 17: Gap of the model (H-TSPN) without and with strengthening when the neighborhoods are segments and balls.

with $|\mathcal{N}| = 10$ and an average number of barriers $|\mathcal{B}| = 18$ and we have compared the performance of the plain formulation without any improvement and with each one of the strengthenings at a time.

Table 3: Computational results for assessing the usefulness of different strenghthenings

$\|\mathcal{N}\|$	$\|\mathcal{B}\|$	*Edge removing*	*Variable fixing*	*Big M estimation*	*#Found*	*Gap*			$Time_{model}$			$Time_{marginal}$		
						mean	min	max	mean	min	max	mean	min	max
10	18	yes	no	no	10	0.29	0	0.87	2643.25	46.93	3600	3.05	1.56	3.12
		no	yes	no	10	0.41	0	1	2920.99	50.99	3600	5.30	3.41	7.71
		no	no	yes	10	0.08	0	0.32	2863.76	227.67	3600	5.30	3.41	7.71
		no	no	no	10	0.34	0.08	0.67	3600	3600	3600	2.25	1.45	3.02

In Table 3, the reader may observe that each one of the improvements separately speeds the solver up to find the optimal solution in less computing time, for those instances in which the best solution is obtained within the time limit. In addition, if none of the three preprocessings is considered, the solver does not find optimal solutions for any of the instances and the best gap for the any of the instances, in this case, is 0.08. Analyzing the qualitative effect of the variable fixing, one can observe that, at times, its application worsen the gaps with respect to the plain model. This can be due to the overlapping with the preprocessing techniques that the solver applies internally to simplify the model before executing its branch and bound algorithms. In spite of that, from the results reported in Table 3, we observe that applying each strengthening separately always decreases the computing time and improves the average gap for most of the instances. Therefore, we can conclude that the combination of these three enhancements improves the solver performance. This supports the combination of all of them in our computational experience.

In Table 4, we detail the values obtained with Gurobi for the third experiment. The first block reports the size of the instances and the average number of barriers for each size. The other three blocks describe the features reported by the solver to compare the efficiency of the model for *radii* equals to 0.25, 0.5 and 1, respectively. For the smallest *radii*, all the instances are almost solved to optimality reporting a maximum average gap of 0.03 for $|\mathcal{N}| = 20$. In the instances with $radii = 0.5$, the number of feasible solutions found starts to decrease from size 16. In terms of the gap, from $|\mathcal{N}| = 14$, the solver is

unable to find the optimal solution because it reaches the time limit. In any case, the maximum average gap for these instances is 0.32 for the largest size. Finally, for $radii = 1$, only instances with $|\mathcal{N}| = 6$ are solved to optimality. For sizes 10, 12 and 14, the solver only finds a feasible solution for one half of the instances, approximately. However, it cannot find any solution for $|\mathcal{N}| \geq 16$. The results obtained for these instances lead us to conclude that the larger the radii of the neighbourhood, the higher the complexity to be solved. This conclusion is in line with the existing trend in the literature, as exposed in Puerto and Valverde (2022) or Blanco et al. (2017).

Table 4: Computational results obtained with (H-TSPN) for Smith instances

		$Radii = 0.25$					$Radii = 0.5$					$Radii = 1$				
$\|\mathcal{N}\|$	$\|\mathcal{B}\|$	$\#Found$	Gap	$Time_{model}$	$Time_{prepro}$	$Time_{total}$	$\#Found$	Gap	$Time_{model}$	$Time_{prepro}$	$Time_{total}$	$\#Found$	Gap	$Time_{model}$	$Time_{prepro}$	$Time_{total}$
6	8.6	10	0	2.58	1.21	3.79	10	0	16.31	1.26	17.57	10	0.03	1512.68	1.4	1514.08
8	12.7	10	0	14.12	3.07	17.19	10	0	171.44	3.16	174.6	10	0.27	3600	3.56	3603.56
10	18	10	0	199.23	7.35	206.58	10	0.03	1453.59	7.55	1461.14	5	0.35	3600	8.5	3608.5
12	19.7	10	0	149.49	10.12	159.61	10	0.09	3454.33	10.54	3464.87	6	0.46	3600	12.46	3612.46
14	24.5	10	0	374.8	23.53	398.33	10	0.11	3600	24.23	3624.23	4	0.65	3600	26.56	3626.56
16	29.4	10	0.01	2486.62	40.2	2526.82	8	0.16	3600	41.1	3641.1	0	-	-	-	-
18	32	10	0.03	3053.15	52.18	3105.33	7	0.21	3600	53.05	3653.05	0	-	-	-	-
20	37	10	0.03	3040.06	75.25	3115.31	6	0.32	3600	80.03	3680.03	0	-	-	-	-

6. Concluding Remarks

This paper has dealt with two problems, the H-SPPN and the H-TSPN. In both cases, we have assumed that barriers do not allow direct movements between neighbourhoods **A4**. The more general case that does not assume **A4** gives rise to nonconvex mixed-integer problems. It is still an open problem whether there is some kind of finite dominating set with polynomial cardinality for the version of the H-TSPN which could help simplify the formulation of the problem. These questions are very interesting but beyond the scope of this paper. Needless to say, we plan to continue its analysis in a follow-up paper.

It would also be interesting to combine in the same model different typologies of barriers such as polygonals and second-order cone-representable sets. It is also interesting to consider the single- or multiple facility problem with barriers and neighbourhoods.

All the above-mentioned problems are natural extensions of the ones considered in this paper and will deserve our attention in the future.

References

Arkin, E. M. and Hassin, R. (1994). Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197–218.

Behdani, B. and Smith, J. C. (2014). An Integer-Programming-Based Approach to the Close-Enough Traveling Salesman Problem. *INFORMS Journal on Computing*, 26(3):415–432.

Blanco, V., Fernández, E., and Puerto, J. (2017). Minimum Spanning Trees with neighborhoods: Mathematical programming formulations and solution methods. *European Journal of Operational Research*, 262(3):863–878.

Boissonnat, J.-D. and Snoeyink, J. (2000). Efficient algorithms for line and curve segment intersection using restricted predicates. *Computational Geometry*, 16(1):35–52.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, Cambridge.

Carrabs, F., Cerrone, C., Cerulli, R., and Golden, B. (2020). An Adaptive Heuristic Approach to Compute Upper and Lower Bounds for the Close-Enough Traveling Salesman Problem. *INFORMS Journal on Computing*, 32(4):1030–1048.

Coutinho, W. P., do Nascimento, R. Q., Pessoa, A. A., and Subramanian, A. (2016). A Branch-and-Bound Algorithm for the Close-Enough Traveling Salesman Problem. *INFORMS Journal on Computing*, 28(4):752–765.

D. Perez, E. J. Powley, D. Whitehouse, P. Rohlfshagen, S. Samothrakis, P. I. Cowling, and S. M. Lucas (2014). Solving the Physical Traveling Salesman Problem: Tree Search and Macro Actions. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(1):31–45.

Gentilini, I., Margot, F., and Shimada, K. (2013). The travelling salesman problem with neighbourhoods: MINLP solution. *Optimization Methods and Software*, 28(2):364–378.

Glock, K. and Meyer, A. (2023). Spatial coverage in routing and path planning problems. *European Journal of Operational Research*, 305(1):1–20.

Gurobi Optimization LLC (2022). Gurobi Optimizer Reference Manual.

J. . -P. Laumond, P. E. Jacobs, M. Taix, and R. M. Murray (1994). A motion planner for nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 10(5):577–593.

Klamroth, K. (2002). *Single-Facility Location Problems with Barriers*. Springer, New York, NY.

Letchford, A. N., Nasiri, S. D., and Theis, D. O. (2013). Compact formulations of the Steiner Traveling Salesman Problem and related problems. *European Journal of Operational Research*, 228(1):83–92.

Lobo, M. S., Vandenberghe, L., Boyd, S., and Lebret, H. (1998). Applications of second-order cone programming. *International Linear Algebra Society (ILAS) Symposium on Fast Algorithms for Control, Signals and Image Processing*, 284(1):193–228.

Mennell, W. K. (2009). *Heuristics for Solving Three Routing Problems: Close-enough Traveling Salesman Problem, Close-Enough Vehicle Routing Problem, and Sequence-Dependent Team Orienteering Problem*. PhD thesis, University of Maryland, College Park, United States – Maryland.

Mitchell, J. S. B. (2017). Shortest Paths and Networks. In *Handbook of Discrete and Computational Geometry*. Chapman and Hall/CRC, 3 edition.

Nesterov, Y. and Nemirovski, A. (1994). *Interior-Point Polynomial Algorithms in Convex Programming*. Studies in Applied and Numerical Mathematics. Society for Industrial and Applied Mathematics.

O'Rourke, J. (2017). Chapter 33: Visibility. In *Handbook of Discrete and Computational Geometry*. Chapman and Hall/CRC, 3 edition.

Papadimitriou, C. H. and Steiglitz, K. (1977). On the Complexity of Local Search for the Traveling Salesman Problem. *SIAM Journal on Computing*, 6(1):76–83.

Puerto, J. and Valverde, C. (2022). Routing for unmanned aerial vehicles: Touring dimensional sets. *European Journal of Operational Research*, 298(1):118–136.

Puerto, J. and Valverde, C. (2023). Instances for the Hampered Travelling Salesman Problem with Neighbourhoods.

T. Moemke and O. Svensson (22). Approximating Graphic TSP by Matchings. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 560–569.

Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Y. K. Hwang and N. Ahuja (1992). A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*, 8(1):23–32.

Yuan, B. and Zhang, T. (2017). Towards Solving TSPN with Arbitrary Neighborhoods: A Hybrid Solution. In Wagner, M., Li, X., and Hendtlass, T., editors, *Artificial Life and Computational Intelligence*, pages 204–215, Cham. Springer International Publishing.

Chapter 8

The Hampered K-Median Problem with Neighbourhoods

The Hampered K-Median Problem with Neighbourhoods

Justo Puerto[a,*], Carlos Valverde[b,*]

[a] *Institute of Mathematics (IMUS) and Department of Statistics and Operations Research, University of Seville, Seville, 41012, Spain*

[b] *Institute of Mathematics (IMUS) and Department of Statistics and Operations Research, University of Seville, Seville, 41012, Spain*

Abstract

This paper deals with facility location problems in a continuous space with neighbours and barriers. Each one of these two elements, neighbours and barriers, make the problems harder than their standard counterparts. Therefore, mixing both together results in a new challenging problem that, as far as we know, has not been addressed before but that has applications for inspection and surveillance activities and the delivery industry assuming uniformly distributed demand in some regions. Specifically, we analyze the K-Median problem with neighbours and polygonal barriers under two different situations. As a first building block, we deal with the problem assuming that neighbourhoods are not visible from one another and therefore there are no rectilinear paths joining any two of them without crossing barriers. Under this hypothesis we derive a valid mixed integer, linear formulation. Removing that hypothesis leads to the more general, realistic problem but at the price of making it more challenging. Adapting the elements of the first formulation, we also develop another valid mixed integer, bilinear formulation. Both formulations handle polygonal barriers and neighbours that are second-order cone (SOC) representable, that we preprocess and strengthen with valid inequalities. These mathematical programming formulations are also instrumental to derive an adapted matheuristic algorithm that provides good quality solutions for both problems in short computing time. The paper also reports an extensive computational experience showing that our exact and heuristic approaches are useful: the exact approach can solve to optimality instances with up to 50 neighbourhoods and different number of barriers within one hour of CPU time, whereas the matheuristic always returns good feasible solutions in less than 100 seconds.

Keywords: Facility location, Continuous Location, Barriers, Mixed integer Conic programming

1. Introduction

Location analysis is a classical branch of operations research that studies the best way to place some facilities to satisfy the demand of customers. In location analysis, problems are usually classified in discrete or continuous facility location problems. The first class is considered when there is a finite number of candidates to allocate facilities (see Ulukan and Demircioğlu (2015) for a survey). Continuous facility location problems arise if facilities can be placed anywhere in some continuous regions. Both versions are widely investigated in the literature (see Drezner and Hamacher (2004) or Nickel and Puerto

*Equally contributing authors

Email addresses: puerto@us.es (Justo Puerto), cvalverde@us.es (Carlos Valverde)

(2007) for more details) by their many applications in transportation, logistics or telecommunication. For these problems, lot of variants have been studied in terms of the objective functions to be optimized, the number of facilities that must be allocated or the maximum capacity that facilities can supply, among many other respects (we refer the reader to Kuehn and Hamburger (1963) and Puerto (2008)).

In Blanco (2019), the Ordered k-Median Problem with Neighbourhoods is presented as a single source uncapacitated continuous facility location problem that extends its respective underlying discrete location problem. In this problem, facilities are allowed to be allocated in certain regions called neighbourhoods. If those are points, the problem reduces to the single source uncapacitated facility discrete location problem, that have been already studied in the literature. Otherwise, the continuous version is considered. In this version, different shapes and sizes for the neighbourhoods allow one to model how imprecise the provided locational information is. This problem also has interest on drone delivery and inspection problems. Neighbourhoods can represent regions that the drone must reach and where the customers are willing to pick up the orders (they can be seen as uniform probability densities) in the delivery industry. Moreover, they can be also used for modelling some areas that must be inspected by the drone (whenever visiting a point of these areas is enough to consider them as inspected). This framework will be called Facility Location with Neighbourhoods, a terminology borrowed from the neighbourhood versions of the Minimum Spanning Tree problem, described in Blanco et al. (2017) and the Traveling Salesman problem, studied in Gentilini et al. (2013), Yuan and Zhang (2017) or Puerto and Valverde (2022).

This paper extends the k-Median Problem with Neighbourhoods by including a set of barriers, represented by line segments, that the trips between demand points and service facilities cannot cross. These barriers simulate buildings in urban areas that vehicles (drones) cannot cross. The resulting problem keeps geometric components from the p-median problem with neighbourhoods that must be exploited to partially overcome the difficulties of the solution approaches and algorithms in the network design among neighbourhoods with barriers. The use of barriers in location problems has been studied (see Klamroth (2002)). However, the combination of both elements has attracted less attention in the Operations Research literature.

Our goal in this paper is to deal with the k-median problem with neighbourhoods and barriers that we call the Hampered k-Median problem with Neighbourhoods (H-KMPN). We present exact mathematical programming formulations assuming linear barriers and second-order cone (SOC) representable neighbourhoods. These formulations are modeled by using a geodesic shortest-path representation, based on problems studied in Mitchell (2017). These assumptions lead to quadratically-constrained mixed-integer formulations. Solving this family of formulations in addition to the classic k-median, that is NP-hard, makes the solution of the problem under study a hard challenge. On-the-shelf solvers can deal only with small-size instances. This fact motivates the design of a matheuristic that provides good quality solutions for medium-size instances.

The paper is organized in six sections. In Section 2 the problem and its variant are introduced and described. Section 3 is devoted to provide quadratically-constrained mixed-integer programming formulations of the problems. In Section 4 the matheuristic approach is described. The results of some computational experiments are reported in Section 5. Finally, some conclusions are presented in Section

6.

2. Description of the Problem

In this section, the framework of the two versions of the problem considered in the manuscript are analyzed: the Hampered k-Median Problem with Hidden Neighbourhoods H-KMPHN and the Hampered k-Median Problem with Neighbourhoods H-KMPN. Since we have in mind their applications to the drone delivery problem with uniformly distributed demand in regions and inspection problems, at times, we will refer to the moving object as the *drone*.

First of all, we state the sets that describe the main elements of the problems. Second, we set the assumptions that barriers must verify. Finally, the goal and the sets of parameters used in the following are defined in order to give valid formulations for these problems.

2.1. Parameters and Assumptions of the Problem

The sets describing both versions of the problem are:

- $\mathcal{S}$: Set of neighbourhoods describing the possible sources where a facility can be allocated. It is assumed, wlog, that one facility can be allocated to each source at most once.
- $\mathcal{T}$: Set of neighbourhoods representing the targets that must be served by a facility. It is assumed, wlog, that each target is served when it has been assigned to a facility.
- $\mathcal{B}$: Set of barriers (line segments) that can not be crossed when a facility is joined with a target. The assumptions made for this set of line segments are the following:

  A1 The line segments of $\mathcal{B}$ are located in general position, i.e., the endpoints of these segments are not aligned. Although it is possible to model the most general case, one can always slightly modify one of the endpoints so that the segments are in general position.

  A2 The line segments of $\mathcal{B}$ are open sets, that is, it is possible that the drone visits endpoints of segments, but entering in its interior is not allowed. Observe that without loss of generality, we can always slightly enlarge these segments to make them open.

  A3 If there are two overlapping barriers, we assume that there is only one barrier given by the union of them.

  A4 There is no rectilinear path joining a pair of source-target neighbourhoods without crossing an obstacle.

The H-KMPN is the relaxed version of the H-KMPHN without imposing assumption **A4**. In this case, it is not required that the barriers separate neighbourhoods completely, i.e., when moving from one neighbourhood to another one it is possible to go following a straight line without crossing any barrier. Figure 1 shows an example of each version of the problem that is being considered. The left picture shows an instance of the H-KMPHN, where green neighbourhoods represent possible sources to allocate the facilities, blue neighbourhoods represent targets to be assigned to the sources and the red

line segments show the barriers that the drone cannot cross. The right picture illustrates an instance of the H-KMPN where some sources and targets can be joined by a rectilinear path.

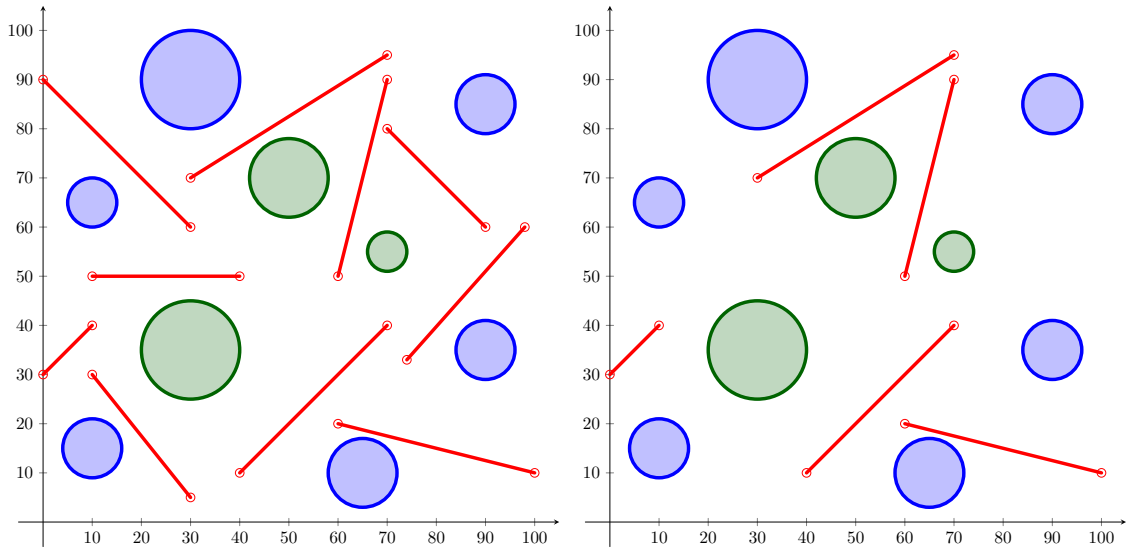


Figure 1: Problem data of the H-KMPHN and H-KMPN

2.2. Description of the Hampered k-Median Problem with Neighbourhoods

The goal of the H-KMPHN is to find a subset of k points in the source set $\mathcal{S}$, at most one in each neighbourhood, and one point in each target set $\mathcal{T}$ that minimize the weighted length of the path joining each target point with its associated source point and the weighted link distance without crossing any barrier of $\mathcal{B}$ assuming **A1-A4**. Recall that the link distance accounts for the numbers of edges of the path joining two points in the underlying graph. The interested reader is referred to de Berg et al. (1990); Daescu et al. (2008) for further details. To state the model, we define the following sets:

- $V_{\mathcal{S}} = \{P_S : S \in \mathcal{S}\}$. Set of the points selected in the sources of $\mathcal{S}$.
- $V_{\mathcal{B}} = \{P_B^1, P_B^2 : B = \overline{P_B^1 P_B^2} \in \mathcal{B}\}$. Set of vertices that come from the endpoints of barriers in the problem.
- $V_{\mathcal{T}} = \{P_T : T \in \mathcal{T}\}$. Set of the points selected in the targets of $\mathcal{T}$.
- $E_{\mathcal{S}} = \{(P_S, P_B^i) : P_S \in V_{\mathcal{S}}, P_B^i \in V_{\mathcal{B}} \text{ and } \overline{P_S P_B^i} \cap B'' = \emptyset, \forall B'' \in \mathcal{B},\ i = 1, 2\}$. Set of edges formed by the line segments that join the point selected in any source neighbourhood $S \in \mathcal{S}$ and every endpoint in the barriers that do not cross any other barrier in $\mathcal{B}$.
- $E_{\mathcal{B}} = \{(P_B^i, P_{B'}^j) : P_B^i, P_{B'}^j \in V_{\mathcal{B}} \text{ and } \overline{P_B^i P_{B'}^j} \cap B'' = \emptyset,\ \forall B'' \in \mathcal{B},\ i, j = 1, 2\}$. Set of edges formed by the line segments that join two vertices of $V_{\mathcal{B}}$ and do not cross any other barrier in $\mathcal{B}$.
- $E_{\mathcal{T}} = \{(P_B^i, P_T) : P_B^i \in V_{\mathcal{B}}, P_T \in V_{\mathcal{T}} \text{ and } \overline{P_B^i P_T} \cap B'' = \emptyset, \forall B'' \in \mathcal{B},\ i = 1, 2\}$. Set of edges formed by the line segments that join the point selected in any target neighbourhood $T \in \mathcal{T}$ and every endpoint in the barriers that do not cross any other barrier in $\mathcal{B}$.

The above sets allow us to define the graph $G_{\text{KMPHN}} = (V_{\text{KMPHN}}, E_{\text{KMPHN}})$ induced by the barriers and neighbourhoods, where $V_{\text{KMPHN}} = V_{\mathcal{S}} \cup V_{\mathcal{B}} \cup V_{\mathcal{T}}$ and $E_{\text{KMPHN}} = E_{\mathcal{S}} \cup E_{\mathcal{B}} \cup E_{\mathcal{T}}$.

By taking the same approach, the graph induced for the relaxed version H-KMPN can be described as $G_{\text{KMPN}} = (V_{\text{KMPN}}, E_{\text{KMPN}})$, $V_{\text{KMPN}} = V_{\text{KMPHN}}$ and $E_{\text{KMPN}} = E_{\text{KMPHN}} \cup E_{\mathcal{ST}}$, where:

- $E_{\mathcal{ST}} = \{(P_S, P_T) : P_S \in V_{\mathcal{S}}, P_T \in V_{\mathcal{T}} \text{ and } \overline{P_S P_T} \cap B'' = \emptyset, \forall B'' \in \mathcal{B}, i = 1, 2\}$. Set of edges formed by the line segments that join the point selected in any source neighbourhood $S \in \mathcal{S}$ and the point selected in any target neighbourhood $T \in \mathcal{T}$.

3. MINLP Formulations

This section proposes a mixed-integer non-linear programming formulation (MINLP) for the problem described in Section 2. First of all, the conic programming representation of the neighbourhoods and distance is presented. Then, the constraints that check if a segment is included in the set of edges E_X with $X \in \{\text{KMPHN}, \text{KMPN}\}$ are set. Finally, the formulations for the H-KMPHN and H-KMPN are described and compared.

First of all, we introduce the decision variables that represent the problem. They are summarized in Table 1.

3.1. Conic programming constraints in the models

For the two problems considered in this paper, namely H-KMPHN and H-KMPN, there exist two typologies of second-order cone constraints. One of them models the distance between each pair of points P and Q in V_X, $X \in \{\text{KMPHN}, \text{KMPN}\}$, and the other, the representation of source and target neighbourhoods, where the points are chosen.

Firstly, we define the non-negative continuous variable $d(PQ)$ that represents the distance between P and Q:

$$\|P - Q\| \leq d(PQ), \quad \forall (P, Q) \in E_X, \tag{d-C}$$

where E_X is the set of edges E_{KMPHN} or E_{KMPN}, depending on the considered problem.

Secondly, since we are assuming that the neighbourhoods are second-order cone (SOC) representable, they can be expressed by means of the constraints:

$$P_N \in N \iff \|A_N^i P_N + b_N^i\| \leq (c_N^i)^T P_N + d_N^i, \quad i = 1, \ldots, n(N), \tag{N-C}$$

where A_N^i, b_N^i, c_N^i and d_N^i are parameters of the constraint i and $n(N)$ denotes the number of constraints that appear in the block associated with the neighbourhood $N \in \mathcal{S} \cup \mathcal{T}$.

These inequalities can model the special case of linear constraints (for $A_N^i, b_N^i \equiv 0$), ellipsoids and hyperbolic constraints (see Lobo et al. (1998) and Boyd and Vandenberghe (2004) for more information).

Table 1: Summary of decision variables used in the mathematical programming model

Binary Decision Variables	
Name	**Description**
$\alpha(P\|QQ')$	1, if the determinant $\det(P\|QQ')$ is positive, 0, otherwise.
$\beta(PP'\|QQ')$	1, if the determinants $\det(P\|QQ')$ and $\det(P'\|QQ')$ have the same sign, 0, otherwise.
$\gamma(PP'\|QQ')$	1, if the determinants $\det(P\|QQ')$ and $\det(P'\|QQ')$ are both positive, 0, otherwise.
$\delta(PP'\|QQ')$	1, if the line segments $\overline{PP'}$ and $\overline{QQ'}$ do not intersect, 0, otherwise.
$\varepsilon(PP')$	1, if the line segment $\overline{PP'}$ does not cross any barrier, 0, otherwise.
$f(PQ\|ST)$	1, if edge (P,Q) is traversed in the path joining S and T, 0, otherwise.
$y(S)$	1, if a facility is allocated in the source S in the solution of the model, 0, otherwise.
$x(ST)$	1, if source S and target T are joined by a path in the solution of the model, 0, otherwise.
Continuous Decision Variables	
Name	**Description**
P_N	Coordinates representing the point selected in the neighbourhood $N \in \mathcal{S} \cup \mathcal{T}$.
$d(PQ)$	Euclidean distance between the points P and Q.

3.2. Checking whether a segment is an edge of the induced graph

The goal of this subsection is to represent by linear constraints a test to check whether given two arbitrary vertices $P, Q \in V_X$, the edge $(P,Q) \in E_X$, with $X \in \{\text{KMPHN}, \text{KMPN}\}$, i.e., whether the line segment $\overline{PQ}$ does not intersect with any barrier of $\mathcal{B}$. The following well-known computational geometry result can be used to check if two line segments intersect.

Remark 1. *Let $\overline{PQ}$ and $B = \overline{P_B^1 P_B^2} \in \mathcal{B}$ be two different line segments. If*

$$\text{sign}\left(\det(P|P_B^1P_B^2)\right) = \text{sign}\left(\det(Q|P_B^1P_B^2)\right) \quad \text{or} \quad \text{sign}\left(\det(P_B^1|PQ)\right) = \text{sign}\left(\det(P_B^2|PQ)\right),$$

then $\overline{PQ}$ and B do not intersect.

Let $P, Q \in V_X$, where V_X can be the set of vertices V_{KMPHN} or V_{KMPN}. Let $P_B^1, P_B^2 \in V_{\mathcal{B}}$ also be the two extreme points determining the barrier $B \in \mathcal{B}$. To model the conditions of the Remark 1, the use of

binary variables that verify the sign of determinants, the equality of signs, and the disjunctive condition are required, since these determinants depend on the location of P and Q.

Firstly, the sign of each determinant in Remark 1 is modelled. The binary variable α is introduced and assumes the value one if the determinant is non-negative and zero, otherwise. Note that determinants can not be null, because the barriers are located in general position.

The following constraints represent the sign condition:

$$\begin{aligned}
\left[1-\alpha(P|P_B^1P_B^2)\right] L(P|P_B^1P_B^2) \le \det(P|P_B^1P_B^2) \le U(P|P_B^1P_B^2)\,\alpha(P|P_B^1P_B^2), \qquad (\alpha\text{-C})\\
\left[1-\alpha(Q|P_B^1P_B^2)\right] L(Q|P_B^1P_B^2) \le \det(Q|P_B^1P_B^2) \le U(Q|P_B^1P_B^2)\,\alpha(Q|P_B^1P_B^2),\\
\left[1-\alpha(P_B^1|PQ)\right] L(P_B^1|PQ) \le \det(P_B^1|PQ) \le U(P_B^1|PQ)\,\alpha(P_B^1|PQ),\\
\left[1-\alpha(P_B^2|PQ)\right] L(P_B^2|PQ) \le \det(P_B^2|PQ) \le U(P_B^2|PQ)\,\alpha(P_B^2|PQ),
\end{aligned}$$

where L and U are lower and upper bounds for the value of the corresponding determinants, respectively. If a determinant is non-negative, then α must be one to make the second inequality feasible. Analogously, if the determinant is not positive, α must be zero to satisfy the correct condition.

Secondly, to check whether the sign of any pair

$$\det(P|P_B^1P_B^2),\ \det(Q|P_B^1P_B^2) \quad \text{or} \quad \det(P_B^1|PQ),\ \det(P_B^2|PQ) \tag{1}$$

of determinants is the same, the binary variable β is defined, that is one if the corresponding pair has the same sign, and zero otherwise.

Hence, the correct value of β variable can be expressed by the following constraint of the α variables

$$\begin{aligned}
\beta(PQ|P_B^1P_B^2) &= \alpha(P|P_B^1P_B^2)\alpha(Q|P_B^1P_B^2) + \left[1-\alpha(P|P_B^1P_B^2)\right]\left[1-\alpha(Q|P_B^1P_B^2)\right],\\
\beta(P_B^1P_B^2|PQ) &= \alpha(P_B^1|PQ)\alpha(P_B^2|PQ) + \left[1-\alpha(P_B^1|PQ)\right]\left[1-\alpha(P_B^2|PQ)\right].
\end{aligned}$$

This condition can be equivalently written by means of an auxiliary binary variable γ that models the product of the α variables:

$$\begin{aligned}
\beta(PQ|P_B^1P_B^2) &= 2\gamma(PQ|P_B^1P_B^2) - \alpha(P|P_B^1P_B^2) - \alpha(Q|P_B^1P_B^2) + 1, \qquad (\beta\text{-C})\\
\beta(P_B^1P_B^2|PQ) &= 2\gamma(P_B^1P_B^2|PQ) - \alpha(P_B^1|PQ) - \alpha(P_B^2|PQ) + 1,
\end{aligned}$$

These γ variables can be linearized by using the following constraints:

$$\begin{aligned}
&\gamma(PQ|P_B^1P_B^2) \le \alpha(P|P_B^1P_B^2), && \gamma(P_B^1P_B^2|PQ) \le \alpha(P_B^1|PQ), \qquad (\gamma\text{-C})\\
&\gamma(PQ|P_B^1P_B^2) \le \alpha(Q|P_B^1P_B^2), && \gamma(P_B^1P_B^2|PQ) \le \alpha(P_B^2|PQ),\\
&\gamma(PQ|P_B^1P_B^2) \ge \alpha(P|P_B^1P_B^2) + \alpha(Q|P_B^1P_B^2) - 1, && \gamma(P_B^1P_B^2|PQ) \ge \alpha(P_B^1|PQ) + \alpha(P_B^2|PQ) - 1.
\end{aligned}$$

Thirdly, verifying whether there exists any coincidence of the sign of determinants is required, so a binary variable δ is defined assuming the value one if segments do not intersect and zero, otherwise. This condition can be modelled by adopting the following disjunctive constraints:

$$\frac{1}{2}\left[\beta(PQ|P_B^1P_B^2) + \beta(P_B^1P_B^2|PQ)\right] \le \delta(PQ|P_B^1P_B^2) \le \beta(PQ|P_B^1P_B^2) + \beta(P_B^1P_B^2|PQ). \qquad (\delta\text{-C})$$

Indeed, the above restrictions state that if there exists a sign coincidence in any of the two pairs of determinants in (1), then δ is one to satisfy the left constraint, and the right one is always fulfilled. However, if none of the signs of any pairs of determinants is the same, then the second constraint is zero and δ must be null.

Finally, to check whether

$$\overline{PQ} \cap B'' = \emptyset, \quad \forall B'' \in \mathcal{B}, \quad \Longleftrightarrow \quad \delta(PQ|P^1_{B''}P^2_{B''}) = 1, \quad \forall B'' \in \mathcal{B},$$

the binary variable $\varepsilon(PQ)$ is introduced, and it is one if this condition is verified for all $B'' \in \mathcal{B}$. This variable can be expressed as:

$$\left[\sum_{B'' \in \mathcal{B}} \delta(PQ|P^1_{B''}P^2_{B''}) - |\mathcal{B}|\right] + 1 \leq \varepsilon(PQ) \leq \frac{1}{|\mathcal{B}|} \sum_{B'' \in \mathcal{B}} \delta(PQ|P^1_{B''}P^2_{B''}). \qquad (\varepsilon\text{-C})$$

If there exists, at least, a barrier $B'' \in \mathcal{B}$ that intersects the segment $\overline{PQ}$, then $\delta(PQ|P^1_{B''}P^2_{B''})$ is zero and the second inequality enforces ε to be zero because the right hand side is fractional and the first inequality is non-positive. However, if no barrier intersects the segment $\overline{PQ}$, then ε is equals to one, because the left hand side of the first inequality is one and the right hand side of the second inequality too.

It is possible to identify the set of actual edges of graph G_X by using the ε variables based on the above description, as follows:

$$E_X = \{(P,Q) : P,Q \in V_X \wedge \varepsilon(PQ) = 1, P \neq Q\}, \quad X \in \{\text{KMPHN}, \text{KMPN}\}.$$

This representation of E_X with $X \in \{\text{KMPHN}, \text{KMPN}\}$ will be applied in the formulations that are presented in the following subsections.

It is interesting to note that $E_{\mathcal{B}}$ is a fixed set whose edges can be computed by using the Remark 1. Then, ε variables can be prefixed in advance. However, edges in $E_X \setminus E_{\mathcal{B}}$ depend on the points selected in the neighbourhoods. A special case that can be highlighted happens when the set of neighbourhoods, $\mathcal{S}$ and $\mathcal{T}$, are represented by points. In that case, the induced graph is completely fixed and it is only necessary to find which edges are included by keeping in mind that the graph must be planar, i.e., without crossings.

3.3. A formulation for the H-KMPHN

The formulation of the H-KMPHN is based on the structure of the well-known k-Median Problem where distances between each pair of source-target neighbourhoods are represented by the shortest path joining them without traversing any barrier.

Note that, although computing the shortest paths between every pair of neighbourhoods is possible, converting an instance of the H-KMPHN into an instance of the standard k-median is not, since the points in neighbourhoods are not fixed. However, this simplification can be applied to produce an approximation to generate lower bounds for the H-KMPHN.

Firstly, it is necessary to define the binary variables inherited from the k-median:

- $y(S)$, that assumes value one if the source neighbourhood $S \in \mathcal{S}$ is selected.

- $x(ST)$, that is one if the target neighbourhood $T \in \mathcal{T}$ is assigned to the selected source $S \in \mathcal{S}$.

Secondly, adjusting a single-commodity flow formulation to ensure connectivity will also be used. The idea is that the model must deliver one unit of commodity from the selected source neighbourhood to each of the required target neighbourhoods. Then, for each edge $(P,Q) \in E_{\text{KMPHN}}$, a binary variable $f(PQ|ST)$ is defined, and takes the value of one when edge (P,Q) is traversed in the path to go from the source S to the target T.

Then, the inequalities

$$f(PQ) \leq |\mathcal{T}|\varepsilon(PQ), \tag{f-C}$$

are included to assure that the delivery can go from P to Q only if the segment $\overline{PQ}$ does not cross any barrier.

Hence, we can adjust the flow formulation to the induced graph G_{KMPHN} as follows:

$$\begin{aligned}
\text{minimize} \quad & \alpha_E \sum_{(P,Q)\in E_{\text{KMPHN}}} \sum_{S\in\mathcal{S}} \sum_{T\in\mathcal{T}} d(PQ) f(PQ|ST) + \frac{\alpha_L}{2} \sum_{(P,Q)\in E_{\text{KMPHN}}} \sum_{S\in\mathcal{S}} \sum_{T\in\mathcal{T}} f(PQ|ST) && \text{(H-KMPHN)}\\
\text{subject to} \quad & \sum_{S\in\mathcal{S}} y(S) = k, && \\
& x(ST) \leq y(S), && \forall S \in \mathcal{S}, \quad \forall T \in \mathcal{T},\\
& \sum_{S\in\mathcal{S}} x(ST) = 1, && \forall T \in \mathcal{T},\\
& \sum_{\{Q\in V_{\text{KMPHN}}:(P,Q)\in E_{\text{KMPHN}}\}} f(PQ|ST) - \sum_{\{Q\in V_{\text{KMPHN}}:(Q,P)\in E_{\text{KMPHN}}\}} f(PQ|ST) = \begin{cases} x(ST), & \text{if } P \in S,\\ 0, & \text{if } P \in V_{\mathcal{B}},\\ -x(ST), & \text{if } P \in T. \end{cases} && \forall S \in \mathcal{S}, \forall T \in \mathcal{T},\\
& (\alpha\text{-C}), (\beta\text{-C}), (\gamma\text{-C}), (\delta\text{-C}) \quad \forall P, Q \in V_{\text{KMPHN}}, \quad \forall P_B^1, P_B^2 \in V_{\mathcal{B}}, && \\
& (\varepsilon\text{-C}), (f\text{-C}), (d\text{-C}) \quad \forall P, Q \in V_{\text{KMPHN}}, && \\
& (N\text{-C}) \quad \forall P \in V_{\mathcal{S}} \cup V_{\mathcal{T}}. &&
\end{aligned}$$

The objective function takes into account both the Euclidean and link weighted distances to join the selected sources with their assigned targets. The first group of constraints imposes that a subset of k sources is selected in $\mathcal{S}$. The second constraints ensure that one target T is assigned to a source S only if it is selected. The third inequalities ensure that every target is assigned to one source. The fourth ones are the flow conservation constraints, where the units of commodity launched from the source must be the number of targets that are assigned to that source. The fifth group of inequalities ensures that the trip can reach, in endurance terms, the target T when it starts from the source S. Constraints (α-C), (β-C), (γ-C), (δ-C), (ε-C), (f-C), (d-C), (N-C) enforce the variables of the problem to be well-defined.

To deal with the bilinear terms that appear in the objective function, the McCormick's envelope are used to linearize them by including variables $p(PQ|ST) \geq 0$ that represent the products and introducing the following constraints:

$$\begin{aligned}
& p(PQ|ST) \geq m(PQ) f(PQ|ST),\\
& p(PQ|ST) \geq d(PQ) - M(PQ)(1 - f(PQ|ST)),
\end{aligned}$$

where $m(PQ)$ and $M(PQ)$ are, respectively, lower and upper bounds of the distance variable $d(PQ)$.

Proposition 1. *The* H-KMPHN *is NP-complete.*

Note that, once a point is fixed in each neighbourhood, the problem that results in the induced graph G_{KMPHN} is the k-median with geodesic distances, that is NP-complete. Figure 2 shows the solutions of the problem data in Figure 1 for $k = 1, 2, 3$, respectively.

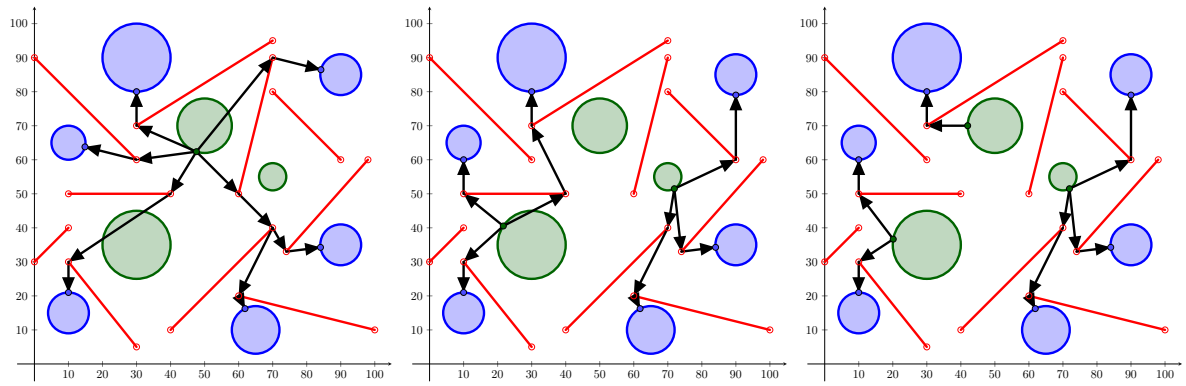


Figure 2: Optimal solution for the H-KMPHN

3.4. Relaxing the assumptions of the problem: The H-KMPN

In this subsection, we analyze the differences between the H-KMPHN and the H-KMPN, where it is allowed that rectilinear paths joining a source neighbourhood with a target neighbourhood may exist. The main difference lies in the description of the edges of the graph induced by the neighbourhoods and the endpoints of the barriers, as shown in Subsection 2.2.

By taking the same approach as before, the sets that describe the graph in the new case are V_{KMPN} and E_{KMPN}, as described in Subsection 3.3. The formulation for the H-KMPN is as follows:

$$\text{minimize} \quad \alpha_E \sum_{(P,Q)\in E_{\text{KMPN}}} \sum_{S\in\mathcal{S}} \sum_{T\in\mathcal{T}} d(PQ) f(PQ|ST) + \frac{\alpha_L}{2} \sum_{(P,Q)\in E_{\text{KMPN}}} \sum_{S\in\mathcal{S}} \sum_{T\in\mathcal{T}} f(PQ|ST) \tag{H-KMPN}$$

$$\text{subject to} \quad \sum_{S\in\mathcal{S}} y(S) = k,$$

$$x(ST) \le y(S), \qquad \forall S \in \mathcal{S}, \quad \forall T \in \mathcal{T},$$

$$\sum_{S\in\mathcal{S}} x(ST) = 1, \qquad \forall T \in \mathcal{T},$$

$$\sum_{\{Q\in V_{\text{KMPN}}:(P,Q)\in E_{\text{KMPN}}\}} f(PQ|ST) - \sum_{\{Q\in V_{\text{KMPN}}:(Q,P)\in E_{\text{KMPN}}\}} f(PQ|ST) = \begin{cases} x(ST), & \text{if } P \in S, \\ 0, & \text{if } P \in V_{\mathcal{B}}, \\ -x(ST), & \text{if } P \in T. \end{cases} \qquad \forall S \in \mathcal{S}, \forall T \in \mathcal{T},$$

$$(\alpha\text{-C}), (\beta\text{-C}), (\gamma\text{-C}), (\delta\text{-C}) \quad \forall P, Q \in V_{\text{KMPN}}, \quad \forall P_B^1, P_B^2 \in V_{\mathcal{B}},$$

$$(\varepsilon\text{-C}), (f\text{-C}), (d\text{-C}) \quad \forall P, Q \in V_{\text{KMPN}},$$

$$(N\text{-C}) \quad \forall P \in V_{\mathcal{S}} \cup V_{\mathcal{T}}.$$

The difference between the set of edges in the H-KMPHN with respect to the graph in H-KMPN is that, in the former case, the edges that join each pair of neighbourhoods must be considered. This fact leads to include product of continuous variables in the α constraints of the model that represent the determinants. These products make the problem to become non-convex. Alternatively, in the H-KMPHN, since two of the three arguments of the determinant are fixed, the α constraints become linear.

Again, the problem data in Figure 1, with a smaller number of barriers, is used to illustrate the solutions of the H-KMPN for $k = 1, 2, 3$, respectively.

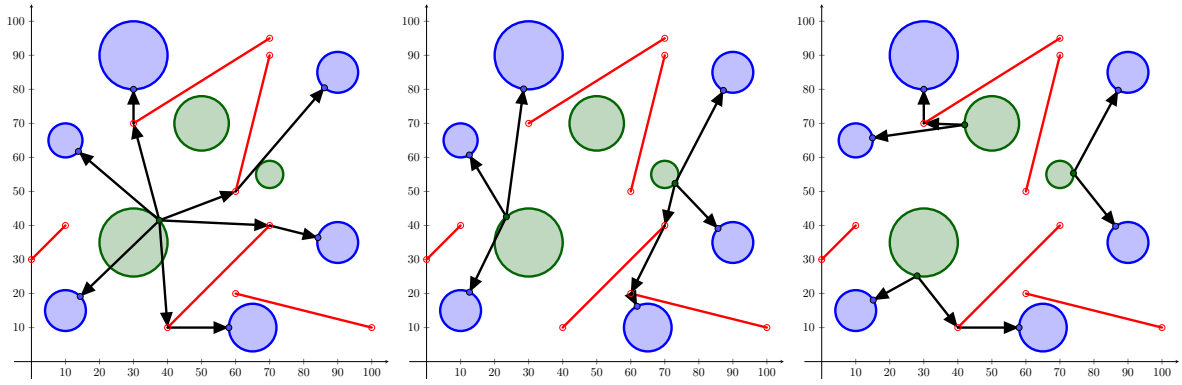


Figure 3: Optimal solution for the H-KMPN

4. Matheuristic Algorithm

The two considered problems are computationally very hard, and, at times, finding even feasible solutions becomes a challenge for large-size instances. In order to provide initial solutions to our algorithms, we have developed an easy procedure based on a reduction to the classical K-median problem with barriers.

In this section, we describe a matheuristic approach based on the formulations presented above, which is able to handle larger instances of the problem. This matheuristic provides solutions of H-KMPHN or H-KMPN, that can be used to initialise the solver for any of the formulations of these models proposed above.

The basic idea of this procedure is to consider only the centres of each neighbourhood as the points chosen in each of them. By fixing those points, the graphs G_{KMPHN} and G_{KMPN} induced in our construction are also fixed, their respective edges can be preprocessed, and the resulting problems become mixed-integer linear. This reduction allows us to obtain feasible solutions for them using any of the available solvers for mixed-integer programming. Furthermore, since the sizes we can handle for the general problems H-KMPHN and H-KMPN are in the range of 50 to 80 neighbourhoods, solving the point-wise versions takes a short time.

5. Computational Experiments

This section is dedicated to evaluating the performance of the formulations H-KMPHN and H-KMPN and the behaviour of the matheuristic applied to the two problems considered. First, we present details of the data generation. Second, the design of the experiments is stated. Finally, we report the results obtained in our computational experiments.

5.1. Data generation

Assumptions **A1-A4** stated in Section 2 are assumed to generate the instances of the experiments. In this case, w.l.o.g., the neighbourhoods generated are circles. The sketch of the procedure is as follows.

1. Random sampling of points in a square.

2. Generation of bisectors that separate any pair of points.

3. Generation of neighbourhoods that satisfy **A4**.

The following pseudocode describes the details of the construction of the instances.

Algorithm 1: Generation of instances of H-KMPHN

Initialization: Let $|\mathcal{N}|$ be the number of neighbourhoods to generate. Let $r_{\text{init}} = 10$ be half of the initial length of the barriers. Set $\mathcal{N} = \{\}$; $points = \{\}$; $\mathcal{B} = \{\overline{(0,0)(100,0)}, \overline{(100,0)(100,100)}, \overline{(100,100)(0,100)}, \overline{(0,100)(0,0)}\}$.

1 Generate $|\mathcal{N}|$ points uniformly distributed in the square $[0,100]^2$ and include them in *points*.
2 **for** $P, P' \in points$ **do**
3 **if** $\overline{PP'} \cap B = \emptyset$, $\forall B \in \mathcal{B}$ **then**
4 Compute $\overrightarrow{d} = \overrightarrow{PP'}$.
5 Compute $M = P + \frac{1}{2}\overrightarrow{d}$.
6 Compute the unitary vector $\overrightarrow{n_u}$ perpendicular to $\overrightarrow{d}$.
7 Set $r = r_{\text{init}}$.
8 Generate the barrier $B(r) = \overline{P_B^+ P_B^-}$ where $P_B^{\pm} = M \pm r\overrightarrow{n_u}$.
9 **while** $B(r) \cap B' \neq \emptyset$ *for some* $B' \in \mathcal{B}$ **do**
10 Set $r := r/2$.
11 Generate the barrier $B(r)$.
12 Include $B(r)$ in $\mathcal{B}$.
13 **for** $P \in points$ **do**
14 Set $r_{\max} = \min_{\{P_B \in B : B \in \mathcal{B}\}} d(P, P_B)$.
15 Generate a random *radii* uniformly distributed in the interval $\left[\frac{1}{2}r_{\max}, r_{\max}\right]$.
16 Set the ball N whose centre is P and radii is *radii*.
17 Include N in $\mathcal{N}$.

Lines 9-11 ensure that neighbourhoods are not enclosed inside of the bisectors so that paths can exit from any neighbourhood to visit another one. Lines 13-17 set a maximum radii for the balls that ensure that they are hidden behind barriers.

Note that, since H-KMPN does not assume **A4**, it is only required to remove some bisectors for the instances generated before to ensure that it is possible to go directly from one neighbourhood to another.

5.2. Configuration of the experiments

To explore the behaviour of the formulations described in the paper, we report on a series of experiments that vary most of the parameters that describe the models. Once they are solved, it is important to give some measures that make them comparable with any others that may be available in the literature. First, the experimental parameters are reported. Then, the computer framework where the experiments were performed is described. Finally, the reported solution values are explained in detail.

Since there are no benchmark instances available for this problem in the literature, five instances for each $|\mathcal{N}| \in \{10, 20, 30, 50, 80\}$ have been generated following Algorithm 1. For each instance, both

sources and targets are the whole $\mathcal{N}$ set, i.e., each neighbourhood can be chosen as a source and each one must be associated with a facility. Furthermore, the number of neighbourhood facilities k, for each problem, is 1 and a percentage perc_k, for $\text{perc}_k \in \{10, 25\}$ of the whole set $\mathcal{N}$. To avoid **A4**, a percentage $\text{perc}_{|\mathcal{B}|}$ of all barriers is selected for $\text{perc}_{|\mathcal{B}|} \in \{10, 20, 50\}$. Finally, we set $\alpha_E = 1$ and $\alpha_L = 50$ to take into account both terms of the distances considered in the objective function.

For each combination of the above factors, both H-KMPHN and H-KMPN formulations are tested without and with the initial solution provided by the matheuristic described in Section 4. A time limit of 3600 seconds was set in the experiments for the formulation and 100 seconds for the matheuristic.

Formulations were coded in Python 3.9.2 (G. van Rossum (Guido) (1995)) and solved in Gurobi 9.1.2 (Gurobi Optimization LLC (2022)) on an AMD® Epyc 7402p 8-core processor.

The values obtained by Gurobi that are reported in our tables are:

- $\#Found$: number of instances in which the solver could find a solution.
- Gap: gap between the best incumbent solution with respect to the best bound found by the solver.
- $Runtime$: time spent by the solver to obtain the best solution found.
- Gap_{math}: relative gap between the best incumbent solution given by the matheuristic and the best solution found by the solver after the time limit.
- $Runtime_{math}$: time spent by the matheuristic to obtain the best solution.
- Gap_{build}: relative gap between the value of the first incumbent solution built by the solver and the solution provided by the matheuristic.
- $Runtime_{build}$: time spent by the solver to build the initial solution provided by the matheuristic.

The above information allows us to compare the difficulty of the problem, and, in addition, to test the matheuristic performance to obtain solutions for instances generated as stated in Subsection 5.1.

5.3. Results of the experiments

Analysing the results in Tables 2 and 3, we first note that the solver is capable of finding feasible solutions up to 80 neighbourhoods when the formulation H-KMPHN is considered. Furthermore, these solutions are optimal up to a number of 30 neighbourhoods. Moreover, the solver reports maximum gaps of $14,46\%$ and $31,4\%$ for a number of 50 and 80 neighbourhoods, respectively. It is noteworthy that these gaps are obtained when a 10% of the entire set must be chosen as facilities. For $k = 1$, the maximum gap reported by the solver is $7,84\%$ after the time limit. In addition, this case is considerably the fastest (and the easiest) one to obtain the optimal solution with Gurobi, whenever it is found. Finally, one can remark that the initial solution found by the matheuristic helps to certify optimality in less time and also to get better gaps whenever the optimal solution is not found within the time limit. Note that the maximum gap between the best incumbent solution obtained by the matheuristic in a time limit of 100 seconds and the best by the exact formulation after an hour is 5.96%. This is an indication that the mathheuristic is a good alternative to the exact method whenever the problem size grows.

With regard to the more general problem modelled with formulation H-KMPN, the non-convex nature of the model makes the problem even harder. This can be seen in terms of the resulting gap after the time limit (see Figure 4). The solver starts to not close this gap already for 20 neighbourhoods and half of the original barriers considered. In addition, for instances with 50 and 80 neighbourhoods, the solver cannot find even a feasible solution when it is not initialised. Moreover, Gurobi cannot even build, in some cases, the solution provided by the matheuristic. The case where only one neighbourhood is selected ($k = 1$, e.g., the single facility case) is the hardest one to be solved. This is counterintuitive and is the opposite of the behaviour shown by the formulation H-KMPHN. This case has become the hardest, in our computational experience, because choosing only one point to serve all the required neighbourhoods implies to jointly satisfy a large bunch of non-convex constraints with the same point. This seems to be very difficult for the solver. Another remarkable fact is that the higher the percentage of barriers that are set, the greater the difficulty of the problem, taking into account that the location of the neighbourhood remains the same for all cases. It can be explained in terms of constraints and variables that depend on the number of barriers that are considered. Finally, we can conclude that the matheuristic works well for medium to large-size instances. For those, the matheuristic algorithm always finds a feasible solution that can not be improved by the solver, as can be observed in terms of the relative gap. In addition, in all cases the feasible solutions given by the matheuristic are obtained in a maximum computing time of 100 seconds. This makes the matheuristic an efficient method to provide good-quality solutions in short computing time.

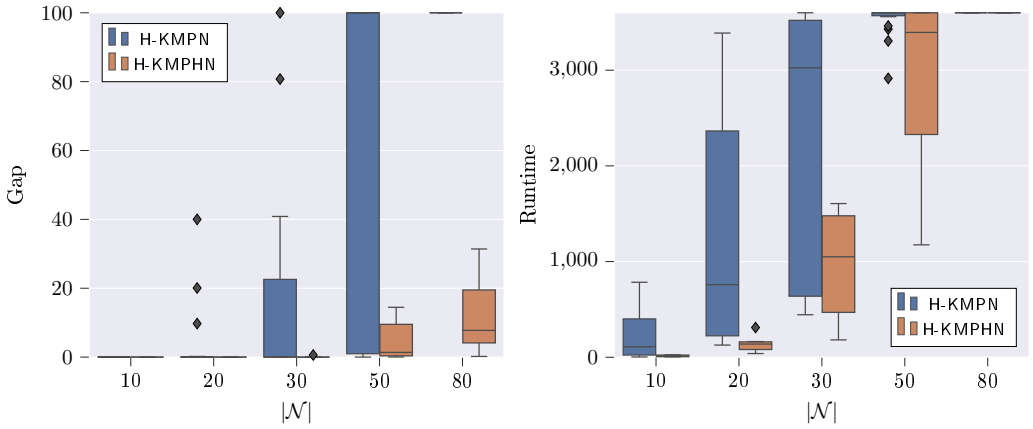


Figure 4: Gap and Runtime reported by Gurobi when the model is initialised by the matheuristic

6. Concluding remarks

This paper has dealt with the H-KMPHN and its general version called H-KMPN, in which the assumption that neighbourhoods are not visible to one another is removed. This last version leads to non-convex mixed-integer problems whereas the first one results in second-order cone mixed-integer problems. The two problems, beyond its similarity, show deep differences in terms of computational difficulty, as explained in Section 5. However, the proposed mathematical programming approaches allow a formal treatment that allows one to optimally solve small to medium-size instances. For larger

size instances, this approach also inspires a matheuristic algorithm providing good quality solutions, in short computing time, by exploiting the structure of the problem. It is still an open question whether there is some kind of finite dominating set with polynomial cardinality for the version H-KMPHN, which certainly will simplify the underlying graph structures and the solution of the problem. Moreover, given the complexity of the problem, studying valid inequalities that reduce the space of feasible solutions will be instrumental in solving larger instances efficiently.

In addition, one can consider an extension of these problems assuming limited lengths for the paths between the source and its associated target. It would also be interesting to combine in the same model different typologies of barriers such as piecewise linear and second-order cone-representable sets. Besides, it will deserve some attention to study three-dimensional barriers that simulate buildings that planar paths cannot traverse, thus approaching even more real-life applications in the drone delivery industry.

All of the problems mentioned above are natural extensions of those considered in this paper and may attract the attention of researchers in the future.

Acknowledgements

This research has been partially supported by the Agencia Estatal de Investigación (AEI) and the European Regional Development Fund (ERDF): PID2020-114594GB-C21; and Regional Government of Andalusia: project P18-FR-1422.

References

Blanco, V. (2019). Ordered p-median problems with neighbourhoods. *Computational Optimization and Applications*, 73(2):603–645.

Blanco, V., Fernández, E., and Puerto, J. (2017). Minimum Spanning Trees with neighborhoods: Mathematical programming formulations and solution methods. *European Journal of Operational Research*, 262(3):863–878.

Boyd, S. P. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press, Cambridge, UK ; New York.

Daescu, O., Mitchell, J. S. B., Ntafos, S., Palmer, J. D., and Yap, C. K. (2008). An Experimental Study of Weighted k-Link Shortest Path Algorithms. In Akella, S., Amato, N. M., Huang, W. H., and Mishra, B., editors, *Algorithmic Foundation of Robotics VII: Selected Contributions of the Seventh International Workshop on the Algorithmic Foundations of Robotics*, Springer Tracts in Advanced Robotics, pages 187–202. Springer, Berlin, Heidelberg.

de Berg, M., van Kreveld, M., Nilsson, B. J., and Overmars, M. H. (1990). Finding shortest paths in the presence of orthogonal obstacles using a combined L1 and link metric. In Gilbert, J. R. and Karlsson, R., editors, *SWAT 90*, Lecture Notes in Computer Science, pages 213–224, Berlin, Heidelberg. Springer.

Drezner, Z. and Hamacher, H. W., editors (2004). *Facility Location: Applications and Theory*. Springer, Berlin.

G. van Rossum (Guido) (1995). Python reference manual. Issue: R 9525 Publication Title: Department of Computer Science [CS].

Gentilini, I., Margot, F., and Shimada, K. (2013). The travelling salesman problem with neighbourhoods: MINLP solution. *Optimization Methods and Software*, 28(2):364–378. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10556788.2011.648932.

Gurobi Optimization LLC (2022). Gurobi Optimizer Reference Manual.

Klamroth, K. (2002). *Single-Facility Location Problems with Barriers*. Springer, New York, NY.

Kuehn, A. A. and Hamburger, M. J. (1963). A Heuristic Program for Locating Warehouses. *Management Science*, 9(4):643–666. Publisher: INFORMS.

Lobo, M. S., Vandenberghe, L., Boyd, S., and Lebret, H. (1998). Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1):193–228.

Mitchell, J. S. B. (2017). Shortest Paths and Networks. In *Handbook of Discrete and Computational Geometry*. Chapman and Hall/CRC, 3 edition. Num Pages: 38.

Nickel, S. and Puerto, J. (2007). S. Nickel and J. Puerto: Location theory: a unified approach. *Mathematical Methods of Operations Research*, 66(2):369–371.

Puerto, J. (2008). A New Formulation of the Capacitated Discrete Ordered Median Problems with {0, 1}-Assignment. In Kalcsics, J. and Nickel, S., editors, *Operations Research Proceedings 2007*, Operations Research Proceedings, pages 165–170, Berlin, Heidelberg. Springer.

Puerto, J. and Valverde, C. (2022). Routing for unmanned aerial vehicles: Touring dimensional sets. *European Journal of Operational Research*, 298(1):118–136.

Ulukan, Z. and Demircioğlu, E. (2015). A Survey of Discrete Facility Location Problems. *World Academy of Science, Engineering and Technology, International Journal of Industrial and Manufacturing Engineering*.

Yuan, B. and Zhang, T. (2017). Towards Solving TSPN with Arbitrary Neighborhoods: A Hybrid Solution. In *ACALCI*, pages 204–215.

$\|\mathcal{N}\|$	A_4	$\text{perc}_{\|\mathcal{B}\|}$	Initialization	k	$\#Found$	Gap	$Runtime$	Gap_{math}	$Runtime_{math}$	Gap_{build}	$Runtime_{build}$
10	no	10	no	1	5	0	362,95	-	-	-	-
				25%	5	0	6,97	-	-	-	-
			yes	1	5	0	2,94	40,72	0,01	27,94	0,61
				25%	5	0	9,54	54,76	0,01	39,15	0,88
		20	no	1	5	0	387,67	-	-	-	-
				25%	5	0	27,96	-	-	-	-
			yes	1	5	0,01	440,19	32,44	0,01	18,99	10,14
				25%	5	0	27,92	46,2	0,02	13,09	17,97
		50	no	1	5	0,02	452,6	-	-	-	-
				25%	5	0	103,53	-	-	-	-
			yes	1	5	0,01	783,8	28,97	0,02	13,19	70,59
				25%	5	0,01	114	37,36	0,04	29,09	19,08
	yes	100	no	1	5	0	6,16	-	-	-	-
				25%	5	0	25,72	-	-	-	-
			yes	1	5	0	4,19	5,28	0,03	2,97	0,8
				25%	5	0	21,99	4,7	0,26	1,83	1,01
20	no	10	no	1	5	0	127,18	-	-	-	-
				10%	5	0,01	132,99	-	-	-	-
				25%	5	0	175,68	-	-	-	-
			yes	1	5	0,02	775,61	36,23	0,04	8,82	57,03
				10%	5	0	173,2	42,66	0,18	28,25	18,98
				25%	5	0	150,86	56,08	0,03	24,36	60,94
		20	no	1	5	0	740,64	-	-	-	-
				10%	5	0	659,85	-	-	-	-
				25%	5	0	370,03	-	-	-	-
			yes	1	5	0	776,45	25,36	0,06	20,65	42,9
				10%	5	0	1039,52	37,39	0,65	28,61	268,46
				25%	5	0	667,29	50,57	0,15	20,16	257,77
		50	no	1	5	0,19	3044,11	-	-	-	-
				10%	5	0,13	2804,13	-	-	-	-
				25%	5	0,01	1747,37	-	-	-	-
			yes	1	5	9,73	3387,96	12,4	0,12	12,4	62,52
				10%	5	20,07	2852,9	21,62	1,32	20,11	87,53
				25%	5	40	2570,65	25,72	1,06	25,4	41,2
	yes	100	no	1	5	0	66,27	-	-	-	-
				10%	5	0	310,58	-	-	-	-
				25%	5	0	151,38	-	-	-	-
			yes	1	5	0	38,96	5,31	0,16	4,45	3,79
				10%	5	0	164,02	5,96	3,03	4,64	2,92
				25%	5	0	124,26	3,97	4,5	1,77	3,05
30	no	10	no	1	5	0	611,18	-	-	-	-
				10%	5	0	560,1	-	-	-	-
				25%	5	0	454,2	-	-	-	-
			yes	1	5	0	552	30,3	0,2	28,55	35,12
				10%	5	0	718,61	43,19	0,43	33,02	172,55
				25%	5	0	444,33	54,66	0,03	37,28	113,71
		20	no	1	3	0,01	3472,27	-	-	-	-
				10%	3	0	3024,87	-	-	-	-
				25%	4	0	2208,42	-	-	-	-
			yes	1	5	23,01	3240,75	15,44	0,25	13,11	104,78
				10%	5	21,24	3023,67	29,52	7,01	25,05	230,37
				25%	5	0	1933,3	47,09	0,58	47,09	194,12
		50	no	1	1	27,7	3600	-	-	-	-
				10%	1	10,59	3600	-	-	-	-
				25%	1	0	3451,41	-	-	-	-
			yes	1	5	100	3600	0	0,37	0	-
				10%	5	80,75	3600	5,6	22,04	5,6	-
				25%	5	40,85	3536,92	22,57	3,98	22,57	-
	yes	100	no	1	5	0	324,44	-	-	-	-
				10%	5	0	1572,14	-	-	-	-
				25%	5	0,6	1606,06	-	-	-	-
			yes	1	5	0	181,53	2,54	0,37	2,26	8,01
				10%	5	0	902,58	2,93	69,94	2,29	11,44
				25%	5	0	1197,54	2,76	46,68	2,08	12,13

Table 2: Computational results for 10, 20 and 30 neighbourhoods

$\|\mathcal{N}\|$	A_4	$\text{perc}_{\|\mathcal{B}\|}$	Initialization	k	$\#Found$	Gap	$Runtime$	Gap_{math}	$Runtime_{math}$	Gap_{build}	$Runtime_{build}$
50	no	10	no	1	1	0	3559,43	-	-	-	-
				10%	1	0	3429,18	-	-	-	-
				25%	4	0	2914,82	-	-	-	-
			yes	1	5	100	3600	0	1,43	0	-
				10%	5	80,01	3600	8,78	7,16	8,78	-
				25%	5	40,07	3461,54	32,39	0,12	32,39	-
		20	no	1	0	-	3600	-	-	-	-
				10%	0	-	3600	-	-	-	-
				25%	2	0,99	3305,34	-	-	-	-
			yes	1	5	100	3600	0	1,67	0	-
				10%	5	100	3600	0	100	0	-
				25%	5	100	3600	0	3,76	0	-
		50	no	1	0	-	3600	-	-	-	-
				10%	0	-	3600	-	-	-	-
				25%	0	-	3600	-	-	-	-
			yes	1	5	100	3600	0	1,94	0	-
				10%	5	100	3600	0	100	0	-
				25%	5	100	3600	0	56,46	0	-
	yes	100	no	1	5	0,03	2041	-	-	-	-
				10%	5	14,46	3600	-	-	-	-
				25%	5	1,39	3600	-	-	-	-
			yes	1	5	0,03	1174,67	2,73	2,54	2,67	26,62
				10%	5	12,22	3600	5,15	100	4,77	16,7
				25%	5	1,32	3189,1	1,44	100	1	26,51
80	no	10	no	1	0	-	3600	-	-	-	-
				10%	0	-	3600	-	-	-	-
				25%	0	-	3600	-	-	-	-
			yes	1	5	100	3600	0	8,55	0	-
				10%	5	100	3600	0	82,05	0	-
				25%	5	100	3600	0	0,42	0	-
		20	no	1	0	-	3600	-	-	-	-
				10%	0	-	3600	-	-	-	-
				25%	0	-	3600	-	-	-	-
			yes	1	5	100	3600	0	8,43	0	-
				10%	5	100	3600	0	100	0	-
				25%	5	100	3600	0	16,16	0	-
		50	no	1	0	-	3600	-	-	-	-
				10%	0	-	3600	-	-	-	-
				25%	0	-	3600	-	-	-	-
			yes	1	5	100	3600	0	10,52	0	-
				10%	5	100	3600	0	100	0	-
				25%	5	100	3600	0	61,43	0	-
	yes	100	no	1	5	7,84	3600	-	-	-	-
				10%	5	31,4	3600	-	-	-	-
				25%	5	7,67	3600	-	-	-	-
			yes	1	5	0,19	3600	2,07	5,48	2,01	60,36
				10%	5	23,38	3600	0,64	100	0,46	33,48
				25%	5	2,93	3600	1,8	100	1,36	30,58

Table 3: Computational results for 50 and 80 neighbourhoods

Chapter 9

Coordinating drones with mothership vehicles: The mothership and drone routing problem with graphs

Computers & Operations Research 136 (2021) 105445

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Coordinating drones with mothership vehicles: The mothership and drone routing problem with graphs

Lavinia Amorosi [a,*,1], Justo Puerto [b,1], Carlos Valverde [b,1]

[a] Department of Statistics, Sapienza, University of Rome, Rome 00185, Italy
[b] Department of Statistics and Operations Research, University of Seville, Seville 41012, Spain

ARTICLE INFO

Keywords:
Arc routing problems
Networks
Drones
Conic programming

ABSTRACT

This paper addresses the optimization of routing problems with drones. It analyzes the coordination of one mothership with one drone to obtain optimal routes that have to visit some target objects modeled as general graphs. The goal is to minimize the overall weighted distance traveled by both vehicles while satisfying the requirements in terms of percentages of visits to targets. We discuss different approaches depending on the assumption made on the route followed by the mothership: i) the mothership can move on a continuous framework (the Euclidean plane), ii) on a connected piecewise linear polygonal chain or iii) on a general graph. In all cases, we develop exact formulations resorting to mixed integer second order cone programs that are compared on a testbed of instances to assess their performance. The high complexity of the exact methods makes it difficult to find optimal solutions in short computing time. For that reason, besides the exact formulations we also provide a tailored matheuristic algorithm that allows one to obtain high quality solutions in reasonable time. Computational experiments show the usefulness of our methods in different scenarios.

1. Introduction

In recent years the progress in the field of automation has led to the increasingly widespread use of drone technology in many sectors (see Otto et al. (2018) and Chung et al. (2020) for a survey in civil applications). Depending on the application, these devices are used to support or replace humans in carrying out operations, and also in the cases of lack of infrastructures (see Poikonen and Campbell (2020) also for future applications and research directions). We can find several examples of this trend in the telecommunication field, where drones can be used to provide connectivity in rural areas without antennas (see for example Amorosi et al. (2019); Chiaraviglio et al. (2019b); Chiaraviglio et al. (2018); Jiménez et al. (2018) and Amorosi et al. (2018)) or in areas affected by natural disasters which have compromised the existing infrastructures (see for example Chiaraviglio et al. (2019a)). In goods delivery activities, especially in the last mile, drones represent a valid tool to support or replace the tasks of drivers by speeding up the service and relieving traffic from big cities or cities with particular configurations where standard vehicles cannot proceed (see for example Pugliese et al. (2017) and Amorosi et al. (2020)). This technology allows to provide a faster and safer response even in emergency contexts, for example for the delivery of medicines or blood bags, Wen et al. (2016). Other uses are also for achieving safer and faster activities of inspection and monitoring, both of networks (such as electricity, gas, telecommunications, railways, roads, etc.) and areas or their portions, depending on the application context. Indeed drones can reach quickly sections of a network that have suffered damage to verify the actual conditions (for example road networks after a storm, electrical or telecommunication networks that have suffered a breakdown, etc.), or allow, for example, to check the state and progress of a fire or an oil spill at sea. The use of this technology in all these different contexts is made advantageous by the fact that, compared to traditional means of transportation (trucks, ships, helicopters), Unmanned Aerial Vehicles (UAVs) have a lower cost per mile, produce less CO2 emissions and can arrive in places that cannot be reached with traditional means. On the other hand, their main limitation is the limited flight time which does not make them usable in full endurance in a number of contexts. For this reason, for some applications, hybrid systems that involve the combined use of drones with other means of transport may represent a more efficient alternative. In this system configurations it is necessary to coordinate and synchronize the operations of drones and other means of transport, taking into account the constraints of limited endurance of the drones and the movement of

* Corresponding author.
E-mail addresses: lavinia.amorosi@uniroma1.it (L. Amorosi), puerto@us.es (J. Puerto), cvalverde@us.es (C. Valverde).
[1] Equally contributing authors.

https://doi.org/10.1016/j.cor.2021.105445
Received 12 February 2021; Received in revised form 21 June 2021; Accepted 22 June 2021
Available online 10 July 2021

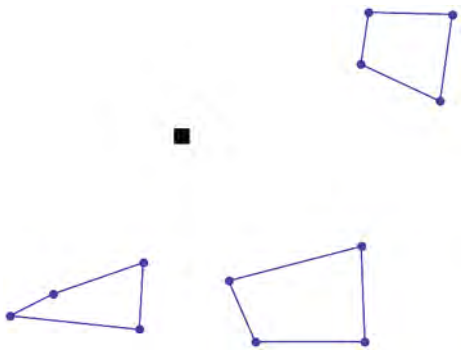


Fig. 1. Example of problem framework.

the other involved vehicles. The different configurations from which these hybrid systems can be characterized have given rise in the literature to different combinatorial optimization problems. Most of the works in the literature have focused on approaches that involve a discretization of the movement space of all the vehicles involved in the system. This provides the advantage of being able to mathematically model the problem more easily and obtain linear or linearizable formulations. However, on the other hand, it does not allow to fully exploit the freedom of movement of the drone which, unlike other means of transport which are constrained to road networks, can move from a point to any other in the continuous space. This paper studies the problem of coordinating a system composed of a mothership (the base vehicle) which supports the operations of one drone which have to visit a set of targets represented by graphs, with the goal of minimizing the total distance travelled by both vehicles. This system configuration can model, for example, monitoring and inspection activities on portions of networks where traditional vehicles cannot arrive, due, for example, to the presence of narrow streets, or because of a natural disaster or a terrorist attack that caused damages on the network. In all these cases, the inspection or monitoring of the drone consists in traversing edges of the network to perform a reconnaissance activity. For this reason we model the targets, to be visited by the drone, as graphs. Differently from previous works in the literature, we assume that the base vehicle and the drone can move freely on the continuous space and we present new Mixed Integer Non-Linear Programming (MINLP) formulations for this problem and a heuristic algorithm derived from the formulation to deal with larger instances. In particular, we consider three variants of the problem, depending on the assumptions made on the route followed by the mothership: i) the case in which it can move on a continuous framework (the Euclidean plane), ii) the case in which it is constrained to move on a polygonal, and ii) the case of a general undirected network. Also for these cases we propose alternative MINLP formulations that exploit the specific characteristics of these models.

The rest of the paper is structured as follows: Section 2 provides a detailed description of the problem under consideration. Section 3 reports the state of the art on routing problems with drones, mainly focusing on hybrid systems involving different transportation means. Section 4 presents alternative MINLP formulations proposed to model the different variants of the problem. At the end of this section, Subsection 4.3 provides upper and lower bounds on the big-M constants introduced in the proposed formulations to tighten them. Section 5 presents the details of the matheuristic algorithm designed to handle large instances. In Section 6 we report the results obtained testing the formulations presented in Section 5 on different classes of planar graphs and the comparison with the ones provided by the matheuristic procedure in order to evaluate its effectiveness. Finally, Section 6 concludes the paper. For the sake of presentation, we have included an Appendix at the end of the paper including some details on some of the formulations and also on how to strengthen them.

2. Description of the problem

In the Mothership and Drone Routing Problem with Graphs (MDRPG), there is one mothership (the base vehicle) and one drone that must operate in coordination to visit a set of targets, represented by graphs, located in a continuous framework that can be modeled as the Euclidean 2-or-3 dimension space. The goal is the minimization of the total distance travelled by both vehicles. The mothership and the drone begin at a common starting location denoted *orig* and they have to coordinate their movements so that the drone visits the set $\mathscr{G}$ of graphs (target objects) whose locations are given. These assumptions allow to model several real situations like inspections and monitoring of roads or wired networks, that can be performed by the drone more easily and safely than standard vehicles.

For each graph $g \in \{1,\dots,|\mathscr{G}|\}$, we require that the drone performs a task consisting in the following four operations: (i) it is launched from the current mothership location (to be determined), (ii) it flies to the graph g that has to be visited, (iii) it traverses the required edges of graph g and then (iv) it returns to the current position of the mothership (to be determined), that most likely is different from the launching point. Indeed, what makes the problem more challenging is that we assume that the mothership can move while the drone visits the targets and thus coordination is a very important issue. Once all target graphs have been visited, the mothership and the drone return to a final location (depot), denoted by *dest*. In all its movements, the drone flies following straight lines.

Fig. 1 shows an example of the problem framework, where the black square represents the origin and destination of the mothership tour. The three blue graphs represent the targets to be visited by the drone, located in the plane.

We assume wlog that the mothership and the drone do not need to arrive at each rendezvous location at the same time: the fastest arriving vehicle may wait for the other at the rendezvous point. In addition, we also assume that vehicles move at constant speeds, although this hypothesis could be relaxed. The mothership and the drone travel at speeds v_M and v_D, respectively. The mothership and the drone must travel together from *orig* to the first launching point. Similarly, after the drone visits the last target graph, the mothership and the drone must meet at the final rendezvous point before traveling together back to *dest*. The first launching and final rendezvous points are allowed to be *orig* and *dest*, respectively, but it is not mandatory. For the ease of presentation, in this paper we will assume that *orig* and *dest* are at the same location. However, all results extend easily to the case that *orig* and *dest* are at different locations.

Summarizing, the MDRPG consists in coordinating the drone with the mothership to minimize the overall distance travelled imposing that the drone has to visit a set of target graphs. In order to do that, it is required to determine: (i) the tour of the mothership starting at *orig*, deciding the different launching and rendezvous points, and returning to *dest*; (ii) the order of visits of the target graphs followed by the drone, determining the corresponding launching and rendezvous points of the drone on each visited graph; and (iii) the tour followed by the drone on each target graph $g \in \mathscr{G}$. The reader should observe that, since we assume constant velocities, the minimization of the travel distances is a natural proxy for the minimization of the overall time, that is the sum of traveling time of the mothership and traveling time of the drone, needed to complete the visits to all target graphs. Indeed, differently from a scheduling problem, we are not minimizing the makespan of the system. In fact, in this context, the length travelled by the drone, and implicitly the travelled time of the drone, has a cost that must be considered in addition to the one associated with the mothership.

Depending on the assumptions made on the movements of the mothership vehicle, this problem gives rise to three different versions: a) the mothership vehicle can move freely on the continuous space (all terrain ground vehicle, boat on the water or aircraft vehicle); b) the mothership can move on a connected piecewise linear polygonal chain;

and c) the mothership can move on a road network (that is, it is a normal truck or van). In the former case, that we will call All terrain Mothership-Drone Routing Problem with Graphs (AMDRPG), each launching and rendezvous location may be chosen from a continuous space (the Euclidean 2-or-3 dimension space). In the latter case, that we will call Network Mothership-Drone Routing Problem with Graphs (NMDRPG), each launching and rendezvous locations must be chosen on a given graph embedded in the considered space. For the sake of presentation and due to the length of the paper, we will mainly focus, on the first model (AMDRPG) and second model (PMDRPG). The third model, namely NMDRPG, is addressed using similar techniques but providing slightly less details reported in the Appendix. Moreover, we consider two different ways for visiting the target graphs: (i) visiting a percentage of each edge of a graph, (ii) visiting a percentage of each graph. These variants derive, for example, from monitoring activities in which it is sufficient a partial visit of the targets. This is the case, for example, of traffic flows monitoring, where, in order to verify if the traffic progression is not disrupted, only inspecting a portion of edge provides a valuable information.

In addition, for the sake of simplicity, we restrict ourselves to the case where there are no obstacles to prevent the drone travelling in straight line. Extensions of this problem with obstacles are very interesting to be further considered although they require different techniques which are beyond the scope of this paper.

3. State of the art

In this section we focus mainly on the previous works in literature related to systems where UAVs are assisted by a vehicle in order to serve a set of targets. In these configurations the vehicle represents a launching and a recharging station for the UAVs and the main problem consists in coordinating the operations performed by one or multiple drones and the mothership. Most of the previous works in literature on this subject focus on node routing problems (NRPs), where the vehicle moves on a road network and the drone is used to visit target points outside the road network. Many of them are related to applications in the delivery sector where the set of targets to be visited is represented by a set of customers. For example, Mathew et al. (2015) studies a delivery system consisting in one drone and one truck. The UAV visits one customer for each trip and the truck can wait at the launching node for the drone to come back or move to a different rendezvous node. In Carlsson and Song (2018) the authors study a continuous approximation on the Horse Fly Problem, where the truck is used as a mobile depot for the drone. In Campbell et al. (2017) the authors evaluate the economic impact of truck-and-drone hybrid models for deliveries by means of a continuous approximation model, considering different model parameters and customer densities. The paper Mourelo Ferrandez et al. (2016) focuses on a delivery system where one truck supports the operations of multiple drones. The authors first clusterize the customers demand by adopting a K-means algorithm to find truck stops that represent hubs for drone deliveries. Then, they determine a TSP of the truck among centroids of these clusters, by means of a genetic algorithm, assuming that drones are not constrained by flight range. In Moshref-Javadi and Lee (2017) the authors consider a similar delivery system where at each stop site the truck waits until all drones come back before moving to the next site. The goal is the minimization of the latency in a customer-oriented distribution system. In Poikonen and Golden (2020) the authors formalize the k-Multi-visit Drone Routing Problem (k-MVDRP) considering a tandem between a truck and a fleet of k drones. The authors assumed that each drone can deliver one or more packages to customers in a single mission. Each drone may return to the truck to swap/recharge batteries, pick up a new set of packages, and depart again to customer locations. The article presents a mathematical formulation including a drone energy drain function that takes into account each package weight, but the problem is then solved by means of a heuristic algorithm. The paper Amorosi et al. (2020) presents a multi-objective mixed integer linear programming model for the management of a hybrid delivery system consisting in one base vehicle and a fleet of drones. The problem consists in determining the tour of the base vehicle and the assignments of the customers to the UAVs simultaneously optimizing the distance travelled by the vehicle, the one travelled by the drones and the maximum completion time. The model is solved on two realistic urban scenarios providing a partial exploration of the Pareto frontier of the problem by means of the weighted sum method.

Other examples of similar configurations in which a base vehicle supports the operations of one or multiple drones can be found also in other sectors. Trotta et al. (2018) studies, for example, the city-scale video monitoring of a set of points of interest performed by a fleet of UAVs whose operations are supported by buses. However, in all the works mentioned so far, the combined operations of vehicles and drones examine routing for a set of locations and these configurations exploit only part of the advantages of the use of drones. Indeed, UAVs can move between any two points in the space not following the road network. Thus, they can be used also for other kind of services in which the targets are represented by edges or part of them. This leads to another class of models, that is arc routing problems (ARPs).

Differently from NRPs, there are relatively few papers on ARP with drones. In Oh et al. (2011) and Oh et al. (2014) the authors study a coordinated road network search problem with multiple UAVs and they formulate it as a Multi-choice Multi-dimensional Knapsack Problem minimizing the flight time. This is a modified Chinese Postman Problem taking into account the UAVs energy capacity constraints. The authors solve the problem by means of a greedy insertion heuristic that models drone travel distance between the road components as a Dubins path. Dille and Singh (2013) faces the area coverage problem in sparse environments with multiple UAVs. Also in this case the UAVs motion is modeled using Dubins paths and it is assumed that they are equipped with a coverage sensor of a given radius. The edge covering problem is solved by discretizing the network in orbits and then solving a TSP among this set of orbits.

In Chow (2016) the authors deal with the problem of dynamically allocate a finite set of UAVs to links in a network that need monitoring, over multiple time periods. The need of drone monitoring is based on data related to traffic conditions. Another work related to multiple-period real-time monitoring of road traffic, adopting UAVs, is Li et al. (2018). The authors propose a mixed integer programming model combining a capacitated arc routing with an inventory routing problem and design a local branching based method to deal with large instances. The ARPs with one UAV, have common characteristics with problems arising in path generation for laser cutting machines or drawing plotters. In particular, the authors of Campbell et al. (2018) study the Drone Rural Postman Problem (DRPP) showing the relation with the Intermittent Cutting Problem (ICP). They present a solution algorithm based on the approximation of curves in the plane by polygonal chains that sequentially increases the number of points in the polygonal where the UAV can enter or leave. Thus, they solve the problem as a discrete optimization problem trying to better define the line by increasing the number of points.

As mentioned above, the number of references about ARPs with drones is limited as compared with the one on Drone NRPs. Moreover, as far as we know, the number of contributions in literature on arc routing problems involving a hybrid system consisting in one vehicle and one or multiple drones is simply reduced to Tokekar et al. (2016) and Garone et al. (2010).

In Tokekar et al. (2016) the authors study the path planning problem of a system composed by a ground robot and one drone in precision agriculture and solved it by applying orienteering algorithms. On the other hand, Garone et al. (2010) studies the paths planning problem for systems consisting in a carrier vehicle and a carried one to visit a set of target points and assuming that the carrier vehicle moves in the continuous space. Heuristic approaches have been proposed to deal with these problems.

Table 1
Nomenclature for AMDRPG.

Problem Parameters
orig: coordinates of the point defining the origin of the mothership path (or tour).
dest: coordinates of the point defining the destination of the mothership path (or tour).
$\mathscr{G}$: set of the target graphs.
$g = (V_g, E_g)$: set of nodes and edges of each target graph $g \in \mathscr{G}$.
$\mathscr{L}(e_g)$: length of edge e of graph $g \in \mathscr{G}$.
B^{e_g}, C^{e_g}: coordinates of the endpoints of edge e of graph $g \in \mathscr{G}$.
α^{e_g}: percentage of edge e of graph $g \in \mathscr{G}$ that must be visited.
α^g: percentage of graph $g \in \mathscr{G}$ that must be visited.
v_D: drone speed.
v_M: mothership speed.
M: big-M constant.

To the best of our knowledge none of the previous papers deals with a drone arc routing problem combined with a node routing problem for a hybrid system consisting of one mothership vehicle and one drone.

In this paper we present new Mixed Integer Non-Linear Programming (MINLP) formulations for the problem of coordinating a system composed by one mothership (the base vehicle) which supports the operations of one drone which has to visit a set of targets represented by graphs, minimizing the total distance travelled by both vehicles. Indeed, differently from the previous works in literature, we assume that the drone can move freely on the continuous space, whereas for the mothership, we study two cases: 1) the base vehicle can move freely on the continuous space; and 2) it is constrained to move on a road network where launching and rendezvous points must be chosen. The main contributions of this article can be summarized as follows:

- A formal analysis of the coordination problem of one base vehicle and one drone in the continuous space that combine several characteristics that have not been previously analyzed simultaneously;
- First mathematical formalization of the problem by means of new second order cone MINLP formulations;
- Development of a new matheuristic algorithm to deal with large instances able to provide high quality solutions in limited computing time;
- Extensive experimental analysis comparing the exact solution of the formulations and the matheuristic on a set of generated instances involving different typologies of planar graphs.

4. Mixed integer non linear programming formulations

In this section we present alternative MINLP formulations for the MDRPG depending on the nature of the mothership that will be compared computationally in later sections. We start analyzing first the situation where the mothership moves in a continuous space, namely the AMDRPG.

4.1. All terrain mothership-drone routing problem with graphs

In this problem, we assume that the mothership is allowed to move freely in a continuous space: $\mathbb{R}^2$ or $\mathbb{R}^3$. In addition, distances are measured by the Euclidean norm, $\|\cdot\|_2$, although this assumption can be extended to any l_p norm, $1 \leqslant p \leqslant \infty$ (see Blanco et al. (2017)).

Our goal is to develop an exact mathematical programming formulation that can be used to solve instances of this problem. In the following, we introduce the elements that formally describe the problem and that are summarized in Table 1.

Let $g = (V_g, E_g)$ be a graph in $\mathscr{G}$, where V_g denotes the set of nodes and E_g denotes the set of edges connecting pairs of nodes. Let e_g be edge e of the graph $_g \in G$ and let $\mathscr{L}(e_g)$ be its length. Each edge e_g is parametrized by its endpoints $B^{e_g} = (B^{e_g}(x_1), B^{e_g}(x_2))$ and $C^{e_g} = (C^{e_g}(x_1), C^{e_g}(x_2))$ and we indicate its length $\mathscr{L}(e_g) = \left\|C^{e_g} - B^{e_g}\right\|$. Moreover, we denote with $\mathscr{L}(g) = \sum_{e_g \in E_g} \mathscr{L}(e_g)$ the total length of graph g.

For each edge e_g, with length $\mathscr{L}(e_g)$, we assign a binary variable μ^{e_g} that indicates whether or not the drone visits the segment e_g and define entry and exit points $R^{e_g} = (B^{e_g}, C^{e_g}, \rho^{e_g})$ and $L^{e_g} = (B^{e_g}, C^{e_g}, \lambda^{e_g})$, respectively, that determine the portion of the edge visited by the drone. Indeed, the coordinates of the points R^{e_g} and L^{e_g} are given, respectively by

$$R^{e_g} = (\rho^{e_g} B^{e_g}(x_1) + (1 - \rho^{e_g}) C^{e_g}(x_1), \rho^{e_g} B^{e_g}(x_2) + (1 - \rho^{e_g}) C^{e_g}(x_2))$$

and

$$L^{e_g} = (\lambda^{e_g} B^{e_g}(x_1) + (1 - \lambda^{e_g}) C^{e_g}(x_1), \lambda^{e_g} B^{e_g}(x_2) + (1 - \lambda^{e_g}) C^{e_g}(x_2)),$$

where $\rho^{e_g} \in [0, 1]$ and $\lambda^{e_g} \in [0, 1]$ are variables to determine the position of the points on the segment.

As mentioned in Section 2, we have considered two modes of visit to the target graphs $g \in \mathscr{G}$ that must be represented by their corresponding constraints:

- Visiting a percentage α^{e_g} of each edge e_g which can be modeled by:

$$|\lambda^{e_g} - \rho^{e_g}| \mu^{e_g} \geqslant \alpha^{e_g}, \quad \forall e_g \in E_g. \qquad (\alpha\text{-E})$$

- Visiting a percentage α^g of the total length $\mathscr{L}(g)$ of the graph g modeled by:

$$\sum_{e_g \in E_g} \mu^{e_g} |\lambda^{e_g} - \rho^{e_g}| \mathscr{L}\left(e_g\right) \geqslant \alpha^g \mathscr{L}\left(g\right) \qquad \left(\alpha\text{-G}\right)$$

In both cases the corresponding constraints are nonlinear. In order to linearize them, we need to introduce a binary variable entry^{e_g} that determines the traveling direction on the edge e_g as well as the definition of the parameter values $\nu_{\min}^{e_g}$ and $\nu_{\max}^{e_g}$ of the access and exit points to that segment. Then, for each edge e_g, the absolute value constraint (α-E) can

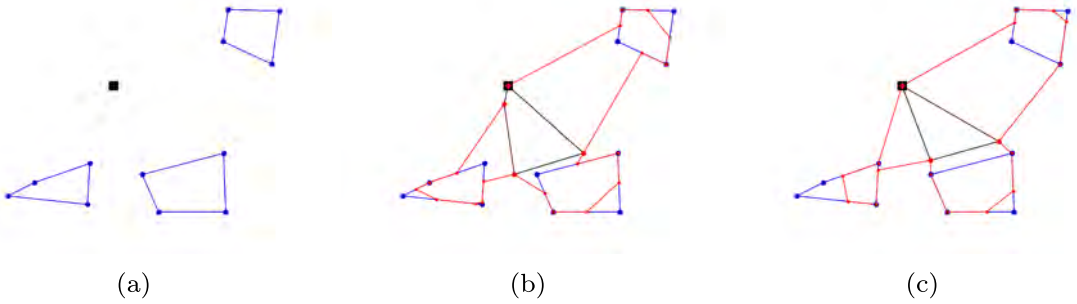


Fig. 2. (a) Origin and target graphs, (b) Visit of α^{e_g}% of each edge, (c) Visit of α_g% of each graph.

Table 2
Decision Variables for AMDRPG-ST.

Binary and Integer Decision Variables
$\mu^{e_g} \in \{0,1\}$, $\forall e_g \in E_g$ $(g \in \mathscr{G})$: equal to 1 if edge e of graph g (or a portion of it) is visited by the drone, and 0 otherwise.
$\text{entry}^{e_g} \in \{0,1\}$, $\forall e_g \in E_g$ $(g \in \mathscr{G})$: auxiliary binary variable used for linearizing expressions.
$u^{e_g t} \in \{0,1\}$, $\forall e_g \in E_g$ $(g \in \mathscr{G})$, $\forall t \in T$: equal to 1 if the drone enters in graph g by e_g at stage t, 0 otherwise.
$z^{e_g e_g'} \in \{0,1\}$, $\forall e_g, e_g' \in E_g$ $(g \in \mathscr{G})$: equal to 1 if the drone goes from e_g to e_g', 0 otherwise.
$v^{e_g t} \in \{0,1\}$, $\forall e_g \in E_g$ $(g \in \mathscr{G})$, $\forall t \in T$: equal to 1 if the drone exits from graph g by e_g at stage t, 0 otherwise.
s^{e_g}, $\forall e_g \in E_g$ $(g \in \mathscr{G})$: integer non negative variable representing the order of visit of edge e of graph g.
Continuous Decision Variables
$\rho^{e_g} \in [0,1]$ and $\lambda^{e_g} \in [0,1]$, $\forall e_g \in E_g$ $(g \in \mathscr{G})$: defining the entry and exit points on e_g.
$\nu_{\min}^{e_g}$ and $\nu_{\max}^{e_g} \in [0,1]$, $\forall e_g \in E_g$ $(g \in \mathscr{G})$: auxiliary variables used for linearizing expressions.
x_L^t, $\forall t \in T$: coordinates representing the point where the mothership launches the drone at stage t.
x_R^t, $\forall t \in T$: coordinates representing the point where the mothership retrieves the drone at stage t.
R^{e_g}, $\forall e_g \in E_g$ $(g \in \mathscr{G})$: coordinates representing the entry point on edge e of graph g.
L^{e_g}, $\forall e_g \in E_g$ $(g \in \mathscr{G})$: coordinates representing the exit point on edge e of graph g.
$d_L^{e_g t} \geqslant 0$, $\forall e_g \in E_g$ $(g \in \mathscr{G})$, $\forall t \in T$: representing the distance travelled by the drone from the launching point x_L^t on the mothership at stage t to the first visiting point R^{e_g} on e_g.
$d^{e_g e_g'} \geqslant 0$, $\forall e_g, e_g' \in E_g (g \in \mathscr{G})$: representing the distance travelled by the drone from the launching point L^{e_g} on e_g to the rendezvous point $R^{e_g'}$ on e_g'.
$d^{e_g} \geqslant 0$, $\forall e_g \in E_g$ $(g \in \mathscr{G})$: representing the distance travelled by the drone from the rendezvous point R^{e_g} to the launching point L^{e_g} on e_g.
$d_R^{e_g t} \geqslant 0$, $\forall e_g \in E_g$ $(g \in \mathscr{G})$, $\forall t \in T$: representing the distance travelled by the drone from the last visiting point L^{e_g} on e_g to the rendezvous point x_R^t on the mothership at stage t.
$d_{LR}^t \geqslant 0$, $\forall t \in T$: representing the distance travelled by the mothership from the launching point x_L^t to the rendezvous point x_R^t at stage t.
$d_{RL}^t \geqslant 0$, $\forall t \in T$: representing the distance travelled by the mothership from the rendezvous point x_R^t at stage t to the launching point $x_L^{(t+1)}$ at the stage $t+1$.

be represented by:

$$\mu^{e_g}|\rho^{e_g} - \lambda^{e_g}| \geqslant \alpha^{e_g} \Leftrightarrow \begin{cases} \rho^{e_g} - \lambda^{e_g} & = & \nu_{\max}^{e_g} - \nu_{\min}^{e_g}, \\ \nu_{\max}^{e_g} & \leqslant & 1 - \text{entry}^{e_g}, \\ \nu_{\min}^{e_g} & \leqslant & \text{entry}^{e_g}, \\ \mu^{e_g}\left(\nu_{\max}^{e_g} + \nu_{\min}^{e_g}\right) & \geqslant & \alpha^{e_g}. \end{cases} \qquad (\alpha\text{-E})$$

The linearization of $(\alpha$-G) is similar to $(\alpha$-E) and only requires changing the last inequality in $(\alpha$-E)) for

$$\sum_{e_g \in E_g} \mu^{e_g}\left(\nu_{\max}^{e_g} + \nu_{\min}^{e_g}\right)\mathscr{L}\left(e_g\right) \geqslant \alpha_g \mathscr{L}\left(g\right) \qquad \left(\alpha\text{-G}\right)$$

Fig. 2(a) shows an example of a configuration with three target graphs, each one with four nodes and four edges. The mothership begins at its starting point, denoted by the black square and it follows the black tour, like the ones in Figs. 2(b) and 2(c), where the red points represent launching and rendezvous points for the drone. Figs. 2(b) and 2(c) show, through red segments, respectively the visit of a percentage α^{e_g} of each edge of the target graphs and the visit of a percentage α_g of each target graph. In this latter case we can observe that for each target graph one edge is not visited by the drone.

4.1.1. A first formulation for AMDRPG based on stages

Our first proposal to model this problem uses stages identified with the order in which the different elements in the problem are visited. We identify each visit to one of the target graphs with a stage of the process. Then, by using the notation below, we define the stages associated with graphs $T := \{1, \ldots, |\mathscr{G}|\}$ and those associated with the launching and rendezvous points including *orig* and *dest* $T' = \{0, \ldots, |\mathscr{G}| + 1\}$. For each stage $t \in T$, the drone follows a path starting from the mothership, visiting the required edges of g and returning to the mothership. Using the notation in Table 2 the sequence of points in the path are the following:

$$x_L^t \rightarrow R^{e_g} \rightarrow L^{e_g} \rightarrow \cdots \rightarrow R^{e_g'} \rightarrow L^{e_g'} \rightarrow \cdots \rightarrow R^{e_g''} \rightarrow x_R^t \rightarrow x_L^{t+1}.$$

This path calls in a natural way for the definition of binary variables that choose:

- The optimal order to visit each graph $g \in \mathscr{G}$. In other words, defining the sequence of the stages.
- The optimal order to visit the edges of each graph in its corresponding stage.

Thus, we can model the route that the drone follows by using the binary variables $u^{e_g t}$, $z^{e_g e_g'}$ and $v^{e_g t}$ defined in Table 2.

$$\sum_{g \in \mathscr{G}} \sum_{e_g \in E_g} u^{e_g t} = 1, \qquad \forall t \in T, \qquad (1)$$

$$\sum_{g \in \mathscr{G}} \sum_{e_g \in E_g} v^{e_g t} = 1, \qquad \forall t \in T, \qquad (2)$$

$$\sum_{e_g \in E_g} \sum_{t \in T} u^{e_g t} = 1, \qquad \forall g \in \mathscr{G}, \qquad (3)$$

$$\sum_{e_g \in E_g} \sum_{t \in T} v^{e_g t} = 1, \qquad \forall g \in \mathscr{G}, \qquad (4)$$

$$\sum_{e_g \in E_g} u^{e_g t} = \sum_{e_g \in E_g} v^{e_g t}, \qquad \forall g \in \mathscr{G}, \forall t \in T, \qquad (5)$$

$$\sum_{e_g' \in E_g} z_g^{e_g' e_g} + \sum_{t \in T} u^{e_g t} = \mu^{e_g}, \qquad \forall e_g \in E_g : g \in \mathscr{G}, \qquad (6)$$

$$\sum_{e_g' \in E_g} z_g^{e_g e_g'} + \sum_{t \in T} v^{e_g t} = \mu^{e_g}, \qquad \forall e_g \in E_g : g \in \mathscr{G}. \qquad (7)$$

Equations (1) and (2) state that in each stage the drone visits (enter and exit, respectively) only one graph. Constraints (3) and (4) assure that each graph is visited at some stage. Constraints (5) enforce that at stage t the drone enters and exits of exactly the same graph. Constraints (6) state that if edge e of graph g is visited by the drone, one of two alternative situations must occur: either e is the first edge of graph g visited by the drone at stage t, or edge e is visited by the drone after visiting another edge e' of graph g. Similarly, constraints (7) state that if edge e of graph g is visited by the drone, either e is the last edge of graph g visited by the drone at stage t, or the drone must move to another edge e' of graph g after visiting edge e.

4.1.2. Elimination of subtours

To prevent the existence of subtours within each graph $g \in \mathscr{G}$ that the drone must visit, one can include, among others, either a formulation that uses the Miller-Tucker-Zemlin constraints (MTZ) or another one that applies the subtour elimination constraints (SEC).

For the MTZ formulation, we use the continuous variables s^{e_g}, defined in Table 2, and establish the following constraints for each $g \in \mathscr{G}$:

Table 3
Decision Variables for AMDRPG-MTZ.

Binary and Integer Decision Variables
$\mu^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G})$: equal to 1 if edge e of graph g (or a portion of it) is visited by the drone, and 0 otherwise.
$\text{entry}^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G})$: auxiliary binary variables for linearization.
$u^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G})$: equal to 1 if the drone enters in graph g by e_g, 0 otherwise.
$z^{e_g e_g'} \in \{0,1\},\ \forall e_g, e_g' \in E_g\ (g \in \mathcal{G})$: equal to 1 if the drone goes from e_g to e_g', 0 otherwise.
$v^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G})$: equal to 1 if the drone exits from graph g by e_g, 0 otherwise.
$w^{gg'} \in \{0,1\},\ \forall g,g' \in \mathcal{G}$: equal to 1 if the mothership moves from x_R^g to $x_L^{g'}$, 0 otherwise.
$s^{e_g},\ \forall e_g \in E_g\ (g \in \mathcal{G})$: integer non negative variables representing the order of visit of edge e of graph g.
Continuous Decision Variables
$\rho^{e_g} \in [0,1]$ and $\lambda^{e_g} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathcal{G})$: defining the entry and exit points on e_g.
$\nu_{\min}^{e_g}$ and $\nu_{\max}^{e_g} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathcal{G})$: auxiliary variables for linearization.
$x_L^g,\ \forall g \in \mathcal{G}$: pairs of coordinates representing the point where the mothership launches the drone to visit graph g.
$x_R^g,\ \forall g \in \mathcal{G}$: pairs of coordinates representing the point where the mothership retrieves the drone after visit graph g.
$R^{e_g},\ \forall e_g \in E_g\ (g \in \mathcal{G})$: coordinates representing the entry point on edge e of graph g.
$L^{e_g},\ \forall e_g \in E_g\ (g \in \mathcal{G})$: coordinates representing the exit point on edge e of graph g.
$d_L^{e_g} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathcal{G})$: representing the distance travelled by the drone from the launching point on the mothership x_L^g to the first visiting point R^{e_g} on edge e_g.
$d^{e_g e_g'} \geqslant 0,\ \forall e_g, e_g' \in E_g (g \in \mathcal{G})$: representing the distance travelled by the drone from launching point L^{e_g} on e_g to the rendezvous point $R^{e_g'}$ on e_g'.
$d^{e_g} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathcal{G})$: representing the distance travelled by the drone from the rendezvous point R^{e_g} to the launching point L^{e_g} on e_g.
$d_R^{e_g} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathcal{G})$: representing the distance travelled by the drone from the last visiting point L^{e_g} on e_g to the rendezvous point x_R^g on the mothership.
$d_{LR}^g \geqslant 0,\ \forall g \in \mathcal{G}$: representing the distance travelled by the mothership from the launching point x_L^g to the rendezvous point x_R^g while the drone is visiting g.
$d_{RL}^{gg'} \geqslant 0,\ \forall g,g' \in \mathcal{G}$: representing the distance travelled by the mothership from the rendezvous point x_R^g for graph g to the launching point $x_L^{g'}$ for graph g'.

$$s^{e_g} - s^{e_g'} + |E_g| z^{e_g e_g'} \leqslant |E_g| - 1, \quad \forall e_g \neq e_g' \in E_g, \qquad (\text{MTZ}_1)$$

$$0 \leqslant s^{e_g} \leqslant |E_g| - 1, \quad \forall e_g \in E_g. \qquad (\text{MTZ}_2)$$

Alternatively, we can also use the family of subtour elimination constraints:

$$\sum_{e_g, e_g' \in S} z_g^{e_g e_g'} \leqslant |S| - 1, \quad \forall S \subset E_g : g \in \mathcal{G}. \qquad (\text{SEC})$$

Since there is an exponential number of SEC constraints, when we implement this formulation we need to perform a row generation procedure including constraints whenever they are required by a separation oracle. To find SEC inequalities, as usual, we search for disconnected components in the current solution. Among them, we choose the shortest subtour found in the solution to be added as a lazy constraint to the model.

The goal of the AMDRPG is to find a feasible solution that minimizes the total distance traveled by the drone and the mothership. To account for the different distances among the decision variables of the model we need to set the continuous variables $d_L^{e_g t}$, d^{e_g}, $d^{e_g e_g'}$, $d_R^{e_g t}$, d_{RL}^t and d_{LR}^t, defined in Table 2. This can be done by means of the following constraints:

$$\begin{aligned}
\|x_L^t - R^{e_g}\| &\leqslant d_L^{e_g t}, && \forall e_g \in E_g : g \in \mathcal{G}, \forall t \in T && (\text{DIST}_{1\text{-t}})\\
\|R^{e_g} - L^{e_g}\| &\leqslant d^{e_g}, && \forall e_g \in E_g : g \in \mathcal{G}, \forall t \in T, && (\text{DIST}_{2\text{-t}})\\
\|R^{e_g} - L^{e_g'}\| &\leqslant d^{e_g e_g'}, && \forall e_g \neq e_g' \in E_g : g \in \mathcal{G}, && (\text{DIST}_{3\text{-t}})\\
\|L^{e_g} - x_R^t\| &\leqslant d_R^{e_g t}, && \forall e_g \in E_g : g \in \mathcal{G}, \forall t \in T, && (\text{DIST}_{4\text{-t}})\\
\|x_R^t - x_L^{t+1}\| &\leqslant d_{RL}^t, && \forall t \in T, && (\text{DIST}_{5\text{-t}})\\
\|x_L^t - x_R^t\| &\leqslant d_{LR}^t, && \forall t \in T. && (\text{DIST}_{6\text{-t}})
\end{aligned}$$

To ensure that the time spent by the drone to visit graph g at stage t is less than or equal to the time that the mothership needs to move from the launching point to the rendezvous point at stage t, we need to define the following coordination constraint for each $g \in \mathcal{G}$ and $t \in T$:

$$\frac{1}{v_D}\left(\sum_{e_g \in E_g} u^{e_g t} d_L^{e_g t} + \sum_{e_g, e_g' \in E_g} z^{e_g e_g'} d^{e_g e_g'} + \sum_{e_g \in E_g} \mu^{e_g} d^{e_g} + \sum_{e_g \in E_g} v^{e_g t} d_R^{e_g t}\right) \leqslant \frac{d_{RL}^t}{v_M} + M\left(1 - \sum_{e_g \in E_g} u^{e_g t}\right) \qquad (\text{DCW-t})$$

Eventually, we have to impose that the tour of the mothership, together with the drone, starts from the origin *orig* and ends at the destination *dest*. To this end, we define the following constraints:

$$\begin{aligned}
x_L^0 &= orig, && (\text{ORIG}_1)\\
x_R^0 &= orig, && (\text{ORIG}_2)\\
x_L^{|\mathcal{G}|+1} &= dest, && (\text{DEST}_1)\\
x_R^{|\mathcal{G}|+1} &= dest. && (\text{DEST}_2)
\end{aligned}$$

Therefore, putting together all the constraints introduced before, the following formulation minimizes the overall distance traveled by the mothership and drone, coordinating their movements and ensuring the required coverage of the targets.

$$\sum_{g \in \mathcal{G}} \sum_{e_g \in E_g} \sum_{t \in T} \left(u^{e_g t} d_L^{e_g t} + v^{e_g t} d_R^{e_g t}\right) + \sum_{g \in \mathcal{G}} \sum_{e_g \in E_g} \mu^{e_g} d^{e_g} + \sum_{g \in \mathcal{G}} \sum_{e_g, e_g' \in E_g} z^{e_g e_g'} d^{e_g e_g'}$$

$$+ \sum_{t \in T} \left(d_{RL}^t + d_{LR}^t\right) \qquad (AMDRPG - ST)$$

s.t. (1)–(7)
(MTZ_1)–(MTZ_2) or (SEC)
(α−E) or (α-G)
(DCW-t)
($\text{DIST}_{1\text{-t}}$)–($\text{DIST}_{6\text{-t}}$).
(ORIG_1)–(DEST_2).

The objective function has four terms: the first one computes the distances to go to and to return from the mothership to the target graph g visited at stage t, the second one accounts for the distance traveled by the drone on the edges of the target graph g between two consecutive *jumps* between edges, the third one computes the distances traveled by the drone while it *jumps* between two consecutive edges on the target graph g and the fourth term expresses the distances made by the mothership. Constraints (1)–(7) model the route followed by the drone, (MTZ_1)-(MTZ_2) or (SEC) ensure that the displacement of the drone in each visit to a target graph is a route, (α-E) or (α-G) define what is required in each visit to a target graph. Constraints ($\text{DIST}_{1-\text{t}}$)-($\text{DIST}_{6-\text{t}}$) set the variables $d_L^{e_g t}$, d^{e_g}, $d^{e_g e_g'}$, $d_R^{e_g t}$, d_{RL}^t and d_{LR}^t, defined in Table 2, which represent Euclidean distances needed in the model. Note that, to deal with the bilinear terms of the objective function, we use McCormick's envelopes to linearize these terms by adding variables $p \geqslant 0$ that represent these products, and by introducing the following constraints:

$$p \geqslant mz, \qquad (8)$$

$$p \leqslant d - M(1-z), \tag{9}$$

where m and M are, respectively, the lower and upper bounds of the distance variable d. These bounds will be adjusted for each bilinear term in Section 4.3.

4.1.3. Alternative formulations for AMDRPG based on enforcing connectivity: MTZ-constraints and subtour elimination-constraints (SEC)

In this family of formulations we replace the variables u^t, v^t and constraints that model the tour using stages, namely (1)–(7), by constraints that ensure connectivity. We will distinguish two different approaches. The first one is based on the rationale of Miller-Tucker-Zemlin (MTZ) whereas the second one uses an extended formulation based on SEC.

We start presenting the formulation based on MTZ. It requires the definition of two families of binary variables that choose:

- The optimal order to visit the graphs $g \in \mathcal{G}$ in the tour of the mothership.
- The optimal order to visit the edges of each graph.

The above concepts can be modeled with the binary variables u^{e_g}, $z^{e_g e_g'}$, v^{e_g} and $w^{gg'}$, summarized in Table 3.

By using these binary variables, we can model the route that the drone follows in each particular graph $g \in \mathcal{G}$:

$$\sum_{e_g \in E_g} u^{e_g} = 1, \quad \forall g \in \mathcal{G}, \tag{10}$$

$$\sum_{e_g \in E_g} v^{e_g} = 1, \quad \forall g \in \mathcal{G}, \tag{11}$$

$$\sum_{e_g' \in E_g} z^{e_g' e_g} + u^{e_g} = \mu^{e_g}, \quad \forall e_g \in E_g : g \in \mathcal{G}, \tag{12}$$

$$\sum_{e_g' \in E_g} z^{e_g e_g'} + v^{e_g} = \mu^{e_g}, \quad \forall e_g \in E_g : g \in \mathcal{G}. \tag{13}$$

On the other hand, to model the tour followed by the mothership, we have to include the following new constraints:

$$\sum_{g \in \mathcal{G}} w^{g0} = 0, \tag{14}$$

$$\sum_{g' \in \mathcal{G}} w^{(n_G+1)g'} = 0, \tag{15}$$

$$\sum_{g' \in \mathcal{G} \setminus \{g\}} w^{gg'} = 1, \quad \forall g \in \mathcal{G}, \tag{16}$$

$$\sum_{g \in \mathcal{G} \setminus \{g'\}} w^{gg'} = 1, \quad \forall g' \in \mathcal{G}, \tag{17}$$

$$\begin{array}{llll} s_g - s_{g'} + |\mathcal{G}| w^{gg'} & \leqslant |\mathcal{G}| - 1, & \forall g \neq g' \in \mathcal{G}, & (\mathrm{MTZ}_3) \\ 0 & \leqslant s_g \leqslant |\mathcal{G}| - 1, & \forall g \in \mathcal{G}, & (\mathrm{MTZ}_4) \\ s_0 & = 0, & & (\mathrm{MTZ}_5) \\ s_{n_G+1} & = n_G + 1. & & (\mathrm{MTZ}_6) \end{array}$$

The reader may note that constraints (16), (17) and (MTZ_3)–(MTZ_6) are of the type MTZ for the complete graph $\widehat{G} = (\widehat{N}, \widehat{E})$ induced by the graphs $g \in \mathcal{G}$ to be visited, where a node $\widehat{v}_g \in \widehat{N}$ if $g \in \mathcal{G}$. Thus, we enforce that the mothership route follows a tour on the graphs to be visited by the drone. Next, we have to connect the visits of the drone to the graphs with the tour followed by the mothership. To this end, we require to introduce variables defining the launching and rendezvous points to account for the distances traveled and to define the inner tour of the mothership. In a similar way, we need to set the continuous variables $d_L^{e_g}$, $d^{e_g e_g'}$, d^{e_g}, $d_R^{e_g}$, d_{LR}^{g} and $d_{RL}^{gg'}$, defined in Table 3, that account for the distances among distinguished points. Thus, we introduce the following constraints:

$$\begin{array}{llll} \|x_L^g - R^{e_g}\| & \leqslant d_L^{e_g}, & \forall e_g \in E_g : g \in \mathcal{G}, & (\mathrm{DIST}_{1\text{-g}}) \\ \|R^{e_g} - L^{e_g}\| & \leqslant d^{e_g}, & \forall e_g \in E_g : g \in \mathcal{G}, & (\mathrm{DIST}_{2\text{-g}}) \\ \|R^{e_g} - L^{e_g'}\| & \leqslant d^{e_g e_g'}, & \forall e_g \neq e_g' \in E_g : g \in \mathcal{G}, & (\mathrm{DIST}_{3\text{-g}}) \\ \|L^{e_g} - x_R^g\| & \leqslant d_R^{e_g}, & \forall e_g \in E_g : g \in \mathcal{G}, & (\mathrm{DIST}_{4\text{-g}}) \\ \|x_R^g - x_L^{g'}\| & \leqslant d_{RL}^{gg'}, & \forall g, g' \in \mathcal{G}, & (\mathrm{DIST}_{5\text{-g}}) \\ \|x_L^g - x_R^g\| & \leqslant d_{LR}^{g}, & \forall g \in \mathcal{G}. & (\mathrm{DIST}_{6\text{-g}}) \end{array}$$

In this formulation of AMDRPG the goal is again to find a feasible solution that minimizes the total distance traveled by the drone and the mothership but without using stages. Instead the model ensures the Hamiltonian nature of the routes using the rationale of connectivity by the MTZ or SEC family of constraints.

Again, we need to be sure that the time spent by the drone to visit the graph g is less than or equal to the time that the mothership needs to move from the launching point to the rendezvous point associated to this graph g. Hence, by using the same argument, as the one used in (DCW-t), we define for each $g \in \mathcal{G}$:

$$\frac{1}{v_D}\left(\sum_{e_g \in E_g} u^{e_g} d_L^{e_g} + \sum_{e_g, e_g' \in E_g} z^{e_g e_g'} d^{e_g e_g'} + \sum_{e_g \in E_g} \mu^{e_g} d^{e_g} + \sum_{e_g \in E_g} v^{e_g} d_R^{e_g}\right) \leqslant \frac{d_{RL}^g}{v_M}, \quad \forall g \in \mathcal{G}. \qquad \left(\text{DCW-g}\right)$$

Therefore, modifying the previous formulation (AMDRPG-ST), replacing constraints based on stages, one obtains the following alternative valid formulation:

$$\sum_{g \in \mathcal{G}} \sum_{e_g \in E_g} \left(u^{e_g} d_L^{e_g} + v^{e_g} d_R^{e_g}\right) + \sum_{g \in \mathcal{G}} \sum_{e_g \in E_g} \mu^{e_g} d^{e_g} + \sum_{g \in \mathcal{G}} \sum_{e_g, e_g' \in E_g} z^{e_g e_g'} d^{e_g e_g'} + \sum_{g \in \mathcal{G}} d_{LR}^g + \sum_{g, g' \in \mathcal{G}} d_{RL}^{gg'} w^{gg'} \qquad \left(\text{AMDRPG-MTZ}\right)$$

s.t. (10)–(17)
(MTZ_1)–(MTZ_2) or (SEC)
(MTZ_3)–(MTZ_6)
(α−E) or (α-G)
(DCW-g)
($\mathrm{DIST}_{1\text{-g}}$)–($\mathrm{DIST}_{6\text{-g}}$).
(ORIG_1)–(ORIG_2)The formulation above (AMDRPG-MTZ) can be slightly modified replacing constraints (MTZ_3)-(MTZ_6) by

$$\sum_{g, g' \in \mathcal{G}} w^{gg'} \leqslant |S| - 1, \quad \forall S \subseteq \left\{1, \ldots, |\mathcal{G}|\right\}. \tag{18}$$

In doing so we obtain another formulation for AMDRPG using SEC rather than MTZ constraints to represent the mothership tour. We will call this formulation (AMDRPG-SEC). Later, we will compare their

Table 4
Nomenclature for PMDRPG.

Problem Parameters
$V^1, \ldots, V^p$: pairs of coordinates defining the breakpoints of the polygonal chain $\mathscr{P}$ representing the road system where the mothership can move.
orig: pair of coordinates defining the origin of the mothership path (or tour).
dest: pair of coordinates defining the destination of the mothership path (or tour).
$\mathscr{G}$: set of the target graphs.
$g = (V_g, E_g)$: set of nodes and edges of each target graph $g \in \mathscr{G}$.
$\mathscr{L}(e_g)$: length of edge e of graph $g \in \mathscr{G}$.
B^{e_g}, C^{e_g}: endpoints of edge e of graph $g \in \mathscr{G}$.
α^{e_g}: percentage of edge e of graph $g \in \mathscr{G}$ that must be visited.
α^g: percentage of graph $g \in \mathscr{G}$ that must be visited.
v_D: drone speed.
v_M: mothership speed.
M: big M.

performance in Section 6.

4.1.4. Comparing the formulations

A straightforward analysis of the two types of formulations, namely by stages and based on connectivity, shows that (AMDRPG-MTZ) has much less variables and constraints than (AMDRPG-ST). Indeed, the latter requires definition of distance variables for each e_g and t which is a quadratic function rather than a linear one as used by the former. In addition, to enforce coordination (AMDRPG-ST) needs constraints (DCW) for each $g \in \mathscr{G},\ t \in T$ whereas (AMDRPG-MTZ) only uses (DCW) constraints for $g \in \mathscr{G}$. This is an important reduction in the dimension of the problems to be solved. This observation is confirmed by its better performance as discussed in Section 6.

4.2. Network Mothership-Drone Routing Problem on a Polygonal with Graphs (PMDRPG)

The most significant difference between AMDRPG and NMDRPG is that, in the latter, the mothership cannot move freely in the framework space and instead it has to move on a network that models a road system where the base vehicle (for example a truck) is constrained to move. An intermediate situation between moving freely and moving on a general road network is to assume that the mothership moves on a closed polygonal, namely a piecewise linear, chain. Modeling this situation is a transition step to achieve the more general case of the mothership moving on general networks. We will present the formulations for the PMDRPG following a scheme similar to that of AMDRPG. First, we present a model by stages and then another one based on connectivity constraints.

Let $\mathscr{P}$ denote a polygonal chain parametrized by its breakpoints $V^1, \ldots, V^p$ ordered in the direction of travel. Observe that we are assuming that there exists a pre-specified orientation that determines the direction of the mothership's displacement.

As in the previous section, we present a summary of the notation, parameters and variables of this model in Tables 4 and 5.

We begin giving a formal description of the distance traveled by the mothership on the polygonal $\mathscr{P}$, between two consecutive events induced by drone operations, either launching-retrieving or retrieving-launching. To this end, we use the variables γ_L^{et} and γ_R^{et}, defined in Table 5. Recall that γ_L^{et} and γ_R^{et} are the values of the parametrizations that define the points x_L^t and x_R^t on the edge e of $\mathscr{P}$, respectively. Fig. 3 illustrates the meaning of the parameters γ_L^{et} and γ_R^{et} on a polygonal chain with 4 pieces.

Table 5
Decision Variables for PMDRPG-ST.

Binary and Integer Decision Variables
$\mu^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: equal to 1 if edge e of graph g (or a portion of it) is visited by the drone, 0 otherwise.
$\text{entry}^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: auxiliary binary variables for linearization.
$u^{e_g t} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathscr{G}),\ \forall t \in T$: equal to 1 if the drone enters in graph g by e_g at stage t, 0 otherwise.
$z^{e_g e_g'} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: equal to 1 if the drone goes from e_g to e_g' of graph g, 0 otherwise.
$v^{e_g t} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathscr{G}),\ \forall t \in T$: equal to 1 if the drone exits from graph g by e_g at stage t, 0 otherwise.
$s^{e_g},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: integer non negative variables representing the order of visit of edge e of graph g.
$z_{LR}^{ee' t} \in \{0,1\},\ \forall e,e' \in \mathscr{P},\ \forall t \in T$: equal to 1 if the launching point x_L^t is located on e and the rendezvous point x_R^t is located on e' at stage t.
$z_{RL}^{ee' t} \in \{0,1\},\ \forall e,e' \in \mathscr{P},\ \forall t \in T$: equal to 1 if the rendezvous point x_R^t is located on e at stage t and the launching point $x_L^{(t+1)}$ is located on e' at stage $t+1$.
$\mu_L^{et} \in \{0,1\},\ \forall e \in \mathscr{P},\ \forall t \in T$: equal to 1 if the launching point x_L^t is located on e at stage t.
$\mu_R^{et} \in \{0,1\},\ \forall e \in \mathscr{P},\ \forall t \in T$: equal to 1 if the rendezvous point x_R^t is located on e at stage t.
Continuous Decision Variables
$\rho^{e_g} \in [0,1]$ and $\lambda^{e_g} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathscr{G})$: defining the entry and exit points on e_g.
$\gamma_R^{et} \in [0,1]$ and $\gamma_L^{et} \in [0,1],\ \forall e \in \mathscr{P},\ \forall t \in T$: defining the launching and rendezvous points on edge e of the polygonal $\mathscr{P}$ at stage t.
$\nu_{\min}^{e_g}$ and $\nu_{\max}^{e_g} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathscr{G})$: auxiliary variables for linearization.
$x_L^t,\ \forall t \in T$: coordinates representing the point where the mothership launches the drone at stage t.
$x_R^t,\ \forall t \in T$: coordinates representing the point where the mothership retrieves the drone at stage t.
$R^{e_g},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: coordinates representing the entry point on edge e of graph g.
$L^{e_g},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: coordinates representing the exit point on edge e of graph g.
$d_L^{e_g t} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathscr{G}),\ \forall t \in T$: representing the distance travelled by the drone from the launching point x_L^t on the mothership at stage t to the first visiting point R^{e_g} on e_g.
$d^{e_g e_g'} \geqslant 0,\ \forall e_g, e_g' \in E_g (g \in \mathscr{G})$: representing the distance travelled by the drone from launching point L^{e_g} on e_g to the rendezvous point $R^{e_g'}$ on e_g'.
$d^{e_g} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathscr{G})$: representing the distance travelled by the drone from the rendezvous point R^{e_g} to the launching point L^{e_g} on e_g.
$d_R^{e_g t} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathscr{G}),\ \forall t \in T$: representing the distance travelled by the drone from the last visiting point L^{e_g} on e_g to the rendezvous point x_R^t on the mothership at stage t.
$d_{LR}^{ee' t} \geqslant 0,\ \forall e,e' \in \mathscr{P},\ \forall t \in T$: representing the distance travelled by the mothership from the launching point x_L^t on e to the rendezvous point x_R^t on e' at stage t.
$d_{RL}^{ee' t} \geqslant 0,\ \forall e,e' \in \mathscr{P},\ \forall t \in T$: representing the distance travelled by the mothership from the rendezvous point x_R^t on e at stage t to the launching point $x_L^{(t+1)}$ on e' at the stage $t+1$.
$d_{LR}^t \geqslant 0,\ \forall t \in T$: representing the total distance travelled by the mothership between the launching point x_L^t and the rendezvous point x_R^t at stage t.
$d_{RL}^t \geqslant 0,\ \forall t \in T$: representing the total distance travelled by the mothership between the rendezvous point x_R^t at stage t and the launching point $x_L^{(t+1)}$ at stage $t+1$.
$\lambda_L^t \geqslant 0,\ \forall t \in T$: representing the absolute position of point x_L^t in the polygonal $\mathscr{P}$.
$\lambda_R^t \geqslant 0,\ \forall t \in T$: representing the absolute position of point x_R^t in the polygonal $\mathscr{P}$.

Then, the distance measured on the polygonal can be obtained by the following expressions for each $t \in T$ and for each $e, e' \in \mathscr{P}$:

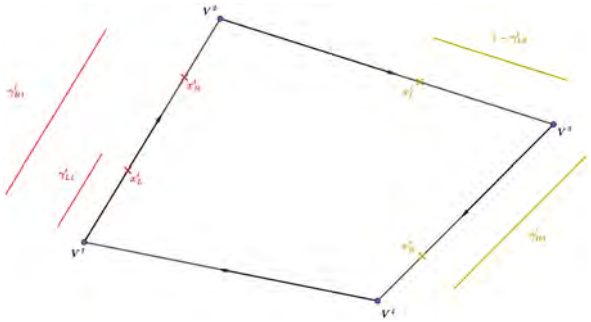


Fig. 3. An example of a parametrization of a mothership route that goes from V^1 to x_L^t to launching the drone, then moves to x_R^t on the edge $\overline{V^1V^2}$ to retrieve it, then it goes to x_L^t on $\overline{V^2V^3}$ to launching it again, and travels until x_R^t on $\overline{V^3V^4}$ to retrieve the drone and returns to the starting point at V^1.

$$d_{LR}^{ee't}=\begin{cases}|\gamma_R^{et}-\gamma_L^{et}|\mathscr{L}(e), & \text{if } e=e',\\ \left(1-\gamma_L^{et}\right)\mathscr{L}\left(e\right)+\sum_{e''=e+1}^{e'-1}\mathscr{L}\left(e''\right)+\gamma_R^{e't}\mathscr{L}\left(e'\right), & \text{if } e<e', \quad (d_{LR}'')\\ \gamma_L^{et}\mathscr{L}\left(e\right)+\sum_{e''=e'+1}^{e-1}\mathscr{L}\left(e''\right)+\left(1-\gamma_R^{e't}\right)\mathscr{L}\left(e'\right), & \text{if } e>e'.\end{cases}$$

$$d_{RL}^{ee't}=\begin{cases}|\gamma_L^{et+1}-\gamma_R^{et}|\mathscr{L}(e), & \text{if } e=e',\\ \left(1-\gamma_R^{et}\right)\mathscr{L}\left(e\right)+\sum_{e''=e+1}^{e'-1}\mathscr{L}\left(e''\right)+\gamma_L^{e't+1}\mathscr{L}\left(e'\right), & \text{if } e<e', \quad (d_{RL}'')\\ \gamma_R^{et}\mathscr{L}\left(e\right)+\sum_{e''=e'+1}^{e-1}\mathscr{L}\left(e''\right)+\left(1-\gamma_L^{e't+1}\right)\mathscr{L}\left(e'\right), & \text{if } e>e'.\end{cases}$$

The reader should observe that, as described in Table 5, $d_{LR}^{ee't}$ accounts for the distance traveled by the mothership on $\mathscr{P}$ between consecutive launching and rendezvous points, whereas $d_{RL}^{ee't}$ does it for consecutive retrieving and next launching points.

In order to actually include these expressions in a formulation one has to decide which one of the two definitions ($e<e'$ or $e>e'$) is to be used in the objective function. This is the goal of the binary variables $z_{LR}^{ee't}$ that determines whether $d_{LR}^{ee'}$ is defined by the first or the second expression in the formula:

$$z_{LR}^{ee't}=1 \text{ if } \mu_L^{et}=1 \text{ and } \mu_R^{e't}=1,$$

where μ_L^{et} and μ_R^{et} are indicator variables that attain the value one if the launching point (resp. rendezvous point) is placed in the segment e of $\mathscr{P}$ at the stage t. Similarly, we define $z_{RL}^{ee't}$:

$$z_{RL}^{ee't}=1 \text{ if } \mu_R^{et}=1 \text{ and } \mu_L^{e't+1}=1.$$

With all this, we can model the movement of the mothership on the polygonal. For each stage $t\in T$, we have

$$x_L^t=\sum_{e=(i,i+1)\in\mathscr{P}}\mu_L^{et}\left[V^i+\gamma_L^{et}\left(V^{i+1}-V^i\right)\right], \tag{19}$$

$$x_R^t=\sum_{e=(i,i+1)\in\mathscr{P}}\mu_R^{et}\left[V^i+\gamma_R^{et}\left(V^{i+1}-V^i\right)\right], \tag{20}$$

$$z_{LR}^{ee't}=\mu_L^{et}\mu_R^{e't}, \quad \forall e,e'\in\mathscr{P}, \tag{21}$$

$$z_{RL}^{ee't}=\mu_R^{et}\mu_L^{e't+1}, \quad \forall e,e'\in\mathscr{P}, \tag{22}$$

$$\sum_{e\in\mathscr{P}}\mu_L^{et}=1, \tag{23}$$

$$\sum_{e\in\mathscr{P}}\mu_R^{et}=1, \tag{24}$$

$$d_{LR}^t=\sum_{e,e'\in\mathscr{P}}z_{LR}^{ee't}d_{LR}^{ee'}, \tag{25}$$

$$d_{RL}^t=\sum_{e,e'\in\mathscr{P}}z_{RL}^{ee't}d_{RL}^{ee't}, \tag{26}$$

where $d_{LR}^{ee'}$ and $d_{RL}^{ee't}$ are defined in $(d_{LR}^{t\prime\prime})$ and $(d_{RL}^{t\prime\prime})$. Constraints (19) and (20) parameterize the launching and rendezvous points on the polygonal chain. Constraints (21) and (22) set the variable z by means of the binary variables μ_L^{et} and μ_R^{et}. Constraints (23) and (24) state that, in each stage, the launching and rendezvous points can be only in one segment, not necessarily the same. Constraints (25) and (26) compute the total distance between the launching and the rendezvous points in each stage.

In addition, for each stage $t\in T$ we can define continuous variables λ_R^t and λ_L^t that model the absolute position of the points x_L^t and x_R^t in the polygonal to ensure that the movement of the mothership goes always in the allowed direction of travel and never comes back when it traverses the polygonal chain.

$$\lambda_L^t-i\geqslant\mu_L^{et}-(p-1)\left(1-\mu_L^{et}\right), \quad \forall e=(i,i+1)\in\mathscr{P}, \tag{27}$$

$$\lambda_L^t-i\leqslant\mu_L^{et}-(p-1)\left(1-\mu_L^{et}\right), \quad \forall e=(i,i+1)\in\mathscr{P}, \tag{28}$$

$$\lambda_R^t-i\geqslant\mu_R^{et}-(p-1)\left(1-\mu_R^{et}\right), \quad \forall e=(i,i+1)\in\mathscr{P}, \tag{29}$$

$$\lambda_R^t-i\leqslant\mu_R^{et}+(p-1)\left(1-\mu_R^{et}\right), \quad \forall e=(i,i+1)\in\mathscr{P}, \tag{30}$$

$$\lambda_R^t\geqslant\lambda_L^t, \quad \forall t\in T, \tag{31}$$

$$\lambda_L^{t+1}\geqslant\lambda_R^t, \quad \forall t\in T. \tag{32}$$

Inequalities (27)–(30) link μ_L^{et} and λ_L^t (resp. μ_R^{et} and λ_R^t). Finally, constraints (31) and (32) ensure that the mothership moves in the right direction on the polygonal. The goal of the (PMDRPG) is, once again, to find a feasible solution that minimizes the total distance covered by the mothership and the drone. Thus, gathering all the above constraints one gets the following valid formulation:

$$\min\sum_{g\in\mathscr{G}}\sum_{e_g\in E_g}\sum_{t\in T}\left(u^{e_gt}d_L^{e_gt}+v^{e_gt}d_R^{e_gt}\right)+\sum_{g\in\mathscr{G}}\sum_{e_g\in E_g}\mu^{e_g}d^{e_g}+\sum_{g\in\mathscr{G}}\sum_{e_g,e_g'\in E_g}z^{e_ge_g'}d^{e_ge_g'}$$
$$+\sum_{t\in T}\left(d_{RL}^t+d_{LR}^t\right) \qquad (PMDRPG)$$

s.t. (1)–(7)
(19)–(32)
(MTZ_1)–(MTZ_2) or (SEC)
(α−E) or (α-G)
(DCW-t)
$(d_{LR}^{t\prime\prime})$, $(d_{RL}^{t\prime\prime})$
($\text{DIST}_{1\text{-t}}$)–($\text{DIST}_{6\text{-t}}$).
(ORIG_1)–(DEST_2).

4.2.1. *Alternative formulation for PMDRPG based on MTZ-constraints*

Analogously to the case of AMDRPG one can also model PMDRPG without any explicit reference to stages. Here, we will follow a very similar approach to that in Section 4.1.3 to describe the formulation for PMDRPG based on MTZ-constraints. Table 6 reports a summary of the different sets of variables used for modelling this problem variant. Let $\mathscr{P}$

Table 6
Decision Variables for PMDRPG-MTZ.

Binary and Integer Decision Variables
$\mu^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: equal to 1 if edge e of graph g (or a portion of it) is visited by the drone, and 0 otherwise.
$\text{entry}^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: auxiliary binary variables for linearization.
$u^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: equal to 1 if the drone enters in graph g by e_g, 0 otherwise.
$z^{e_g e'_g} \in \{0,1\},\ \forall e_g, e'_g \in E_g\ (g \in \mathscr{G})$: equal to 1 if the drone goes from e_g to e'_g of graph g, 0 otherwise.
$v^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: equal to 1 if the drone exits from graph g by e_g, 0 otherwise.
$w^{gg'} \in \{0,1\},\ \forall g,g' \in \mathscr{G}$: equal to 1 if the mothership moves from x_R^g to $x_L^{g'}$, 0 otherwise.
$s^{e_g},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: integer non negative variables representing the order of visit of edge e of graph g.
$z_{LR}^{ee'g} \in \{0,1\},\ \forall e,e' \in \mathscr{P},\ \forall g \in \mathscr{G}$: equal to 1 if the launching point x_L^g, associated with graph g, is located on e and the rendezvous point x_R^g is located on e'.
$z_{RL}^{ee'gg'} \in \{0,1\},\ \forall e,e' \in \mathscr{P},\ \forall g,g' \in \mathscr{G}$: equal to 1 if the rendezvous point x_R^g, associated with graph g, is located on e and the launching point $x_L^{g'}$, associated with graph g', is located on e'.
$\mu_L^{eg} \in \{0,1\},\ \forall e \in \mathscr{P},\ \forall g \in \mathscr{G}$: equal to 1 if the launching point x_L^g, associated with graph g, is located on e.
$\mu_R^{eg} \in \{0,1\},\ \forall e \in \mathscr{P},\ \forall g \in \mathscr{G}$: equal to 1 if the rendezvous point x_R^g, associated with graph g, is located on e.
Continuous Decision Variables
$\rho^{e_g} \in [0,1]$ and $\lambda^{e_g} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathscr{G})$: defining the entry and exit points on edge e_g.
$\gamma_R^{eg} \in [0,1]$ and $\gamma_L^{eg} \in [0,1],\ \forall e \in \mathscr{P},\ \forall g \in \mathscr{G}$: defining the launching and rendezvous points, associated with graph g, on edge e.
$\nu_{\min}^{e_g}$ and $\nu_{\max}^{e_g} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathscr{G})$: auxiliary variables for linearization.
$x_L^g,\ \forall g \in \mathscr{G}$: coordinates representing the point where the mothership launches the drone to visit graph g.
$x_R^g,\ \forall g \in \mathscr{G}$: coordinates representing the point where the mothership retrieves the drone to visit graph g.
$R^{e_g},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: coordinates representing the entry point on edge e of graph g.
$L^{e_g},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: coordinates representing the exit point on edge e of graph g.
$d_L^{e_g} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathscr{G})$: distance travelled by drone from launching point x_L^g to the first visiting point R^{e_g} on e_g.
$d^{e_g e'_g} \geqslant 0,\ \forall e_g, e'_g \in E_g (g \in \mathscr{G})$: distance travelled by drone from launching point L^{e_g} on e_g to the rendezvous point $R^{e'_g}$ on e'_g.
$d^{e_g} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathscr{G})$: representing the distance travelled by the drone from the rendezvous point R^{e_g} to the launching point L^{e_g} on e_g.
$d_R^{e_g} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathscr{G})$: representing the distance travelled by the drone from the last visiting point L^{e_g} on graph g to the rendezvous point x_R^g on the mothership.
$d_{LR}^{ee'g} \geqslant 0,\ \forall e,e' \in \mathscr{P},\ \forall g \in \mathscr{G}$: representing the distance travelled by the mothership from the launching point x_L^g on e to the rendezvous point x_R^g on e' to visit graph g.
$d_{RL}^{ee'gg'} \geqslant 0,\ \forall e,e' \in \mathscr{P},\ \forall g,g' \in \mathscr{G}$: representing the distance travelled by the mothership from the rendezvous point x_R^g on e, for graph g, to the launching point $x_L^{g'}$ on e', for graph g'.
$d_{LR}^{g} \geqslant 0,\ \forall g \in \mathscr{G}$: representing the total distance travelled by the mothership between the launching point x_L^g and the rendezvous point x_R^g to visit graph g.
$d_{RL}^{gg'} \geqslant 0,\ \forall g,g' \in \mathscr{G}$: representing the total distance travelled by the mothership between the rendezvous point x_R^g, for graph g, and the launching point $x_L^{g'}$, for graph g'.

denote, as before, the polygonal chain parameterized by its endpoints $V^1, \ldots, V^p$ ordered in the direction of travel. In addition, let also γ_L^{eg} and γ_R^{eg} be the parameter values that define the points x_L^g and x_R^g on the line segment $\overline{V^i V^{i+1}}$ in $\mathscr{P}$, respectively. Then, the distance traveled on the polygonal, by the mothership, can be obtained by one of the following expressions. The first one refers to distances covered by the mothership between the launching and rendezvous points of a drone operation, whereas the second one models the distance traveled by the mothership between the rendezvous and launching points of two consecutive target graphs in its route.

$$d_{LR}^{ee'g} = \begin{cases} |\gamma_L^{eg} - \gamma_R^{eg}|\mathscr{L}(e), & \text{if } e = e', \\ \left(1-\gamma_L^{eg}\right)\mathscr{L}\left(e\right) + \sum_{e''=e+1}^{e'-1} \mathscr{L}\left(e''\right) + \gamma_R^{e'g}\mathscr{L}\left(e'\right), & \text{if } e < e', \\ \gamma_L^{eg}\mathscr{L}\left(e\right) + \sum_{e''=e+1}^{e'-1} \mathscr{L}\left(e''\right) + \left(1-\gamma_R^{e'g}\right)\mathscr{L}\left(e'\right), & \text{if } e > e'. \end{cases} \quad (d_{LR}^{g''})$$

$$d_{RL}^{ee'gg'} = \begin{cases} |\gamma_L^{eg} - \gamma_R^{eg'}|\mathscr{L}(e), & \text{if } e = e', \\ \left(1-\gamma_R^{eg}\right)\mathscr{L}\left(e\right) + \sum_{e''=e+1}^{e'-1} \mathscr{L}\left(e''\right) + \gamma_L^{e'g'}\mathscr{L}\left(e'\right), & \text{if } e < e', \\ \gamma_R^{eg}\mathscr{L}\left(e\right) + \sum_{e''=e+1}^{e'-1} \mathscr{L}\left(e''\right) + \left(1-\gamma_L^{e'g'}\right)\mathscr{L}\left(e'\right), & \text{if } e > e'. \end{cases} \quad (d_{RL}^{g''})$$

This distance requires binary variables $z_{LR}^{ee'g}$ and $z_{RL}^{ee'gg'}$ that determines which of the expressions in $(d_{LR}^{g''})$ and $(d_{RL}^{g''})$ define $d_{LR}^{ee'g}$ and $d_{RL}^{ee'gg'}$, respectively.

$$z_{LR}^{ee'g} = 1 \text{ iff } \mu_L^{eg} = 1 \text{ and } \mu_R^{e'g} = 1, \qquad z_{RL}^{ee'gg'} = 1 \text{ iff } \mu_L^{eg} = 1 \text{ and } \mu_R^{e'g'} = 1,$$

where μ_L^{eg} and μ_R^{eg} are indicator variables that attain the value one if the launching point (resp. rendezvous point) associated with the graph g is placed in the segment $\overline{V^i V^{i+1}}$ of the edge $e = (i, i+1) \in \mathscr{P}$. With all this, we can model the movement of the mothership on the polygonal as follows:

$$x_L^g = \sum_{e=(i,i+1)\in\mathscr{P}} \mu_L^{eg}\left[V^i + \gamma_L^{eg}\left(V^{i+1} - V^i\right)\right], \quad \forall g \in \mathscr{G}, \tag{33}$$

$$x_R^g = \sum_{e=(i,i+1)\in\mathscr{P}} \mu_R^{eg}\left[V^i + \gamma_R^{eg}\left(V^{i+1} - V^i\right)\right], \quad \forall g \in \mathscr{G}, \tag{34}$$

$$z_{LR}^{ee'g} = \mu_L^{eg}\mu_R^{e'g}, \quad \forall e,e' \in \mathscr{P},\ \forall g \in \mathscr{G}, \tag{35}$$

$$z_{RL}^{ee'gg'} = \mu_R^{eg}\mu_L^{e'g'}, \quad \forall e,e' \in \mathscr{P},\ \forall g,g' \in \mathscr{G}, \tag{36}$$

$$\sum_e \mu_L^{eg} = 1, \quad \forall g \in \mathscr{G}, \tag{37}$$

$$\sum_e \mu_R^{eg} = 1, \quad \forall g \in \mathscr{G}, \tag{38}$$

$$d_{LR}^{g} = \sum_{e,e'\in\mathscr{P}} z_{LR}^{ee'g} d_{LR}^{ee'g}, \quad \forall g \in \mathscr{G}, \tag{39}$$

$$d_{RL}^{gg'} = \sum_{e,e'\in\mathscr{P}} z_{RL}^{ee'gg'} d_{RL}^{ee'gg'}, \quad \forall g,g' \in \mathscr{G}, \tag{40}$$

where $d_{LR}^{ee'g}$ and $d_{RL}^{ee'gg'}$ are defined in $(d_{LR}^{g''})$ and $(d_{RL}^{g''})$, respectively. Constraints (33) and (34) parameterize the launching and rendezvous points on the polygonal chain. Constraints (35) and (36) set the binary variables $z_{LR}^{ee'g}$ and $z_{RL}^{ee'gg'}$ by means of the binary variables μ_L^{eg} and μ_R^{eg}. Constraints (37) and (38) state that, for each target graph, the launching and rendezvous points can be only in one segment, not necessarily the same. Constraints (39) and (40) compute the total distance between the launching and the rendezvous points for each pair of points. The reader may note that the above representation is similar to that given by (19)–(26) with the exception that constraints (33)–(40) are indexed in the set of graphs $g \in \mathscr{G}$ instead that on the stages. This allows a remarkable

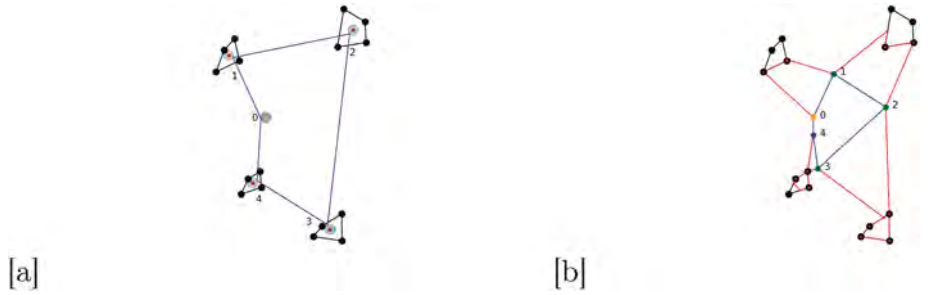


Fig. 4. Illustrative example.

reduction in the number of distance variables and constraints as already discussed in Section 4.1.4. The goal of the PMDRPG is to find a feasible solution that minimizes the total distance covered by the drone and the mothership, i.e.:

$$\min \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g}\left(u^{e_g}d_L^{e_g}+v^{e_g}d_R^{e_g}\right)+\sum_{g\in\mathcal{G}}\sum_{e_g\in E_g}\mu^{e_g}d^{e_g}+\sum_{g\in\mathcal{G}}\sum_{e_g,e_g'\in E_g}z^{e_g e_g'}d^{e_g e_g'}+\sum_{g\in\mathcal{G}}d_{LR}^{g}$$
$$+\sum_{g,g'\in\mathcal{G}}d_{RL}^{gg'}w^{gg'}\qquad\left(\text{PMDRPG-MTZ}\right)$$

s.t. (10)–(17)
(MTZ_1)–(MTZ_2) or (SEC)
(MTZ_3)–(MTZ_6) or (SEC)
(33)–(40)
($\alpha-$E) or α-G
DCW-g
($d_{LR}^{g\prime}$), ($d_{RL}^{g\prime}$)
($DIST_1$-g)–($DIST_6$-g).
($ORIG_1$)–($ORIG_2$). The reader may note that the above formulation actually encloses two different ones depending whether it includes constraints (MTZ_1)-(MTZ_2) or (SEC). However, for the sake of presentation we have preferred to present them in this compact form to reduce the length of the paper.

As mentioned at the beginning of this section, the PMDRPG is an intermediate variant of the problem, between the case in which the mothership moves freely in the continuous space and the more general situation in which the mothership moves on a general network. For modelling details of this latter case we refer the reader to the Appendix.

4.3. Strengthening the formulations of MDRPG

The different models that we have proposed include in one way or another big-M constants. We have defined different big-M constants along this work. In order to strengthen the formulations we provide tight upper and lower bounds for those constants. The computation of these bounds is rather technical and may distract the reader from the main focus of the paper. Therefore, we have preferred to present in the Appendix, at the end of the paper, all the details and results that adjust the different bounds for each one of the models studied in this paper.

5. A Matheuristic for the Mothership-Drone Routing Problem with Graphs

This section is devoted to present our matheuristic approach to address the solution of the MDRPG. Our motivation comes from the fact that the exact methods presented in the previous section are highly time demanding. Alternatively, the matheuristic provides good quality solution in limited computing times.

First, we focus on the case in which the mothership can move freely on the continuous space, namely AMDRPG. Since dealing with the exact model is a hard task for medium size instances, our strategy is to split the problem. The rationale of this algorithm rests on decomposing the problem in simpler subproblems decoupling the decisions made on the route followed by the mothership and the ones made on the drone. To do that, first we solve a TSP for the mothership over the neighborhoods of centroids of target graphs, that is an extension of the crossing postman problem where the Hamiltonian routes have to visit neighborhoods or polygonal chains rather than edges (XPPN, see Puerto and Valverde (2020)) and then we solve drone operations one at a time based on the order previously defined for the mothership. In this way each subproblem is much simpler and once their solutions are found one can integrate them into a feasible solution of the whole problem. Specifically, the basic idea of the algorithm is to determine first the sequence of visits to the target graphs in $\mathcal{G}$. This route is obtained replacing each graph in $\mathcal{G}$ by its centroid and then solving a TSP over the neighborhoods of those centroids, namely a XPPN. Then, given such an order of visits, the launching/rendezvous points where the mothership must stop, are determined by solving an AMDRPG problem, but limited to one single graph each time, following the sequence previously computed. Successively, the solution obtained is improved by providing it to the solver as initial partial solution and by solving, once more, another AMDRPG problem, where now we fix the launching/rendezvous points but leave free the variables w that give the sequence of visits to the targets graphs, i.e., without fixing the order of visits. If this solution induces a different order of visits to the graphs with respect to the previous one, the procedure is repeated to generate a new feasible solution. The stopping criterion is to find a solution that does not change with respect to the one obtained in the previous cycle of the algorithm. In the following, we present the pseudo-code of this algorithm:

STEP 1 (Centroids of the graphs)
- Let *orig* be the origin/destination of the mothership tour (orange point with label 0 in Fig. 4 [a]). For each graph $g \in \mathcal{G}$:
- identify its centroid c_g and consider its neighborhood defined as the circle $\rho(c_g, 2)$ centered at c_g and with radius 2 (red points in Fig. 4 [a]).

STEP 2 (Order of visit of the graphs) Determine an order of visit for the graphs in $\mathcal{G}$ by solving the XPPN of the mothership over the set of the neighborhoods associated with the centroids of those graphs (blue tour 0, 1, 2, 3, 4, 0 in Fig. 4 [a]).

STEP 3 (Determining the location of launching/rendezvous points) Let $\overline{w}_{gg'}, \forall g, g' \in \mathcal{G}$ be the optimal values of the variables $w_{gg'}$ generated by STEP 2.

Following this order of visit, set the launching point for the first graph as the depot, then solve the resulting AMDRPG limited to the first graph.

Repeat the same procedure for the remaining graphs to be visited, by solving AMDRPG on one single graph each time, by fixing as launching point of the current graph the rendezvous point of the previous graph.

STEP 4 (Solution update) Let $\overline{z}$ be the solution obtained by STEP 3, consisting of the tour of the drone on each target (Fig. 4 [b]), and let $\overline{x}_L^g, \overline{x}_R^g \ \forall g \in \mathcal{G}$ be the associated launching/rendezvous points (green

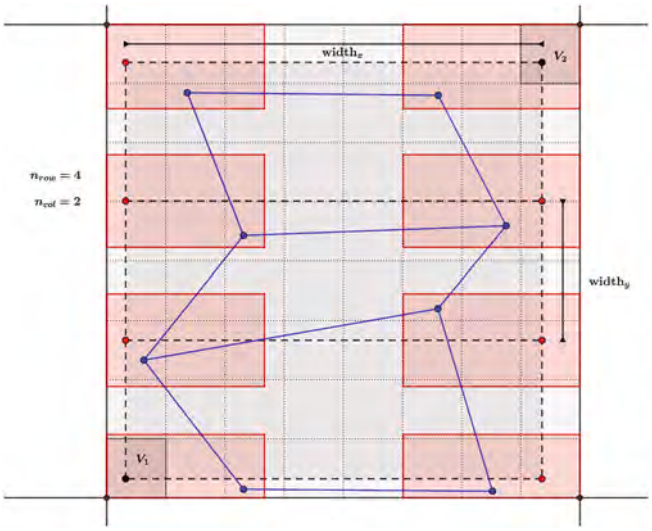


Fig. 5. Example of generation of a grid graph.

Table 7
Comparison between formulations for grid instances.

Gap %	Average		Min		Max	
Solver	Cplex	Gurobi	Cplex	Gurobi	Cplex	Gurobi
Formulation						
Stages	87	87	85	84	88	88
MTZ	66	62	59	58	72	65
SEC	65	61	59	57	70	64

and blue points in Fig. 4 [b]).

Solve the model AMDRPG with these launching/rendezvous points but leaving free the $w_{gg'}$ $\forall g, g' \in \mathcal{G}$ variables and providing to the solver $\bar{z}$ as initial partial solution.

STEP 5 Let $\widehat{z}$ be the updated solution obtained by STEP 4 (see Fig. 4 [b] for this illustrative example) and $\widehat{w}_{gg'}$ the associated order of visit of the graphs.

If the $\widehat{w}_{gg'} \neq \overline{w}_{gg'}$ repeat from STEP 3, otherwise stop.

Note that in the example shown in Fig. 4 we solved the AMDRPG model by imposing that at least a given percentage of each target graph must be visited. For this reason the solution consists in drone's missions which visit only part of each graph, as represented by the red segments in Fig. 4 [b]. Moreover, for this example, most of the launching/rendezvous points coincide. More specifically, considering Fig. 4 [b], the drone starts from the origin and visits part of the first target. The mothership retrieves it at the point labelled 1. From the same point the drone starts the second mission to visit part of the second target and then reaches the mothership at the point labelled 2. From that point the drone starts for visiting the third target meeting the mothership at the point labelled 3. From this latter point the last mission of the drone starts and ends on the mothership at the point labelled 4. Then the mothership with the drone go back to the origin. In this example the solution obtained at STEP 4, by

Table 8
Comparison between formulations for Delauney instances.

Gap %	Average		Min		Max	
Solver	Cplex	Gurobi	Cplex	Gurobi	Cplex	Gurobi
Formulation						
Stages	91	91	90	89	93	93
MTZ	78	74	74	70	82	79
SEC	77	75	73	69	82	81

Table 9
Heuristic performances.

#	Grid			Delauney		
	Best Obj	Obj Heuristic	CPU Time	Best Obj	Obj Heuristic	CPU Time
0	1087,87	1117,83	50,99	947,01	934,46	52,49
1	1100,38	1319,64	24,64	986,22	938,68	72,73
2	1350,67	1126,35	46,06	888,48	865,66	1073,80
3	1218,66	1476,36	27,18	1249,69	1154,62	1703,33
4	1297,77	1424,37	40,91	1239,93	1184,67	81,15

fixing the launching/rendezvous points but not the order of visit, does not change with respect to the one obtained at STEP 3. Thus the procedure stops after the first iteration and provides the solution shown in Fig. 4 [b]. We note in passing that the exact solution of AMDRPG on STEP 4 can still be hard, specially on large instances. For this reason the stopping criterion of each call of the solver within the heuristic procedure is set to a maximum number of feasible solutions found. The procedure followed for the NMDRPG problem consists in the same steps of the matheuristic presented above for the AMDRPG problem but, at STEP 4, the partial solution, given as input to the solver, includes also the values of the variables modelling the routing of the mothership on the graph.

6. Experimental results

In this section we discuss the experimental results obtained testing the formulations presented in Section 4 and the matheuristic procedure proposed in Section 5 on different sets of instances. In particular, we generate two sets of instances of the AMDRPG problem. The first one consists of targets, to be visited by the drone, that are represented by grid graphs, while the second one involves a different typology of targets, namely, Delaunay graphs. For both typologies we generate 5 instances of 10 graphs each, with different cardinality of the set of nodes. More precisely, each instance is composed of 3 graphs with 4 nodes, 3 graphs with 6 nodes, 3 graphs with 8 nodes and 1 graph with 10 nodes. Moreover, we assume that the drone's speed is twice that of the mothership and that a percentage equal to 80% of each target must be visited by the drone.

Table 10
Comparison between exact resolution with and without initialization.

List	%	Grid			Delauney		
		% Gap (i)	Time_h	% Gap (wi)	% Gap (i)	Time_h	% Gap (wi)
0	e	72	105.12	73	78	154.92	74
	g	55	58.92	54	62	92.64	67
1	e	76	241.99	76	80	314.69	79
	g	71	182.61	70	74	353.04	75
2	e	76	367.69	76	80	447.61	80
	g	71	326.49	72	76	429.16	76
3	e	75	481.68	74	80	514.98	76*
	g	71	492.27	70	77	582.90	77

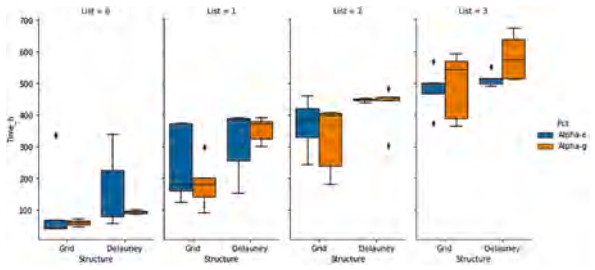


Fig. 6. Matheuristic running time.

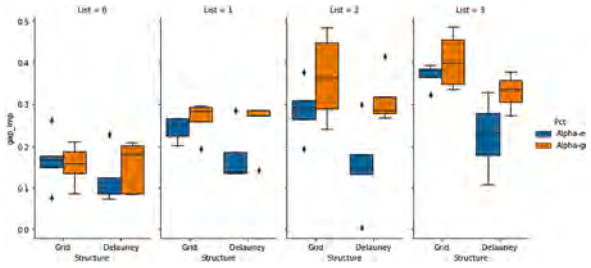


Fig. 7. Matheuristic improved gap.

In order to place the 10 graphs of a single instance, we consider a square of side 100 units. Then we divide the original square in subsquares of side 5 and we randomly select among them the locations for the ten target graphs of the instance. The generation procedure of the single graph in a selected square, depends on the graph typology. As regards grid graphs, the single subsquare, like the one represented in Fig. 5, is further partitioned in subsquares of side $\frac{5}{n}$ where n is the cardinality of the set of nodes of the graph to build. Two opposite corner subsquares are considered (like V_1 and V_2 in Fig. 5) and one point inside each of them is randomly selected (black points in the subsquares V_1 and V_2 in Fig. 5). Then, a rectangle whose diagonal joins these two points is built (the black dotted one in Fig. 5). A grid of n points is identified by locating $\frac{n}{m}$ equally spaced points on the two sides square (the red ones and the two original points in black in Fig. 5), where m is randomly selected in the set of divisors of the number of points of the graph. The links of the graphs connect each point to its adjacent ones lying on the same side and with the one located on the opposite side of the square. Let $width_x$ and $width_y$ be the lengths of these edges as show in Fig. 5. In order to perturb the coordinates of these points, we randomly add a value, ranging between $-\frac{width_x}{3}$ and $\frac{width_x}{3}$ to the x coordinate and between $-\frac{width_y}{3}$ and $\frac{width_y}{3}$ to the y coordinate, always imposing that the perturbed point still belongs to the square. The resulting grid graph is obtained connecting the same pairs of points but with perturbed coordinates (blue graph in Fig. 5).

As regards the instances related to Delaunay graphs, we select their locations following the same procedure as for the grid ones. Then, for each randomly chosen subsquare, given the set of n nodes of the graph to be generated, a Delaunay triangulation of them is computed by using the Python class scipy.spatial.Delaunay (see Virtanen et al. (2020) for further details).

We run the three formulations proposed in Section 4 for the AMDRPG problem (Stages, MTZ and SEC) with two different commercial solvers, Cplex 12.8 and Gurobi 9.03, called by means of Python, by setting a time limit for each run equal to 3600 s. In Table 7 we report the results obtained, in terms of average, minimum and maximum percentage gap with both solvers, on the instances consisting of grid graphs. First, we can observe that Gurobi has better performances with respect to Cplex for all the instances. Moreover, for this set of instances, the SEC formulation is the best one among the three proposed, as the associated average gap is equal to 61% and also the minimum and the maximum percentage gap are smaller than the ones associated with formulations Stages and MTZ.

Similarly, Table 8 summarizes the results obtained on the instances with Delaunay graphs. Also in this case the Gurobi performance is better than Cplex. However, among the three formulations, the MTZ one provides the best results in terms of average and maximum percentage gap.

In order to test the performance of the matheuristic proposed in Section 5, we code it in Python and we run it on the same sets of instances (Grid and Delaunay) on which the three exact formulations were solved. Table 9 reports for each of the five instances, numbered from 0 to 4, distinguishing between Grid and Delaunay, respectively, the best objective function provided by the best formulation, the objective function provided by the matheuristic and the associated CPU time. As already noticed from Table 8, for grid graph instances, SEC formulation has the best behaviour, with the exception of the instance number 3 for which the MTZ provides a smaller value of the objective function. As for the Delaunay graph instances, MTZ is the best formulation, but also in this case there is an exception on the instance number 2 for which SEC formulation returns a smaller value of the objective function.

The results show that the matheuristic returns a solution with value of the objective function that is higher than the one provided by the SEC formulation on grid instances. However, these values are smaller than the ones provided by the Stages and the MTZ formulations. Moreover, the saving in terms of solution time is very significant as the maximum CPU time is less than 1 min. As regards the Delaunay instances, the matheuristic performance is even better, as it finds a solution that is better than the best one provided by the MTZ formulation and in a solution time that is at most 28 min.

We perform a second set of experiments by observing that, even if there are small differences between the SEC and the MTZ formulations depending on the type of instances, their performances are comparable. Thus, in the rest of the tests we focus on the MTZ formulation. We compare its performance, with or without providing the initial solution found by the matheuristic, on a set of larger instances. More precisely, we generate 20 instances with targets represented by grid graphs and 20 instances with targets represented by Delaunay graphs. The instances of each typology are split in 4 groups of 5 instances each, consisting respectively of 5, 10, 15 and 20 targets to be visited. In each instance the same percentage of graphs (20%) has respectively 4, 6, 8, 10 and 12 nodes. Moreover, we assume that the origin coincides with the destination in all instances and we randomly generate with uniform distribution between 0 and 1, two values representing the percentage of each edge and of each graph to be visited. As regards the speeds, we set the speed of the drone three times the one of the mothership. We run the MTZ formulation using Gurobi and setting a time limit of 7200 s. for each instance. On the same instances also the matheuristic is applied. Note that, in order to define a stopping rule for the exact solution of the AMDRPG model within the matheuristic procedure (STEP 3 and STEP 4), we set a maximum number of solutions generated by the solver equal to five. For each instance, the solution provided by the matheuristic is then used to initialize the exact application of the MTZ formulation in order to try to speed up the solution process. Table 10 shows the results of the comparison between the exact formulation with and without

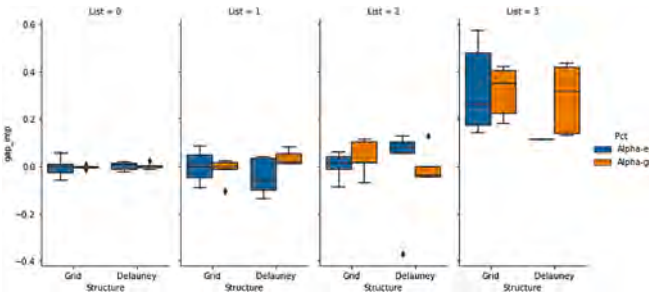


Fig. 8. Improved gap of MTZ formulation with and without initialization.

Table 11
Comparison between formulations of NMDRPG.

	Net Struct	1		2		3	
List	%	Stages	MTZ	Stages	MTZ	Stages	MTZ
0	e	89	33	88	24	87	39
	g	86	29	89	18	90	42
1	e	92	43	92	33	92	46
	g	91	36	92	23	92	39

Table 12
Comparison between exact resolution with and without initialization of NMDRPG.

	Net Struct	1			2			3		
List	%	% Gap (i)	T_h	% Gap (wi)	% Gap (i)	T_h	% Gap (wi)	% Gap (i)	T_h	% Gap (wi)
0	e	32	109.96	33	24	207	24	39	177.57	39
	g	30	1192	29	18	163.36	18	45	149.68	42
1	e	48	1030.64	43	39	802.3	33	53	770.05	46
	g	33	479.36	36	35	639.09	23	42	689.51	39

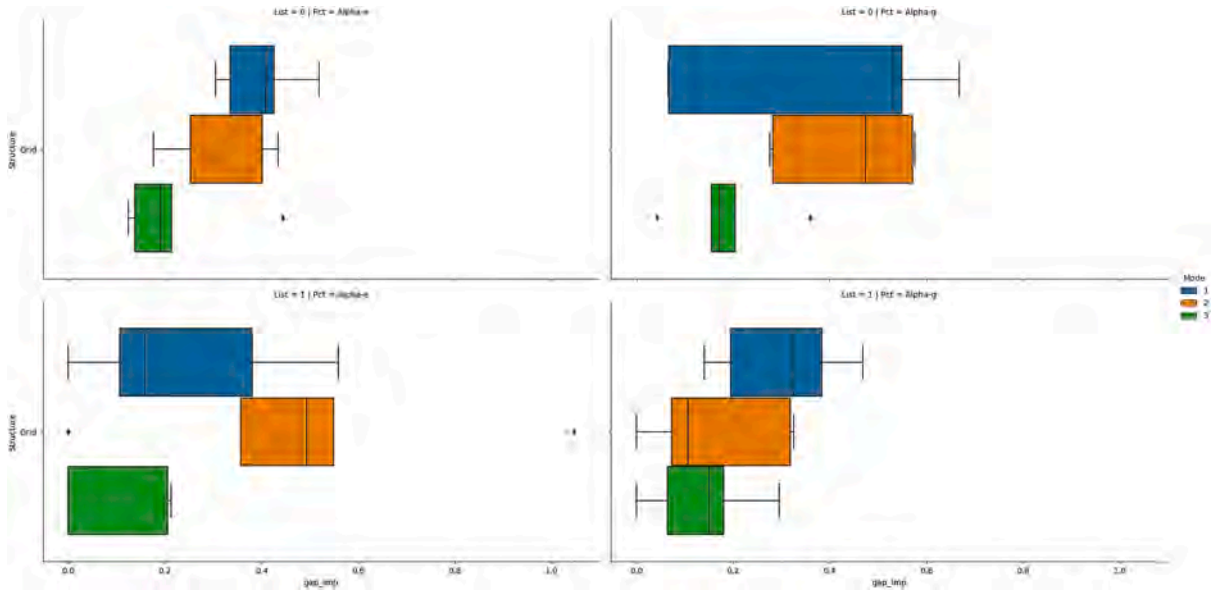


Fig. 9. Matheuristic improved gap for NMDRPG.

initialization. In the first column, named List, we report the size of the instances in terms of number of targets to be visited (0, 1, 2 and 3 identifies instances respectively with 5, 10, 15 and 20 graphs). The second column refers to the two variants of the model, that is, a given percentage of each edge of the targets (e) or a given percentage of each target (g) must be visited by the drone. The other columns report respectively the average percentage gap of the solutions found within the time limit starting from the initial solution provided by the matheuristic (% Gap (i)), the average running time of the matheuristic (Time h) and the average percentage gap of the solutions found within the time limit without initialization (% Gap (wi)). This information is reported for both Grid and Delauney instances.

From Table 10 we can notice that in most of the cases the average gaps associated with the solution found within the time limit, with and without initialization by the solution found by the matheuristic, are the same or very close (note that in the last column the $*$ indicates that only one instance has been solved within the time limit). As regards the running time of the matheuristic, we can see also from the boxplots in Fig. 6, that it increases with the number of targets to be visited both for Grid and Delaunay instances. Considering the model variants based on the minimum percentage of each edge or each graph to visit, we can observe that for Grid instances the average running time of the model imposing a minimum percentage of each edge to be visited, is greater than the one associated with the other variant, with the exception of the instances of biggest size (List = 3).

The boxplots in Fig. 7 represent the percentage gap of the solution provided by the matheuristic with respect to the one provided by the exact resolution of the MTZ model within the time limit, with initialization by the solution found by the matheuristic. From them we can notice that the gap increases with the size both for Grid and Delaunay instances but it is always less than 50%. Fig. 8 shows the percentage gap of the solution provided by the exact solution of the MTZ formulation within the time limit without the initialization, with respect to the one found with the initialization. These observations suggest that, even if the initialization of the model by the solution provided by the matheuristic does not speed up the convergence to the optimal solution, the matheuristic provides solutions of very good quality. Indeed, it generates in less than 10 min solutions that are very close to the ones provided by the model within 2 h.

As regards the NMDRPG problem, we generate three sets of instances with targets represented by grid graphs considering different structures of the network where the mothership can move. In particular, we define a first set of instances where the mothership network is represented by a graph of 6 nodes with a tree structure with origin of the path of the base vehicle different from the destination. A second set of instances involving a mothership network consisting of a complete graph of 4 vertices with origin of the path of the base vehicle different from the destination. A third set of instances characterized by star graphs of 7 nodes representing the mothership network, where the origin coincides with the destination and it is located at the centre of the star. We generate 10 instances for each of these three classes, 5 of them with 5 targets and 5 with 10 targets to be visited. Moreover, as for the AMDRPG, for each of these 10 instances we randomly generate two values representing the percentage of each edge and of each graph that must be visited by the drone. We run on these sets of instances both Stages and MTZ formulations. Table 11 summarizes the results obtained comparing them. The first column identifies the size of the instances, similarly to Table 10, (0 for instances with 5 targets and 1 for instances with 10 targets). The second column distinguishes between minimum percentage of each edge (e) or of each graph (g) to be visited by the drone. The remaining columns refer to the three different classes of instances described above (1 for the networks with a tree structure, 2 for

complete networks and 3 for start networks). For each of these sets of instances the average percentage gap of the solutions found within the time limit of 7200 s. by the two formulations (Stages and MTZ) is reported.

We can observe that for each class of instances and model variants, based on the percentage of each edge or each graph to be visited, the MTZ formulation performs better than the Stages one. In all the cases the percentage average gap associated with the MTZ formulation is one third or half of that associated with the Stages formulation. For this reason, in the following tests, related to the comparison between the exact solution of the NMDRPG model with and without the initialization by the solution found by the matheuristic, we focus only on the MTZ formulation.

Table 12 summarizes the results of this comparison distinguishing again between the different network structures (columns labelled 1, 2 and 3), the different size (rows labelled 0 and 1) characterizing the instances and model variants (minimum percentage of each edge (e) or each graph (g) to be visited). For each combination of network structure, size and model variant we report the average percentage gap with initialization (% Gap (i)), the solution time of the matheuristic (T_h) and the average percentage gap without initialization by the solution found by the matheuristic (% Gap (wi)).

We can observe that, similarly to the AMDRPG problem, the average gaps associated with the solution found within the time limit, with and without initialization by the solution found by the matheuristic, are very close. Considering the average running time we can notice that the NMDRPG problem is more challenging to be solved than the AMDRPG. The solution time increases very fast with the size of the instances especially for the case in which the network where the mothership moves has a tree structure. Moreover, as for the Grid instances in the continuous case, the model variant imposing a minimum percentage of each edge to be visited takes more time to be solved.

The boxplots showed in Fig. 9 report the percentage gap of the solution provided by the matheuristic with respect to the one provided by the exact solution of the MTZ model within the time limit, with initialization by the solution found by the matheuristic. We can notice that, excluding the outliers, this gap ranges between 0% and 60% and its lowest values are observed for the instances in which the network where the mothership moves has a star structure (green boxplots). From the previous observations, similarly to the AMDRPG, we can conclude that the behaviour of the matheuristic is very good in terms of quality of the solutions provided, even if used as initialization for the MTZ model does not help in speeding up the convergence to the optimal solution.

7. Concluding remarks

This papers has analyzed the coordination problem that arises between a mothership vehicle and a drone that must adjust their routes to minimize travel distances while visiting a set of targets modeled by graphs. We have presented exact formulations for different versions of the problem depending on the constraints imposed to the mothership movement (free on a continuous space or on a given network). Our computational results show that the considered problem is rather hard and only small to medium size problems can be solved to optimality. Additionally, we have proposed a matheuristic algorithm, applicable to all the versions of the problem with minimum changes, that provides acceptable feasible solutions in short computing time; so that it is a good alternative to the exact methods.

Further research in this topic includes the coordination of the operations of several drones with a mothership, the possibility of visiting more than one target per operation and combinations of both cases. These problems being very interesting are beyond the scope of this paper and will be the focus of a follow up paper.

CRediT authorship contribution statement

Lavinia Amorosi: Conceptualization, Methodology, Formal analysis, Writing - original draft, Writing - review & editing, Visualization. **Justo Puerto:** Supervision, Conceptualization, Methodology, Formal analysis, Writing - original draft, Writing - review & editing. **Carlos Valverde:** Conceptualization, Methodology, Formal analysis, Software, Writing - original draft, Writing - review & editing, Visualization.

Acknowledgements

This research has been partially supported by Spanish Ministry of Education and Science/FEDER Grant No. MTM2016-74983-C02-(01-02), Junta de Andalucia project P18-FR-1422 and projects FEDER-US-1256951, CEI-3-FQM331 and *NetmeetData*: Ayudas Fundación BBVA a equipos de investigación cientóifica 2019, Ministerio de Ciencia e Innovación PID2020-114594GB-C21 and by University of Rome, Sapienza Grant No.: RM11916B7F962975.

Appendix A

A.1. Strengthening the formulations of MDRPG

A.1.1. Big-M constants bounding the distance from the launching/ rendezvous point on the path followed by the mothership to the rendezvous/ launching point on the target graph $g \in \mathcal{G}$

- AMDRPG. To linearize the first term of the objective function in AMDRPG, we define the auxiliar non-negative continuous variables $p_L^{e_g t}$ (resp. $p_R^{e_g t}$) to model the product $d_L^{eg} u^{e_g t}$ and include the constraints (8) and (9), namely:

$$p_L^{e_g t} \geqslant m_L^{e_g} u^{e_g t}, p_L^{e_g t} \leqslant d_L^{e_g} - M_L^{e_g t}\left(1 - u^{e_g t}\right). \tag{41}$$

The best upper bound $M_R^{e_g t}$ or $M_L^{e_g t}$ that we can consider is the full diameter of the data, that is the maximum distance between every pair of vertices of the graphs $g \in \mathcal{G}$. every launching or rendezvous point is inside the circle whose diametrically opposite points are determined by the following expression.

$$M_R^{e_g t} = \max_{\left\{v \in V_g, v' \in V_{g'} : g, g' \in \mathcal{G}\right\}} \|v - v'\| = M_L^{e_g t}.$$

On the other hand, the minimum distance in this case can be zero. This bound is attainable whenever the launching or the rendezvous points of the mothership is the same that the rendezvous or launching point on the target graph $g \in \mathcal{G}$.

Table 13
Nomenclature for NMDRPG.

Problem Parameters
$\mathcal{N} = (V, E)$: set of nodes and edges of the network representing the road system where the mothership can move.
$e = (i,j)$,$e' = (i',j')$: starting edge and ending edge of the mothership tour.
$\mathcal{G}$: set of the target graphs.
$g = (V_g, E_g)$: set of nodes and edges of each target graph $g \in \mathcal{G}$.
$\mathcal{L}(e_g)$: length of edge e of graph $g \in \mathcal{G}$.
B^{e_g}, C^{e_g}: endpoints of edge e of graph $g \in \mathcal{G}$.
α^{e_g}: percentage of edge e of graph $g \in \mathcal{G}$ that must be visited.
α^{g}: percentage of graph $g \in \mathcal{G}$ that must be visited.
v_D: drone speed.
v_M: mothership speed.
M: big M.

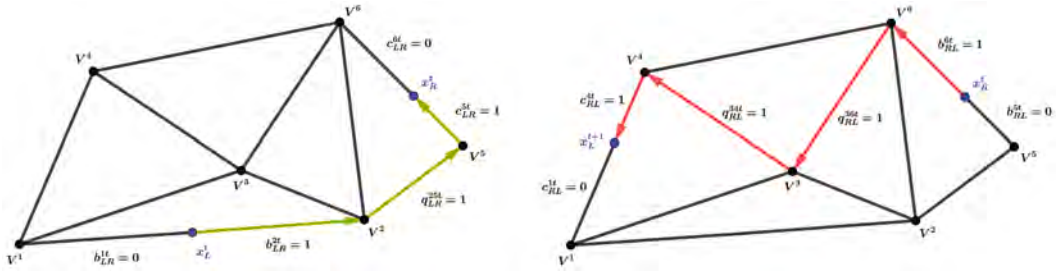


Fig. 10. An example of a parameterization of a mothership route for a stage t. In the first picture, the mothership launches the drone at the point x^t_L on the edge $\overline{V^1V^2}$, then traverses the edge $\overline{V^2V^3}$ and finally moves to x^t_R on the edge $\overline{V^4V^5}$ to retrieve it. In the second one, the mothership retrieves the drone at the point x^t_R on the edge $\overline{V^4V^5}$, then traverses the edge $\overline{V^3V^6}$, the edge $\overline{V^3V^4}$ and finally moves to x^{t+1}_L on the edge $\overline{V^1V^4}$ to launch the drone for the next stage.

Table 14
Decision Variables for NMDRPG-ST.

Binary and Integer Decision Variables
$\mu^{e_g} \in \{0,1\}$, $\forall e_g \in E_g$ $(g \in \mathcal{G})$: equal to 1 if edge e of graph g (or a portion of it) is visited by the drone, and 0 otherwise.
$\text{entry}^{e_g} \in \{0,1\}$, $\forall e_g \in E_g$ $(g \in \mathcal{G})$: auxiliary binary variables for linearization.
$u^{e_g t} \in \{0,1\}$, $\forall e_g \in E_g$ $(g \in \mathcal{G})$, $\forall t \in T$: equal to 1 if the drone enters in graph g by e_g at stage t, 0 otherwise.
$z^{e_g e'_g} \in \{0,1\}$, $\forall e_g, e'_g \in E_g$ $(g \in \mathcal{G})$: equal to 1 if the drone goes from e_g to e'_g, 0 otherwise.
$v^{e_g t} \in \{0,1\}$, $\forall e_g \in E_g$ $(g \in \mathcal{G})$, $\forall t \in T$: equal to 1 if the drone exits from graph g by e_g at stage t, 0 otherwise.
s^{e_g}, $\forall e_g \in E_g$ $(g \in \mathcal{G})$: integer non negative variable representing the order of visit of edge e of graph g.
$\mu^{et}_L \in \{0,1\}$, $\forall e \in E$, $\forall t \in T$: equal to 1 if the launching point x^t_L is located on e at stage t.
$\mu^{et}_R \in \{0,1\}$, $\forall e \in E$, $\forall t \in T$: equal to 1 if the rendezvous point x^t_R is located on e at stage t.
$z^{ee't}_{LR} \in \{0,1\}$, $\forall e, e' \in E$, $\forall t \in T$: equal to 1 if the launching point x^t_L is located on e and the rendezvous point x^t_R is located on e' at stage t.
$b^{it}_{LR} \in \{0,1\}$, $\forall i : e = (i,j) \in E$, $\forall t \in T$: equal to 1 if the mothership exits from x^t_L by the vertex V^i of the edge e.
$c^{it}_{LR} \in \{0,1\}$, $\forall i : e = (i,j) \in E$, $\forall t \in T$: equal to 1 if the mothership enters in x^t_R by the vertex V^i of the edge e.
$q^{et}_{LR} \geqslant 0$, $\forall e \in E$, $\forall t \in T$: integer variable counting the number of times the mothership fully traverses edge e to move between the launching point x^t_L on e to the rendezvous point x^t_R on e' at stage t.
$z^{ee't}_{RL} \in \{0,1\}$, $\forall e, e' \in E$, $\forall t \in T$: equal to 1 if the rendezvous point x^t_R is located on e at stage t and the launching point $x^{(t+1)}_L$ is located on e' at stage $t+1$.
$b^{it}_{RL} \in \{0,1\}$, $\forall i : e = (i,j) \in E$, $\forall t \in T$: equal to 1 if the mothership exits from x^t_R by the vertex V^i of the edge e.
$c^{it}_{RL} \in \{0,1\}$, $\forall i : e = (i,j) \in E$, $\forall t \in T$: equal to 1 if the mothership enters in x^{t+1}_L by the vertex V^i of the edge e.
$q^{et}_{RL} \geqslant 0$, $\forall e \in E$, $\forall t \in T$: integer variable counting the number of times the mothership fully traverses edge e to move between the rendezvous point x^t_R on e to the launch point for the next stage $x^{(t+1)}_L$.
Continuous Decision Variables
$\rho^{e_g} \in [0,1]$ and $\lambda^{e_g} \in [0,1]$, $\forall e_g \in E_g$ $(g \in \mathcal{G})$: defining the entry and exit points on edge e_g.
$\gamma^{et}_L \in [0,1]$ and $\gamma^{et}_R \in [0,1]$, $\forall e \in E$, $\forall t \in T$: defining the launching and rendezvous points on edge e of the network $\mathcal{N}$ at stage t.
$\nu^{e_g}_{\min}$ and $\nu^{e_g}_{\max} \in [0,1]$, $\forall e_g \in E_g$ $(g \in \mathcal{G})$: auxiliary variables for linearization.
x^t_L, $\forall t \in T$: coordinates representing the point where the mothership launches the drone at stage t.
x^t_R, $\forall t \in T$: coordinates representing the point where the mothership retrieves the drone at stage t.
R^{e_g}, $\forall e_g \in E_g$ $(g \in \mathcal{G})$: coordinates representing the entry point in edge e of graph g.
L^{e_g}, $\forall e_g \in E_g$ $(g \in \mathcal{G})$: coordinates representing the exit point from edge e of graph g.

(*continued on next page*)

Table 14 (*continued*)

$d_L^{e_g t} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathscr{G}),\ \forall t \in T$: representing the distance travelled by the drone from the launching point x_L^t on the mothership at stage t to the first visiting point R^{e_g} on e_g.
$d^{e_g e_g'} \geqslant 0,\ \forall e_g, e_g' \in E_g (g \in \mathscr{G})$: representing the distance travelled by the drone from launching point L^{e_g} on e_g to the rendezvous point $R^{e_g'}$ on e_g'.
$d^{e_g} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathscr{G})$: representing the distance travelled by the drone from the rendezvous point R^{e_g} to the launching point L^{e_g} on e_g.
$d_R^{e_g t} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathscr{G}),\ \forall t \in T$: representing the distance travelled by the drone from the last visiting point L^{e_g} to the rendezvous point x_R^t on the mothership at stage t.
$d_{LR}^{ee' t} \geqslant 0,\ \forall e, e' \in E,\ \forall t \in T$: representing the distance travelled by the mothership from the launching point x_L^t on e to the rendezvous point x_R^t on e' at stage t.
$d_{RL}^{ee' t} \geqslant 0,\ \forall e, e' \in E,\ \forall t \in T$: representing the distance travelled by the mothership from the rendezvous point x_R^t on e at stage t to the launching point $x_L^{(t+1)}$ on e' at stage $t+1$.
$d_{LR}^t \geqslant 0,\ \forall t \in T$: representing the total distance travelled by the mothership between the launching point x_L^t and the rendezvous point x_R^t at stage t.
$d_{RL}^t \geqslant 0,\ \forall t \in T$, representing the total distance travelled by the mothership between the rendezvous point x_R^t at stage t and the launching point $x_L^{(t+1)}$ at stage $t+1$.

- NMDRPG. In this case, the best upper bounds for $M_R^{e_g t}$ or $M_L^{e_g t}$ is the maximum distance between the polygonal chain $\mathscr{P}$ or the graph $\mathscr{N}$ and any of the target graphs $g \in \mathscr{G}$:

$$M_R^{e_g t} = \max_{\{v \in V_g, w \in \mathscr{N}\}} \|v - w\| = M_L^{i_g t}.$$

Table 15
Decision Variables for NMDRPG-MTZ.

Binary and Integer Decision Variables
$\mu^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: equal to 1 if edge e of graph g (or a portion of it) is visited by the drone, and 0 otherwise.
$\text{entry}^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: auxiliary binary variables for linearization.
$u^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: equal to 1 if the drone enter in graph g by edge e_g, 0 otherwise.
$z^{e_g e_g'} \in \{0,1\},\ \forall e_g, e_g' \in E_g\ (g \in \mathscr{G})$: equal to 1 if the drone goes from e_g to e_g', 0 otherwise.
$v^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: equal to 1 if the drone exits from graph g by e_g, 0 otherwise.
$w^{gg'} \in \{0,1\},\ \forall g, g' \in \mathscr{G}$: equal to 1 if the mothership moves from x_R^g to $x_L^{g'}$, 0 otherwise.
$s^{e_g},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: integer non negative variables representing the order of visit of edge e of graph g.
$\mu_L^{eg} \in \{0,1\},\ \forall e \in E,\ \forall g \in \mathscr{G}$: equal to 1 if the launching point x_L^g to visit graph g is located on e.
$\mu_R^{eg} \in \{0,1\},\ \forall e \in E,\ \forall g \in \mathscr{G}$: equal to 1 if the rendezvous point to visit graph g is located on e.
$z_{LR}^{ee' g} \in \{0,1\},\ \forall e, e' \in E,\ \forall g \in \mathscr{G}$: equal to 1 if the launching point x_L^g to visit graph g is located on e and the rendezvous point x_R^g is located on e'.
$b_{LR}^{ig} \in \{0,1\},\ \forall i : e = (i,j) \in E,\ \forall g \in \mathscr{G}$: equal to 1 if the mothership exits from x_L^g by the vertex V^i of the edge e.
$c_{LR}^{ig} \in \{0,1\},\ \forall i : e = (i,j) \in E,\ \forall g \in \mathscr{G}$: equal to 1 if the mothership enters in x_R^g by the vertex V^i of the edge e.
$q_{LR}^{eg} \geqslant 0,\ \forall e \in E,\ \forall g \in \mathscr{G}$: integer variable counting the number of times the mothership fully traverses edge e to move between the launching point x_L^g on e to the rendezvous point x_R^g on e' for graph g.
$z_{RL}^{ee' gg'} \in \{0,1\},\ \forall e, e' \in E,\ \forall g, g' \in \mathscr{G}$: equal to 1 if the rendezvous point x_R^g, for graph g, is located on e and the launching point $x_L^{g'}$, for graph g', is located on e'.
$b_{RL}^{ig} \in \{0,1\},\ \forall i : e = (i,j) \in E,\ \forall g \in \mathscr{G}$: equal to 1 if the mothership exits from x_R^g by the vertex V^i of the edge e.
$c_{RL}^{ig} \in \{0,1\},\ \forall i : e = (i,j) \in E,\ \forall g \in \mathscr{G}$: equal to 1 if the mothership enters in x_L^g by the vertex V^i of the edge e.
$q_{RL}^{egg'} \geqslant 0,\ \forall e \in E,\ \forall g \in \mathscr{G}$: integer variable counting the number of times the mothership fully traverses edge e to move between the rendezvous point x_R^g to the launching point $x_L^{g'}$ for graph g.
Continuous Decision Variables
$\rho^{e_g} \in [0,1]$ and $\lambda^{e_g} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathscr{G})$: defining the entry and exit points on edge e_g.
$\gamma_R^{eg} \in [0,1]$ and $\gamma_L^{eg} \in [0,1],\ \forall e \in E,\ \forall g \in \mathscr{G}$: defining the launching and rendezvous points, associated with graph g, located on edge e of the network $\mathscr{N}$.
$\nu_{\min}^{e_g}$ and $\nu_{\max}^{e_g} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathscr{G})$: auxiliary variables for linearization.
$x_L^g,\ g \in \mathscr{G}$: coordinates representing the point where the mothership launches the drone to visit graph g.
$x_R^g,\ g \in \mathscr{G}$: coordinates representing the point where the mothership retrieves the drone to visit graph g.
$R^{e_g},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: coordinates representing the entry point on edge e of graph g.
$L^{e_g},\ \forall e_g \in E_g\ (g \in \mathscr{G})$: coordinates representing the exit point on edge e of graph g.
$d_L^{e_g} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathscr{G})$: representing the distance travelled by the drone from the launching point x_L^g on the mothership to the first visiting point R^{e_g} on e_g.
$d^{e_g e_g'} \geqslant 0,\ \forall e_g, e_g' \in E_g (g \in \mathscr{G})$: representing the distance travelled by the drone from launching point L^{e_g} on e_g to the rendezvous point $R^{e_g'}$ on e_g'.
$d^{e_g} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathscr{G})$: representing the distance travelled by the drone from the rendezvous point R^{e_g} to the launching point L^{e_g} on e_g.
$d_R^{e_g} \geqslant 0,\ \forall e_g \in E_g\ (g \in \mathscr{G})$: representing the distance travelled by the drone from the last visiting point for graph g L^{e_g} to the rendezvous point x_R^g on the mothership.
$d_{LR}^{ee' g} \geqslant 0,\ \forall e, e' \in E,\ \forall g \in \mathscr{G}$: representing the distance travelled by the mothership from the launching point x_L^g on e and the rendezvous point x_R^g on e' to visit graph g.
$d_{RL}^{ee' gg'} \geqslant 0,\ \forall e, e' \in E,\ \forall g, g' \in \mathscr{G}$: representing the distance travelled by the mothership from the rendezvous point x_R^g, for graph g, on e to the launching point $x_L^{g'}$, for graph g', on e'.
$d_{LR}^g \geqslant 0,\ \forall g \in \mathscr{G}$: representing the total distance travelled by the mothership between the launching point x_L^g and the rendezvous point x_R^g to visit graph g.
$d_{RL}^{gg'} \geqslant 0,\ \forall g, g' \in \mathscr{G}$: representing the total distance travelled by the mothership between the rendezvous point x_R^g, for graph g, and the launching point $x_L^{g'}$, for graph g'.

On the other hand, the minimum distance can be computed by taking the closest points between the graph g and the network $\mathcal{N}$:

$$m_R^{e_g t} = \min_{\{v \in V_g, w \in \mathcal{N}\}} \|v - w\| = m_L^{e_g t}.$$

A.1.2. Bounds on big-M constants for distances from the launching to the rendezvous points for the MTZ/SEC formulations in AMDRPG

To linearize the product of $d_{RL}^{gg'} w^{gg'}$ we use the constraints:

$$\begin{aligned} p^{gg'} &\geqslant m_{RL}^{gg'} d_{RL}^{gg'}, \\ p^{gg'} &\leqslant d_{RL}^{gg'} - M_{RL}^{gg'}(1 - w^{gg'}). \end{aligned}$$

The upper bound on $M_{RL}^{gg'}$ is given by the diameter of $g \cup g'$, namely $M_{RL}^{gg'} = \max_{\{v \in V_g, v' \in V_{g'}\}} \|v - v'\|$.

A.1.3. Bounds on big-M constants for distances from launching to rendezvous points on target graph $g \in \mathcal{G}$.

When the drone visits a graph g, it has to go from one edge e_g to another edge e_g' depending on the order given by $z^{e_g e_g'}$. This fact produces another product of variables linearized by the following constraints:

$$\begin{aligned} p^{e_g e_g'} &\geqslant m^{e_g e_g'} d_{RL}^{gg'}, \\ p^{e_g e_g'} &\leqslant d^{e_g e_g'} - M^{e_g e_g'}(1 - z^{e_g e_g'}). \end{aligned}$$

Since we are taking into account the distance between two edges $e = (B^{e_g}, C^{e_g}), e' = (B^{e_g'}, C^{e_g'}) \in E_g$, the maximum and minimum distances between their vertices give us the upper and lower bounds:

$$\begin{aligned} M^{e_g e_g'} &= \max\{\|B^{e_g} - C^{e_g'}\|, \|B^{e_g} - B^{e_g'}\|, \|C^{e_g} - B^{e_g'}\|, \|C^{e_g} - C^{j_g}\|\}, \\ m^{e_g e_g'} &= \min\{\|B^{e_g} - C^{e_g'}\|, \|B^{e_g} - B^{e_g'}\|, \|C^{e_g} - B^{e_g'}\|, \|C^{e_g} - C^{e_g'}\|\}. \end{aligned}$$

A.1.4. Bounds on big-M constants for distances covered by the mothership on the polygonal for the PMDRPG model during one drone operation.

In the case of PMDRPG, we can also set tighter upper bounds for the distance covered by the drone inside the polygonal during an operation that starts in e and finishes at e' (or vice versa) (see (25) and (26)). This is clearly bounded above by the total length of the line segments where the mothership is located.

$$M_{LR}^{ee't} = M_{RL}^{ee't} = \begin{cases} \mathcal{L}(e), & \text{if } e = e', \\ \sum_{e < e'' < e'} \mathcal{L}\left(e''\right), & \text{if } e < e', \\ \sum_{e' < e'' < e} \mathcal{L}\left(e''\right), & \text{if } e > e'. \end{cases}$$

A.1.5. Bounds on big-M constants for distances covered by the drone during an operation in models by stages.

To link the drone operation with the mothership travel in the models by stages, we have defined the constraint (DCW-t) that includes the M:

$$\left(\sum_{e_g \in E_g} u^{e_g t} d_L^{e_g t} + \sum_{e_g, e_g' \in E_g} z^{e_g e_g'} d^{e_g e_g'} + \sum_{e_g \in E_g} \mu^{e_g} d^{e_g} + \sum_{e_g \in E_g} v^{e_g t} d_R^{e_g t}\right) \Big/ v_D \leqslant d_{RL}^t \Big/ v_M + M\left(1 - \sum_{e_g \in E_g} u^{e_g t}\right).$$

To obtain this upper bound M we add to the length of the graph $\mathcal{L}(g)$ the big-Ms. computed for $u^{e_g t}$ and $v^{e_g t}$, that is, $M_L^{e_g t}$ and $M_R^{e_g t}$, and the upper bounds on the distances traveled by the drone to move between edges on g. Thus,:

$$M = \mathcal{L}\left(g\right) + M_L^{e_g t} + M_R^{e_g t} + \sum_{e_g, e_g' \in E_g} M^{e_g e_g'}.$$

A.2. Network Mothership-Drone Routing Problem with Graphs (NMDRPG)

This model is a refinement of the one presented in Section 4.2. Indeed, in this case the mothership has to move on a general undirected network rather than on a polygonal chain. Even though the model in Section 4.2, namely PMDRPG, can be seen as a particular case of NMDRPG, for the sake of presentation, we have preferred to include it first to ease the understanding of the more advanced NMDRPG model.

As before, we start with an approach that models the problem by stages. Later, we shall extend the analysis using the rationale of connectivity, as already done in the previous models AMDRPG and PMDRPG. Table 13 summarizes all the parameters for this problem variant.

Let $\mathscr{N}=(V,E)$ be the network that models the space of movement for the mothership. For each stage $t\in T:=\{1,\ldots,|\mathscr{G}|\}$, the mothership can start from an edge $e=(i,j)\in E$ and end in another one $e'=(i',j')\in E$, moving between each other following a route on $\mathscr{N}$. Thus, it must follow a route from the launching to the rendezvous point and vice versa, determined by a sequence of intermediate points as shown in Fig. 10. All the decision variables of this formulation proposed for the NMDRPG model are reported in Table 14. As already explained before in previous sections, the distance traveled by the mothership, between two consecutive launching and rendezvous points in two edges, not necessarily distinct, of the graph can be represented as:

$$d_{LR}^{ee't}=\begin{cases}|\gamma_L^{et}-\gamma_R^{et}|\mathscr{L}(e), & \text{if } e=e',\\ \left[b_{LR}^{it}\gamma_L^{et}+b_{LR}^{jt}(1-\gamma_L^{et})\right]\mathscr{L}\left(e\right)+\sum_{e''\in\mathscr{N}}q_{LR}^{e''t}\mathscr{L}\left(e''\right)+\left[c_{LR}^{i't}\gamma_R^{e't}+c_{LR}^{j't}(1-\gamma_R^{e't})\right]\mathscr{L}\left(e'\right), & \text{if} e\neq e',\end{cases}\qquad \left(d_{LR}^{t\mathscr{N}}\right)$$

where b_{LR}^{it} (resp. c_{LR}^{it}) are binary variables that determine from which of the end-points of e (respectively e') one has to account for the distance and q_{LR}^{et} is an integer variable that counts how many times the mothership fully traverses the edge e. Furthermore, we need to define another binary variable $z_{LR}^{ee't}$ that models the correct definition of the distance in $(d_{LR}^{t\mathscr{N}})$. With the above definition one can account for the movement of the mothership at each stage $t\in T$ from a launching to a rendezvous point:

$$x_L^t=\sum_{e=(i,j)\in E}\mu_L^{et}\left[V^i+\gamma_L^{et}(V^j-V^i)\right],\quad \forall t\in T, \tag{42}$$

$$x_R^t=\sum_{e=(i,j)\in E}\mu_R^{et}\left[V^i+\gamma_R^{et}(V^j-V^i)\right],\quad \forall t\in T, \tag{43}$$

$$z_{LR}^{ee't}=\mu_L^{et}\mu_R^{e't},\quad \forall e,e'\in E,\ \forall t\in T, \tag{44}$$

$$b_{LR}^{it}\leqslant\sum_{e\in\delta(i)}\mu_L^{et},\qquad \forall i\in V,\ \forall t\in T, \tag{45}$$

$$c_{LR}^{it}\leqslant\sum_{e\in\delta(i)}\mu_R^{et},\qquad \forall i\in V,\ \forall t\in T, \tag{46}$$

$$b_{LR}^{it}+b_{LR}^{jt}\geqslant\mu_L^{et},\qquad \forall e=(i,j)\in E,\ \forall t\in T, \tag{47}$$

$$c_{LR}^{it}+c_{LR}^{jt}\geqslant\mu_R^{et},\qquad \forall e=(i,j)\in E,\ \forall t\in T, \tag{48}$$

$$b_{LR}^{it}+\sum_{\{j:(i,j)\in E\}}q_{LR}^{ijt}=\sum_{\{j:(i,j)\in E\}}q_{LR}^{ijt}+c_{LR}^{it},\qquad \forall i\in V,\ \forall t\in T, \tag{49}$$

$$\sum_{e\in E}\mu_L^{et}=1,\ \forall t\in T, \tag{50}$$

$$\sum_{e\in E}\mu_R^{et}=1,\ \forall t\in T, \tag{51}$$

$$d_{LR}^{t}=\sum_{e,e'\in E}z_{LR}^{ee't}d_{LR}^{ee't},\ \forall t\in T. \tag{52}$$

Constraints (42) and (43) parameterize the launching and rendezvous points in the network $\mathscr{N}$ at stage t. Constraints (44) set the binary variables $z_{LR}^{ee't}$ by means of the binary variables μ_L^{et} and $\mu_R^{e't}$. Constraints (45) and (46) state that if the launching point (resp. rendezvous point) is not on the edge e, the mothership cannot go (resp. exit) to the vertex that is incident to e. Constraints (47) state that if the mothership leaves the edge e to go to e', it must exit from one of the incident vertices to e. In the same way, constraints (48) express that if the mothership leaves the edge e' to go to e, it must enter to the edge e from one of its incident vertices. Flow conservation constraints (49) ensure that the number of incoming edges to each vertex i is equal to the number of outgoing edges in the route followed by the mothership. Constraints (50) and (51) state that, in each stage, the selected points can be located only on one edge. Finally, constraints (52) define the total distance between the launching and the rendezvous points at stage t.

Similarly, the distance covered by the mothership along the path on the network from the rendezvous point x_R^t to the next launching point x_L^{t+1} can be modeled using the following definition of distance:

$$d_{RL}^{ee't}=\begin{cases}|\gamma_R^{et}-\gamma_L^{et+1}|\mathscr{L}(e), & \text{if } e=e',\\ \left[b_{RL}^{it}\gamma_R^{et}+b_{RL}^{jt}(1-\gamma_R^{et})\right]\mathscr{L}(e)+\sum_{e''\in\mathscr{N}}q_{RL}^{e''t}\mathscr{L}(e'')+\left[c_{RL}^{i't}\gamma_L^{e't+1}+c_{RL}^{j't}(1-\gamma_L^{e't+1})\right]\mathscr{L}(e'), & \text{if } e\neq e'.\end{cases}\qquad (d_{RL}^{t\mathscr{N}})$$

In this case, the binary variable $z_{RL}^{ee't}$ links the rendezvous point at stage t with the launching point at stage $t+1$. Then, we can use a set of constraints similar to those used above and the distance from x_R^t to x_L^{t+1} can be computed by means of the following additional constraints:

$$z_{RL}^{ee't} = \mu_R^{et}\mu_L^{e't+1}, \qquad \forall e,e' \in E,\ \forall t \in T, \tag{53}$$

$$b_{RL}^{it} \leqslant \sum_{e\in\delta(i)} \mu_R^{et}, \qquad \forall i \in V,\ \forall t \in T, \tag{54}$$

$$c_{RL}^{it} \leqslant \sum_{e\in\delta(i)} \mu_L^{et}, \qquad \forall i \in V,\ \forall t \in T, \tag{55}$$

$$b_{RL}^{it} + b_{RL}^{jt} \geqslant \mu_R^{et}, \qquad \forall e = (i,j) \in E,\ \forall t \in T, \tag{56}$$

$$c_{RL}^{it} + c_{RL}^{jt} \geqslant \mu_L^{et+1}, \qquad \forall e = (i,j) \in E,\ \forall t \in T, \tag{57}$$

$$b_{RL}^{it} + \sum_{\{j:(i,j)\in E\}} q_{RL}^{jit} = \sum_{\{j:(i,j)\in E\}} q_{RL}^{jit} + c_{RL}^{it}, \qquad \forall i \in V,\ \forall t \in T, \tag{58}$$

$$\sum_{e\in E} \mu_L^{et} = 1, \qquad \forall t \in T, \tag{59}$$

$$\sum_{e\in E} \mu_R^{et} = 1, \qquad \forall t \in T, \tag{60}$$

$$d_{RL}^{t} = \sum_{e,e'\in E} z_{RL}^{ee't} d_{RL}^{ee't}, \qquad \forall t \in T. \tag{61}$$

Constraints (53) set the binary variables $z_{RL}^{ee't}$ by means of the binary variables μ_R^{et} and $\mu_L^{e't+1}$. Constraints (54) and (55) state that if the rendezvous point (resp. launching point) is not on the edge e, the mothership cannot go (resp. exit) to the end vertices of the edge e. Constraints (56) state that if the mothership leaves the edge e, it must exit via one of the end vertices of e. In the same way, constraints (57) state that if the mothership goes to the edge e, it necessarily must enter to e from one incident vertex of e. Flow conservation constraints (58) ensure that in the route followed by the mothership the number of used incoming edges to each vertex i is equal to the number of used outgoing edges. Constraints (59) and (60) state that, in each stage, the selected points can be only on one edge. Constraints (61) express the total distance between the rendezvous and the launching points at stage t.

Hence, once the distances inside the graph are set with the above two families of constraints, we can state the mathematical programming formulation of the problem as:

$$\min \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g}\sum_{t\in T}\left(u^{e_g t} d_L^{e_g t} + v^{e_g t} d_R^{e_g t}\right) + \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} \mu^{e_g} d^{e_g} + \sum_{g\in\mathcal{G}}\sum_{e_g,e_g'\in E_g} z^{e_g e_g'} d^{e_g e_g'} + \sum_{t\in T}\left(d_{RL}^t + d_{LR}^t\right) \qquad \text{(NMDRPG-ST)}$$

s.t. (1)–(7)
(42)–(61)
(MTZ_1)–(MTZ_2) or (SEC)
(α −E) or (α-G)
(DCW-t)
($d_{LR}^{t\mathcal{N}}$), ($d_{RL}^{t\mathcal{N}}$)
(ORIG_1)–(DEST_2)

Again, the objective function has four terms: the first three compute the distances traveled by the drone, while the last one computes the distances traveled by the mothership. Constraints (1)–(7) model the tour made by the drone. Constraints (42)–(61) model the path followed by the mothership in the graph. The rest of the constraints are similar to those explained in the formulation (AMDRPG-ST).

A.2.1. MTZ formulation for the Network Mothership-Drone Routing Problem with Graphs

In order to apply this type of constraints to model connectivity of the routes we have to reformulate the expressions for the distances so that they are not related to stages. Table 15 summarizes the variables required to formulate this model. In this case, we observe that the distance between launching and rendezvous points in two edges $e = (i,j), e' = (i',j') \in E$, not necessarily distinct, of the graph can be represented as:

$$d_{LR}^{ee'g} = \begin{cases} |\gamma_L^{eg} - \gamma_R^{eg}|\mathscr{L}(e), & \text{if } e = e', \\ \left[b_{LR}^{ig}\gamma_L^{eg} + b_{LR}^{jg}(1-\gamma_L^{eg})\right]\mathscr{L}\left(e\right) + \sum_{e''\in\mathcal{N}} q_{LR}^{e''g}\mathscr{L}\left(e''\right) + \left[c_{LR}^{i'g}\gamma_R^{e'g} + c_{LR}^{j'g}(1-\gamma_R^{e'g})\right]\mathscr{L}\left(e'\right), & \text{if } e \neq e', \end{cases} \qquad \left(d_{LR}^{g\mathcal{N}}\right)$$

where b_{LR}^{ig} (resp. c_{LR}^{ig}) are binary variables that determine from which of the end-points of e (respectively e') one has to account for the distance and q_{LR}^{eg} is a binary variable that is equal to one when the mothership fully traverses the edge e. Furthermore, we need to define binary variables $z_{LR}^{ee'g}$ that model the choice of the model for right definition (first or second formulas of the distance in ($d_{LR}^{g\mathcal{N}}$). All the above arguments give the necessary elements to account for the movement of the mothership on the network $\mathcal{N} = (V,E)$:

$$x_L^g = \sum_{e=(i,j)\in E} \mu_L^{eg}\left[V^i + \gamma_L^{eg}\left(V^j - V^i\right)\right], \qquad \forall g \in \mathcal{G}, \tag{62}$$

$$x_R^g = \sum_{e=(i,j)\in E} \mu_R^{eg}\left[V^i + \gamma_R^{eg}\left(V^j - V^i\right)\right], \qquad \forall g \in \mathcal{G}, \tag{63}$$

$$z_{LR}^{ee'g} = \mu_L^{eg}\mu_R^{e'g}, \qquad \forall e, e' \in E,\ \forall g \in \mathcal{G}, \tag{64}$$

$$b_{LR}^{ig} \leqslant \sum_{e\in\delta(i)} \mu_L^{eg}, \qquad \forall i \in V, \quad \forall g \in \mathcal{G}, \tag{65}$$

$$c_{LR}^{ig} \leqslant \sum_{e\in\delta(i)} \mu_R^{eg}, \qquad \forall i \in V, \quad \forall g \in \mathcal{G}, \tag{66}$$

$$b_{LR}^{ig} + b_{LR}^{jg} \geqslant \mu_L^{eg}, \qquad \forall e = (i,j) \in E, \quad \forall g \in \mathcal{G}, \tag{67}$$

$$c_{LR}^{ig} + c_{LR}^{jg} \geqslant \mu_R^{eg}, \qquad \forall e = (i,j) \in E, \quad \forall g \in \mathcal{G}, \tag{68}$$

$$b_{LR}^{ig} + \sum_{\{j:(i,j)\in E\}} q_{LR}^{iig} = \sum_{\{j:(i,j)\in E\}} q_{LR}^{ijg} + c_{LR}^{ig}, \qquad \forall i \in V, \quad \forall g \in \mathcal{G}, \tag{69}$$

$$\sum_{e\in E} \mu_L^{eg} = 1, \qquad \forall g \in \mathcal{G}, \tag{70}$$

$$\sum_{e\in E} \mu_R^{eg} = 1, \qquad \forall g \in \mathcal{G}, \tag{71}$$

$$d_{LR}^{g} = \sum_{e,e'\in E} z_{LR}^{ee'g} d_{LR}^{ee'g}, \qquad \forall g \in \mathcal{G}. \tag{72}$$

Constraints (62) and (63) parameterize the launching and rendezvous points in the network $\mathcal{N}$ induced by the visit to the graph $g \in \mathcal{G}$. Constraints (64) set the binary variables $z_{LR}^{ee'g}$ by means of the binary variables μ_L^{eg} and $\mu_R^{e'g}$. Constraints (65) and (66) state that if the launching point (resp. rendezvous point) is not on the edge e, the mothership cannot go (resp. exit) to the vertices of the edge e. Constraints (67) state that if the mothership leaves the edge e, it must exit from one of the incident vertices to e. In the same way, constraints (68) state that if the mothership leaves the edge e', it necessarily must enter to e from one incident vertex of e. Flow conservation constraints (69) ensure that, in the route to be defined, the number of incoming edges to each vertex i is equal to the number of outgoing edges. Constraints (70) and (71) state that, for the visit to the graph $g \in \mathcal{G}$, launching and rendezvous points can be only on one edge. Constraints (72) returns the total distance traveled by the drone on the graph $g \in \mathcal{G}$.

Similarly, the distance covered by the mothership along the path on the network $\mathcal{N}$, from the rendezvous point $x_R^g \in e$, after the visit to $g \in \mathcal{G}$ to the next launching point $x_L^{g'} \in e'$ (to go to the graph $g' \in \mathcal{G}$), can be modeled using the following definition of distance:

$$d_{RL}^{ee'gg'} = \begin{cases} |\gamma_R^{eg} - \gamma_L^{eg'}|\mathcal{L}(e), & \text{if } e = e', \\ \left[b_{RL}^{ig}\gamma_R^{eg} + b_{RL}^{jg}\left(1-\gamma_R^{eg}\right)\right]\mathcal{L}\left(e\right) + \sum_{e''\in\mathcal{N}} q_{RL}^{e''gg'}\mathcal{L}\left(e''\right) + \left[c_{RL}^{i'g'}\gamma_L^{e'g'} + c_{RL}^{j'g'}\left(1-\gamma_L^{e'g'}\right)\right]\mathcal{L}\left(e'\right), & \text{if } e \neq e'. \end{cases} \qquad \left(d_{RL}^{g,\mathcal{N}}\right)$$

In this case, the binary variable $z_{RL}^{ee'gg'}$ links the rendezvous point at g with the launching point at g'. Then, we can use a set of constraints similar to those used above and the distance from x_R^g to $x_L^{g'}$ can be computed by means of the following additional constraints:

$$z_{RL}^{ee'gg'} = \mu_R^{eg}\mu_L^{e'g'}, \qquad \forall e, e' \in E,\ \forall g, g' \in \mathcal{G}, \tag{73}$$

$$b_{RL}^{ig} \leqslant \sum_{e\in\delta(i)} \mu_R^{eg}, \qquad \forall i \in V,\ \forall g \in \mathcal{G}, \tag{74}$$

$$c_{RL}^{ig} \leqslant \sum_{e\in\delta(i)} \mu_L^{eg}, \qquad \forall i \in V,\ \forall g \in \mathcal{G}, \tag{75}$$

$$b_{RL}^{ig} + b_{RL}^{jg} \geqslant \mu_R^{eg}, \qquad \forall e = (i,j) \in E,\ \forall g \in \mathcal{G}, \tag{76}$$

$$c_{RL}^{ig} + c_{RL}^{jg} \geqslant \mu_L^{eg}, \qquad \forall e = (i,j) \in E,\ \forall g \in \mathcal{G}, \tag{77}$$

$$b_{RL}^{ig} + \sum_{\{j:(i,j)\in E\}} q_{RL}^{iigg'} = \sum_{\{j:(i,j)\in E\}} q_{RL}^{ijgg'} + c_{RL}^{ig'}, \qquad \forall i \in V,\ \forall g, g' \in \mathcal{G}, \tag{78}$$

$$d_{RL}^{gg'} = \sum_{e,e'\in E} z_{RL}^{ee'gg'} d_{RL}^{ee'gg'}, \qquad \forall g, g' \in \mathcal{G}. \tag{79}$$

Hence, once the distances inside the graph are set with the above two families of constraints, we can state the mathematical programming formulation of the problem as:

$$\sum_{g\in\mathcal{G}}\sum_{e_g\in E_g}\left(u^{e_g}d_L^{e_g}+v^{e_g}d_R^{e_g}\right)+\sum_{g\in\mathcal{G}}\sum_{e_g\in E_g}\mu^{e_g}d^{e_g}+\sum_{g\in\mathcal{G}}\sum_{e_g,e_g'\in E_g}z^{e_g e_g'}d^{e_g e_g'}+\sum_{g\in\mathcal{G}}d_{LR}^{g}+\sum_{g,g'\in\mathcal{G}}d_{RL}^{gg'}w^{gg'} \qquad \text{(NMDRPG-MTZ)}$$

s.t. (10)–(17)
(62)–(79)
(MTZ_1)–(MTZ_2) or (SEC)
(MTZ_3)–(MTZ_6) or (SEC)
(α −E) or (α-G)
(DCW-g)
($d_{LR}^{g\mathcal{V}'}$), ($d_{RL}^{g\mathcal{V}'}$)
($DIST_{1-g}$)–($DIST_{6-g}$)
($ORIG_1$)–($DEST_2$)

References

Amorosi, L., Chiaraviglio, L., D'Andreagiovanni, F., Blefari-Melazzi, N., 2018. Energy-efficient mission planning of UAVs for 5G coverage in rural zones. In: Proceedings of the IEEE International Conference on Environmental Engineering, EE 2018.

Amorosi, L., Chiaraviglio, L., Galan-Jimenez, J., 2019. Optimal energy management of uav-based cellular networks powered by solar panels and batteries: Formulation and solutions. IEEE Access 7, 53698–53717.

Amorosi, L., Caprari, R., Crainic, T., Dell'Olmo, P., Ricciardi, N., 2020. CIRRELT-2020-17 An Integrated Routing-Scheduling Model for a Hybrid UAV-Based Delivery System. CIRRELT.

Blanco, V., Fernández, E., Puerto, J., 2017. Minimum spanning trees with neighborhoods: Mathematical programming formulations and solution methods. Eur. J. Oper. Res. 262 (3), 863–878.

Campbell, J.F., Sweeney, D., Zhang, J., 2017. Strategic design for delivery with trucks and drones. Comput. Sci.

Campbell, J.F., Corberán, Á., Plana, I., Sanchis, J.M., 2018. Drone arc routing problems. Networks 72 (4), 543–559.

Carlsson, J.G., Song, S., 2018. Coordinated logistics with a truck and a drone. Manage. Sci. 64 (9), 4052–4069.

Chiaraviglio, L., Amorosi, L., Blefari-Melazzi, N., Dell'Olmo, P., Natalino, C., Monti, P., 2018. Optimal design of 5g networks in rural zones with uavs, optical rings, solar panels and batteries. In: Proceedings of the 20th International Conference on Transparent Optical Networks, ICTON 2018.

Chiaraviglio, L., Amorosi, L., Blefari-Melazzi, N., Dell'olmo, P., Lo Mastro, A., Natalino, C., Monti, P., 2019a. Minimum cost design of cellular networks in rural areas with uavs, optical rings, solar panels, and batteries. IEEE Trans. Green Commun. Netw. 3 (4), 901–918.

Chiaraviglio, L., Amorosi, L., Malandrino, F., Chiasserini, C.F., Dell'Olmo, P., Casetti, C., 2019. Optimal throughput management in UAV-based networks during disasters. In: INFOCOM 2019 – IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2019. pp. 307–312.

Chow, J.Y., 2016. Dynamic UAV-based traffic monitoring under uncertainty as a stochastic arc-inventory routing policy. Int. J. Transp. Sci. Technol. 5 (3), 167–185.

Chung, S.H., Sah, B., Lee, J., 2020. Optimization for drone and drone-truck combined operations: a review of the state of the art and future directions. Comput. Oper. Res. 123, 105004 .

Dille, M., Singh, S., 2013. Efficient aerial coverage search in road networks. AIAA Guidance, Navigation, and Control (GNC) Conference.

Garone, E., Naldi, R., Casavola, A., Frazzoli, E., 2010. Cooperative mission planning for a class of carrier-vehicle systems. In: Proceedings of the IEEE Conference on Decision and Control, pp. 1354–1359.

Jimenez, J.G., Chiaraviglio, L., Amorosi, L., Blefari-Melazzi, N., 2018. Multi-period mission planning of UAVs for 5G coverage in rural areas: A heuristic approach. In: Proceedings of the 2018 9th International Conference on the Network of the Future, NOF 2018. pp. 52–59.

Li, M., Zhen, L., Wang, S., Lv, W., Qu, X., 2018. Unmanned aerial vehicle scheduling problem for traffic monitoring. Comput. Ind. Eng. 122, 15–23.

Mathew, N., Smith, S.L., Waslander, S.L., 2015. Planning paths for package delivery in heterogeneous multirobot teams. IEEE Trans. Autom. Sci. Eng. 12 (4), 1298–1308.

Moshref-Javadi, M., Lee, S., 2017. Using drones to minimize latency in distribution systems. In: IE Annual Conference Proceedings, pp. 235–240.

Mourelo Ferrandez, S., Harbison, T., Weber, T., Sturges, R., Rich, R., 2016. Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. J. Ind. Eng. Manage. 9(2) (2016). doi: 10.3926/jiem.1929.

Oh, H., Shin, H.S., Tsourdos, A., White, B.A., Silson, P., 2011. Coordinated Road Network Search for Multiple UAVs using Dubins Path BT – Advances in Aerospace Guidance, Navigation and Control. Berlin, Heidelberg. Springer, Berlin Heidelberg. pp. 55–65.

Oh, H., Kim, S., Tsourdos, A., White, B.A., 2014. Coordinated road-network search route planning by a team of UAVs. Int. J. Syst. Sci. 45 (5), 825–840.

Otto, A., Agatz, N., Campbell, J., Golden, B., Pesch, E., 2018. Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: a survey. Networks 72, 1–48.

Poikonen, S., Campbell, J., 2020. Future directions in drone routing research. Networks 1–11.

Poikonen, S., Golden, B., 2020. Multi-visit drone routing problem. Comput. Oper. Res. 113, 104802 .

Puerto, J., Valverde, C., 2020. Routing for unmanned aerial vehicles. Touring Dimensional Sets.

Di Puglia Pugliese, L., Guerriero, F., 2017. Last-Mile Deliveries by Using Drones and Classical Vehicles BT – Optimization and Decision Science: Methodologies and Applications. Cham. Springer International Publishing. pp. 557–565.

Tokekar, P., Hook, J.V., Mulla, D., Isler, V., 2016. Sensor planning for a symbiotic UAV and UGV system for precision agriculture. IEEE Trans. Rob. 32 (6), 1498–1511.

Trotta, A., Andreagiovanni, F.D., Di Felice, M., Natalizio, E., Chowdhury, K.R., 2018. When UAVs ride a bus: Towards energy-efficient city-scale video surveillance. In: Proceedings – IEEE INFOCOM, 2018-April:1043–1051.

Virtanen, P., Gommers, R., Oliphant, T.E., et al., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods 17 (3), 261–272.

Wen, T., Zhang, Z., Wong, K.K.L., 2016. Multi-objective algorithm for blood supply via unmanned aerial vehicles to the wounded in an emergency situation. PLoS ONE 11 (5).

Chapter 10

An extended model of coordination of an all-terrain vehicle and a multivisit drone

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 0 (2022) 1–27
DOI: 10.1111/itor.13179

An extended model of coordination of an all-terrain vehicle and a multivisit drone

Lavinia Amorosi[a], Justo Puerto[b] and Carlos Valverde[b,*]

[a] *Department of Statistical Sciences, Sapienza University of Rome, Rome 00185, Italy*
[b] *Department of Statistical Sciences and Operational Research and IMUS, University of Seville, Sevilla 41012, Spain*
E-mail: lavinia.amorosi@uniroma1.it [Amorosi]; puerto@us.es [Puerto]; cvalverde@us.es [Valverde]

Received 30 September 2021; received in revised form 28 June 2022; accepted 30 June 2022

Abstract

In this paper, a model that combines the movement of a multivisit drone with a limited endurance and a base vehicle that can move freely in the continuous space is considered. The mothership is used to charge the battery of the drone, whereas the drone performs the task of visiting multiple targets of distinct shapes: points and polygonal chains. For polygonal chains, it is required to traverse a given fraction of its lengths that represent surveillance/inspection activities. The goal of the problem is to minimize the overall weighted distance traveled by both vehicles. A mixed integer second-order cone program is developed and strengthened using valid inequalities and giving good bounds for the Big-M constants that appear in the model. A refined matheuristic that provides reasonable solutions in short computing time is also established. The quality of the solutions provided by both approaches is compared and analyzed on an extensive battery of instances with different number and shapes of targets, which shows the usefulness of our approach and its applicability in different situations.

Keywords: routing; networks; logistics; drones; mixed integer conic programming

1. Introduction

Increasingly frequent new technologies, such as robots, self-driving vehicles, and drones, are used to replace humans in some activities, especially the most repetitive or dangerous ones (Chui et al., 2016), or to create infrastructures and service networks alternative to traditional ones (see, e.g., Amorosi et al., 2019; Chiaraviglio et al., 2019). In this context, many management and coordination problems arise, which can also be addressed and solved by means of optimization

*Corresponding author.

models. The variety of problems and applications in this area has already led to a wide scientific production and to the development of extensions of existing combinatorial optimization models (see, e.g., Di Puglia Pugliese and Guerriero, 2017) or to the formalization of new classes of problems, also by resorting to nonlinear programming as in Amorosi et al. (2021a) and Amorosi et al. (2021b). In Cavani et al. (2021), Roberti and Ruthmair (2021), and Coindreau et al. (2021), the authors study exact methods for the traveling salesman problem with one or multiple drones. An interesting literature review on drone routing problems is presented in Macrina et al. (2020). In particular, the use of drone technology in various sectors is a well-studied topic that continues to receive growing interest (see, e.g., Vidal et al., 2020; Coindreau et al., 2021; Dell'Amico et al., 2022; Mbiadou Saleu et al., 2021; Wang et al., 2022). Indeed, in a context where the urgency of sustainable solutions with low environmental impact is growing, this technology represents a valid alternative to the use of traditional means of transport. Furthermore, drones can also reach areas that are difficult to be reached by people and in a faster and safer way. Examples of this can be found both in parcel delivery and in many inspection and monitoring activities also in postdisaster contexts (for extensive surveys, see Otto et al., 2018; Chung et al., 2020; Rojas Viloria et al., 2021). In this work, we refer to these latter activities, resorting to the use of one drone supported by a mothership vehicle working as a mobile recharging station to manage its limited endurance. Such a system requires coordination and synchronization of the vehicles involved. In Amorosi et al. (2021a), the authors studied the mothership drone routing problem with graphs (AMDRPG) where one drone is supported by a mothership, and they formulated the coordination problem to visit a set of target graphs by minimizing the total distance traveled by both vehicles. In Amorosi et al. (2021b), an extension of this problem with multiple drones, called the mothership and multiple drones routing problem with graphs has been investigated. In this paper, we face the case in which one drone, supported by a mothership vehicle, must visit a set of targets but, differently with respect to the previously mentioned works, in each mission, the drone can visit more than one target or a portion of it. We assume that the mothership can move freely in the continuous space and we consider two possible shapes of the target to be visited: (i) the targets are points or (ii) the targets are polygonal chains. This is an important contribution of this work as compared with recent papers in the literature where launching and retrieving points are forced to be nodes of a given network and the targets are always points (Cavani et al., 2021; Coindreau et al., 2021; Dell'Amico et al., 2022). We mathematically formulate the problem as a mixed integer nonlinear programming (MINLP) model for which we also derive valid inequalities to reinforce it. Moreover, to deal with larger sized instances, we design alternative matheuristic procedures. Extensive computational experiments are performed reporting the usefulness of our exact and heuristic methods to solve the problem.

The rest of this paper is structured as follows. Section 2 describes the problem details and the proposed mathematical programming formulation. Section 3 discusses a strengthening of the formulation also by resorting to valid inequalities. Section 4 presents (alternative) matheuristic procedures to deal with larger sized instances of the problem. Section 5 reports the experimental results obtained by testing the model on different instances of points and polygonal chains and the comparison with the ones provided by the matheuristic algorithms to evaluate their usefulness. Finally, Section 6 concludes the paper.

2. Problem description and valid formulation

2.1. Problem description

In the multitarget mothership and drone routing problem (Multitarget-MDRP), there exists one drone that has to be coordinated with one mothership (that plays the role of the base vehicle) to complete a number of operations consisting on visiting some targets. All these targets must be visited by the drone before finishing the complete tour. Both vehicles start at a known location, denoted *orig*, then they depart to perform all the operations and, once all the targets are visited, they must return together to a final location, called *dest*. We refer to an operation as the sequence of launching the drone from the mothership, visiting one or more targets and returning to the mothership. The shapes of the targets that are considered in this paper are points and polygonal chains. A similar analysis allows to extend the model from points to convex sets as well as from polygonal chains to general graphs (see Amorosi et al., 2021a, 2021b). Nevertheless, for the sake of simplicity and to improve the readability of this paper, we restrict ourselves to the abovementioned cases that already capture the essence of the problem. The operation of visiting a point consists in getting to it and coming back, whereas for polygonal chains the drone has to traverse a given fraction of its length for considering a successful visit. In Amorosi et al. (2021a) and Puerto and Valverde (2021), the idea of traversing a polygonal chain is discussed, and the authors consider two different modes: (1) visiting a fraction of the total length of the graph and (2) traversing a given fraction of the length of each one of its segments. From an application point of view, the reader may understand these operations as reliability inspections so that testing a given fraction of the target suffices to certify a correct operation. Looking at the difficulty of these operation modes, the computational results reported in those works suggest that there is not a meaningful difference in terms of the difficulty induced by the considered mode of visit. Hence, for the sake of simplicity, in this paper we consider only a simpler form of the first case where the drone, once it enters the polygonal chain, has to traverse the entire required fraction before leaving the target. In other words, no preemption is allowed. We also assume that the mothership and the drone travel at constant velocities v_M and v_D, respectively, although it can be extended to more general cases where these velocities can be modeled as a time-dependent function. Moreover, the drone has a limited time N (endurance) to complete each operation and return to the base vehicle to recharge batteries. We assume that the drone and the base vehicle movements follow straight lines on a continuous space. This assumption can model the case where the base vehicle is a helicopter or a boat, so that there are no obstacles or restrictions to its movement. Nowadays, this type of system consisting of a boat and a fleet of drones is used, for example, by coast guards to perform surveillance activities to identify immigrants that need help in the sea (see AltiGator, 2015). This implies that Euclidean distance is used to measure displacements.

Figure 1 shows an example of the problem framework, where the black squares represent the origin and the destination of the mothership tour.

Moreover, at each operation, the drone must be launched from the base vehicle (the launching points have to be determined) and it must be retrieved when its battery needs to be recharged (the rendezvous points also have to be determined). It is assumed that the time spent by the drone to visit the targets associated with an operation must be lower than or equal to the time that the mothership

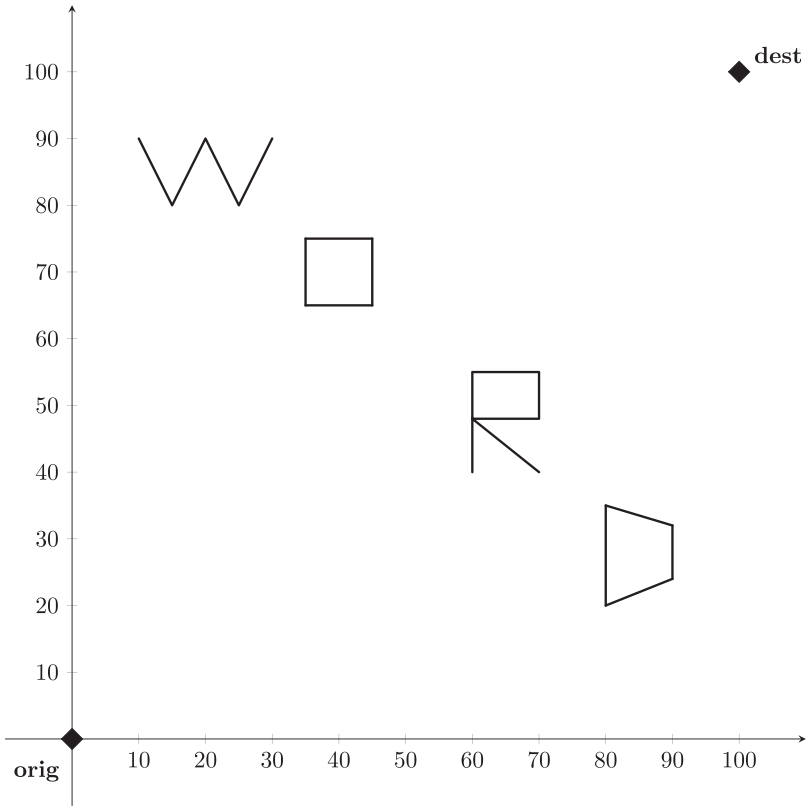


Fig. 1. Example of problem instance with polygonal targets.

needs to move from the launching point to the rendezvous point for this operation. Furthermore, it is supposed that the cost produced by the drone's trip is lower compared to that incurred by the base vehicle. Therefore, the goal is to minimize the weighted total distance traveled by the mothership and the drone. Some works assume that this cost is negligible in comparison with the mothership (Amorosi et al., 2021a). The reader may note that the extension not including the distances traveled by the drone in the objective function is straightforward by setting the corresponding weight to zero.

The goal of the Multitarget-MDRP is to find the launching and rendezvous points of the drone satisfying the visit requirements for the targets in $\mathcal{T}$ and minimizing the weighted length of the paths traveled by the mothership and the drone.

2.2. Mixed Integer Nonlinear Programming Formulations

The purpose of this section is to present an MINLP formulation for the Multitarget-MDRP that can be applied to solve instances of this problem.

Table 1
Nomenclature for the Multitarget-MDRP

Problem parameters		
Parameter name	Range	Description
orig	$\mathbb{R}^2$	Coordinates of the point defining the origin of the mothership path (or tour).
dest	$\mathbb{R}^2$	Coordinates of the point defining the destination of the mothership path (or tour).
$\mathcal{B}$	$\{b_1, \ldots, b_{\lvert\mathcal{B}\rvert}\}$	Set of points.
$b = (b(x_1), b(x_2))$	$\mathbb{R}^2$	Coordinates of the point $b \in \mathcal{B}$.
$\mathcal{P}$	$\{p_1, \ldots, p_{\lvert\mathcal{P}\rvert}\}$	Set of polygonal chains.
$p = (V_p, S_p)$	$\mathbb{R}^2$	Set of breakpoints and segments of each polygonal chain $p \in \mathcal{P}$.
$V = B \cup (\bigcup_{p\in\mathcal{P}} V_p)$	$\mathbb{R}^2$	Set of target points and set of breakpoints of polygonal targets/
$\mathcal{L}(p)$	$\mathbb{R}_+$	Total length of the polygonal chain $p \in \mathcal{P}$.
$\mathcal{L}(s_p)$	$\mathbb{R}_+$	Length of the segment s_p of the polygonal chain $p \in \mathcal{P}$.
B^{v_p}	$\mathbb{R}^2$	Coordinates of the point v_p of the polygonal $p \in \mathcal{P}$.
$\mathcal{T} = \mathcal{B} \cup \mathcal{P}$	$\{b_1, \ldots, b_{\lvert\mathcal{B}\rvert}, p_1, \ldots, p_{\lvert\mathcal{P}\rvert}\}$	Set of targets.
$\mathcal{O}$	$\{o_1, \ldots, o_{\lvert\mathcal{O}\rvert}\}$	Set of operations.
α^t	[0, 1]	Fraction of the target that must be visited. If t is a point, $\alpha^t = 0$.
v_δ	$\mathbb{R}_+$	Drone speed.
v_M	$\mathbb{R}_+$	Mothership speed.
ω_δ	$\mathbb{R}_+$	Weighting factor for the distance covered by the drone.
ω_M	$\mathbb{R}_+$	Weighting factor for the distance covered by the mothership.
N	$\mathbb{R}_+$	Drone endurance.
M	$\mathbb{R}_+$	Big-M constant.
m	$\mathbb{R}_+$	Small-M constant.

In Section 2.1, we mention that the mothership can move without any restriction in a continuous space that for simplicity is supposed to be $\mathbb{R}^2$. Although it is possible to measure distances with any l_τ-norm, $1 \leq \tau \leq \infty$ (see Blanco et al., 2017), for the sake of presentation, in this work the distances are measured by the Euclidean norm ($\tau = 2$).

First of all, we introduce the parameters or initial data that formally describe the problem and that are summarized in Table 1.

Tables 2 and 3 detail the set of binary and continuous decision variables that appear in the formulation, respectively.

To represent the movement of the drone within a polygonal $p \in \mathcal{P}$, we proceed to introduce some notation related to p. Let $p = (V_p, S_p)$ be a polygonal chain in $\mathcal{P}$ whose total length is denoted by $\mathcal{L}(p)$. Here, V_p denotes the set of vertices and S_p denotes the set of segments connecting pairs of consecutive vertices whose cardinality is $|S_p|$. Let $s_p = \overline{v_p(v+1)_p}$ be the segment s of the polygonal $p \in \mathcal{P}$ and let $\mathcal{L}(s_p)$ be its length. Since we need to refer to interior points of the segment s_p, these continuum of points is parameterized by the two endpoints $B^{v_p} = (B^{v_p}(x_1), B^{v_p}(x_2))$ and $B^{(v+1)_p} = (B^{(v+1)_p}(x_1), B^{(v+1)_p}(x_2))$ of the segment: $x \in [B^{v_p}, B^{(v+1)_p}]$ if and only if $\exists\ \gamma \in [0, 1]$ such that $x = \gamma B^{v_p} + (1-\gamma)B^{(v+1)_p}$. Hence, the length of the segment s_p is $\mathcal{L}(s_p) = \|B^{v_p} - B^{(v+1)_p}\|$.

Table 2
Summary of binary variables used in the mathematical programming model

Binary decision variables			
Name	Set	Domain	Description
$\mu_A^{s_p}$	$s_p \in S_p : p \in \mathcal{P}$	Binary	1, if the arrival point on the polygonal chain is located in the line segment s_p, 0, otherwise.
$\mu_D^{s_p}$	$s_p \in S_p : p \in \mathcal{P}$	Binary	1, if the departure point on the polygonal chain is located in the line segment s_p, 0, otherwise.
$z_p^{s_p s'_p}$	$s_p, s'_p \in S_p : p \in \mathcal{P}$	Binary	1, if the arrival and departure points are located in the line segments s_p and s'_p, respectively, 0, otherwise.
u_L^{to}	$t \in \mathcal{T}, o \in \mathcal{O}$	Binary	1, if the drone starts the operation o in the target t, 0, otherwise.
u_R^{to}	$t \in \mathcal{T}, o \in \mathcal{O}$	Binary	1, if the drone finishes the operation o in the target t, 0, otherwise.
χ^{to}	$t \in \mathcal{T}, o \in \mathcal{O}$	Binary	1, if the drone visits the target t in the operation o, 0, otherwise.
$y^{tt'o}$	$t, t' \in \mathcal{T}, o \in \mathcal{O}$	Binary	1, if the drone goes from the target t to the target t' in the operation o, 0, otherwise.

Next, we need to determine the placement of the arrival point, A^p, on the polygonal chain p introducing the following inequalities for each $p \in \mathcal{P}$:

$$A^p \in p \iff \begin{cases} \gamma_A^{1_p} & \leq \mu_A^{1_p}, \\ \gamma_A^{s_p} & \leq \mu_A^{s_p-1} + \mu_A^{s_p}, \quad s_p \in S_p \setminus \{1\}, \\ \gamma_A^{|V_p|_p} & \leq \mu_A^{|S_p|_p}, \\ \displaystyle\sum_{s_p \in S_p} \mu_A^{s_p} = 1, & \\ \displaystyle\sum_{v_p \in V_p} \gamma_A^{v_p} = 1, & \\ A^p & = \displaystyle\sum_{v_p \in V_p} \gamma_A^{v_p} B^{v_p}. \end{cases} \tag{$\mathcal{P}$-C}$$

The first, second, and third inequalities link $\mu_A^{s_p}$ and $\gamma_A^{s_p}$ variables: they state that the variable $\gamma_A^{s_p}$ that gives the representation of a point A^p on the line segment s_p is active (nonnull) only if this line segment is chosen to enter in polygonal p, that is, $\mu_A^{s_p} = 1$. The fourth equation sets that only one line segment is chosen for entering each polygonal chain. Finally, the fifth and sixth equations set the representation of A^p as a convex combination of the extreme points of the selected line segment. In the same way, we can model the location of the departure point, D^p, by using the variables D^p, $\mu_D^{s_p}$, and $\gamma_D^{s_p}$ explained in Tables 2 and 3, respectively.

In our approach to model the Multitarget-MDRP we use operations identified with the order in which the different elements in the problem are visited. Let us denote by $\mathcal{O}$ the set of operations that the mothership and the drone have to carry out. An operation $o \in \mathcal{O}$ is referred to as the one in which the mothership launches a drone from a taking-off location, denoted by x_L^o and later it takes it back on a rendezvous location x_R^o. Each operation consists in the drone visit to one or more targets

Table 3
Summary of continuous variables used in the mathematical programming model

Continuous decision variables			
Name	Set	Domain	Description
A^t	$t \in \mathcal{T}$	$\mathbb{R}^2$	Coordinates representing the arrival point on the target t. If the target is a point, it coincides with D^t.
$\gamma_A^{v_p}$	$v_p \in V_p : p \in \mathcal{P}$	[0, 1]	Parameterization of the arrival point A^p on the segment $s_p = \overline{v_p(v+1)_p}$ of the polygonal chain.
D^t	$t \in \mathcal{T}$	$\mathbb{R}^2$	Coordinates representing the departure point on each target. If the target is a point, it coincides with A^t.
$\gamma_D^{v_p}$	$v_p \in V_p : p \in \mathcal{P}$	[0, 1]	Parameterization of the departure point D^p on the segment $s_p = \overline{v_p(v+1)_p}$ of the polygonal chain.
x_L^o	$o \in \mathcal{O}$	$\mathbb{R}^2$	Coordinates representing the point where the mothership launches the drone at operation o.
x_R^o	$o \in \mathcal{O}$	$\mathbb{R}^2$	Coordinates representing the point where the mothership retrieves the drone at operation o.
d_L^{to}	$t \in \mathcal{T}, o \in \mathcal{O}$	$\mathbb{R}_+$	Distance traveled by the drone from the launching point x_L^o on the mothership to the first target point A^t associated to the operation o.
$d_{\text{out}}^{tt'}$	$t, t' \in \mathcal{T}$	$\mathbb{R}_+$	Distance traveled by the drone from the departure point D^t on one target to the arrival point $A^{t'}$ on another one.
d_R^{to}	$t \in \mathcal{T}, o \in \mathcal{O}$	$\mathbb{R}_+$	Distance traveled by the drone from the departure point of the last visited target D^t to the rendezvous point x_R^o associated to the operation o.
d_{LR}^o	$o \in \mathcal{O}$	$\mathbb{R}_+$	Distance traveled by the drone from the launching point x_L^o to the rendezvous point x_R^o at operation o.
d_{RL}^o	$o \in \mathcal{O} \setminus \{\lvert\mathcal{O}\rvert\}$	$\mathbb{R}_+$	Distance traveled by the mothership and the drone from the rendezvous point x_R^o at operation o to the launching point x_L^{o+1} at the operation $o+1$.
$d_p^{s_p s'_p}$	$s_p, s'_p \in S_p : p \in \mathcal{P}$	$\mathbb{R}_+$	Distance traveled by the drone from the departure point on the segment s_p on one polygonal to the arrival point on the segment s'_p of the same polygonal.
d_{in}^t	$t \in \mathcal{T}$	$\mathbb{R}_+$	Distance traveled by the drone from the arrival point A^t to the departure point D^t on the same target. If the target is a point, this distance is 0.

in $\mathcal{T}$ with the required constraints. At this point, it is relevant to note that the pair of locations x_L^o and x_R^o must be selected in the plane where the mothership is presumed to move. Observe that $|\mathcal{O}| \leq |\mathcal{T}|$ because of the assumption that at least one target is visited for each operation.

Figure 1 shows an example of a problem instance with four polygonal targets. The point *orig* from where the mothership together with the drone must start their tour, is located in the origin, while the destination point *dest* is the point (100,100).

To include the definition of the paths followed by the drone in our mathematical programming formulation, we need to make decisions to choose:

- the optimal assignment of targets to each operation o;
- the optimal order to visit targets for its corresponding operation o.

We can model the route followed by the drone using the binary variables u_L^{to}, $y^{tt'o}$, χ^{to}, and u_R^{to} defined in Table 2:

$$\sum_{t\in\mathcal{T}} u_L^{to} \leq 1, \quad \forall o \in \mathcal{O}, \tag{1}$$

$$\sum_{t\in\mathcal{T}} u_R^{to} \leq 1, \quad \forall o \in \mathcal{O}, \tag{2}$$

$$\sum_{o\in\mathcal{O}} \chi^{to} = 1, \quad \forall t \in \mathcal{T}, \tag{3}$$

$$\chi^{to} - u_L^{to} = \sum_{t'\neq t} y^{t'to}, \quad \forall t \in \mathcal{T},\ \forall o \in \mathcal{O}, \tag{4}$$

$$\chi^{to} - u_R^{to} = \sum_{t'\neq t} y^{tt'o}, \quad \forall t \in \mathcal{T},\ \forall o \in \mathcal{O}, \tag{5}$$

$$\sum_{t,t'\in S} y^{tt'o} \leq |S| - 1, \quad \forall S \subsetneq \mathcal{T},\ \forall o \in \mathcal{O}. \tag{6}$$

Constraints (1) and (2) state that for each operation the drone can only enter and exit, respectively, by one target. Constraints (3) ensure that every target will be visited in some operation. Constraints (4) assure that if target t is visited by the drone for the operation o, one of two alternative conditions must take place: either t is the first target for the operation o or target t is visited by the drone after visiting another target t' for the operation o. Similarly, constraints (5) state that if the target t for the operation o is visited by the drone, either t is the last target of the operation, or the drone must move to another target t' of the operation o after visiting target t. Finally, inequalities (6) are the subtour elimination constraints (SEC) applied to each operation. Note that the complete family of SEC constraints cannot be included in the model when we implement this formulation, because there is an exponential number of them and it can induce a memory problem on-the-shelf solvers. This problem can be solved by performing a row generation procedure that includes the constraints whenever they are needed by a separation oracle. To detect these SEC inequalities, as usual, we look for disconnected components in the current solution. Among them, we choose the shortest subtour found in the solution to be added as a lazy constraint to the model to make the formulation easier to be solved.

To take into account the different distances among the decision variables of the model, we need to set the continuous variables d_L^{to}, $d_{\text{out}}^{tt'}$, d_{in}^{t}, d_R^{to}, d_{RL}^{o}, and d_{LR}^{o}, defined in Table 3. This can be done by means of the following constraints:

$$\|x_L^o - A^t\| \leq d_L^{to}, \quad \forall t \in \mathcal{T}, \forall o \in \mathcal{O}, \tag{DIST$_1$}$$

$$\|A^t - D^{t'}\| \leq d_{\text{out}}^{tt'}, \quad \forall t, t' \in \mathcal{T}, \tag{DIST$_2$}$$

$$\|D^t - x_R^o\| \leq d_R^{to}, \quad \forall t \in \mathcal{T}, \forall o \in \mathcal{O}, \tag{DIST$_3$}$$

$$\|x_L^o - x_R^o\| \leq d_{LR}^{o}, \quad \forall o \in \mathcal{O}. \tag{DIST$_4$}$$

$$\|x_R^o - x_L^{o+1}\| \leq d_{RL}^{o}, \quad \forall o \in \mathcal{O} \setminus \{|\mathcal{O}|\}. \tag{DIST$_5$}$$

Note that d^t_{in} is zero when the target is a point. However, to compute the distance inside the polygonal, we need to set the following expressions for each $p \in \mathcal{P}$:

$$d_p^{s_p s'_p} = \begin{cases} |\gamma_A^{s_p} - \gamma_D^{s'_p}| \mathcal{L}(s_p), & \text{if } s_p = s'_p, \\ (1 - \gamma_D^{s_p})\mathcal{L}(s_p) + \sum_{s''=s_p+1}^{s'_p - 1} \mathcal{L}(s'') + \gamma_A^{s'_p} \mathcal{L}(s'_p), & \text{if } s_p < s'_p, \\ \gamma_D^{s_p} \mathcal{L}(s_p) + \sum_{s''=s'_p+1}^{s_p - 1} \mathcal{L}(s'') + (1 - \gamma_A^{s'_p})\mathcal{L}(s'_p), & \text{if } s_p > s'_p. \end{cases} \tag{d$\mathcal{P}$}$$

This expression needs to define a binary variable z_p that determines whether d_p is defined by the first, the second or the third expression in the formula:

$$z_p^{s_p s'_p} = \mu_A^{s_p} \mu_D^{s'_p}.$$

Finally, we can compute the total distance between the departure and the arrival points in each polygonal $p \in \mathcal{P}$:

$$d^p_{\text{in}} = \sum_{s_p, s'_p \in S_p} d_p^{s_p s'_p} z_p^{s_p s'_p}. \tag{DIST$_6$}$$

Moreover, to include the requirement of visiting a fraction α^p of each polygonal chain p, we need to impose that

$$d^p_{\text{in}} \geq \alpha^p \mathcal{L}(p). \tag{$\alpha - \mathcal{P}$}$$

There exists a special case of the above condition, when all segments of the polygonal chain have the same length, which enables a simplified representation. We refer the interested reader to the Appendix A for further details of these constraints.

The coordination between the drone and the mothership must ensure that the time spent by the drone to do the operation o is less than or equal to the time that the mothership needs to move from the launching point to the rendezvous point during this operation. To this end, we include the following coordination constraint for each operation $o \in \mathcal{O}$:

$$\frac{1}{v_\delta} \left(\sum_{t \in \mathcal{T}} u_L^{to} d_L^{to} + \sum_{t,t' \in \mathcal{T}} y^{tt'o} d^{tt'}_{\text{out}} + \sum_{t \in \mathcal{T}} \chi^{to} d^t_{\text{in}} + \sum_{t \in \mathcal{T}} u_R^{to} d_R^{to} \right) \leq \frac{d^o_{LR}}{v_M}. \tag{DCW}$$

Eventually, we have to impose that the tour of the mothership, together with the drone, starts from the origin *orig* and ends at the destination *dest*. This is ensured by including the following constraints:

$$x_L^0 = orig, \tag{ORIG$_1$}$$

$$x_R^0 = orig, \tag{ORIG$_2$}$$

$$x_L^{|\mathcal{O}|+1} = dest, \tag{DEST$_1$}$$

$$x_R^{|\mathcal{O}|+1} = dest. \tag{DEST$_2$}$$

Observe that one of the addends of the objective function of this problem minimizes the right-hand side of (DCW). Thus, this constraint will become an equality and thus, it is able to model the time endurance restriction for a particular operation $o \in \mathcal{O}$ by limiting the space traveled by the mothership for this operation:

$$d_{LR}^o \leq N. \tag{Endurance}$$

The goal of the Multitarget-MDRP is to find a feasible solution that minimizes the total weighted distance traveled by the mothership and the drone. The following formulation, which includes all the constraints explained before, gives an exact model for this problem:

$$\begin{aligned}
\min \quad & \omega_\delta \left(\sum_{t\in\mathcal{T}} \sum_{o\in\mathcal{O}} u_L^{to} d_L^{to} + \sum_{t\in\mathcal{T}} \sum_{o\in\mathcal{O}} \chi^{to} d_{\text{in}}^t + \sum_{t\neq t'\in\mathcal{T}} \sum_{o\in\mathcal{O}} y^{tt'o} d_{\text{out}}^{tt'} + \sum_{t\in\mathcal{T}} \sum_{o\in\mathcal{O}} u_R^{to} d_R^{to} \right) \\
& + \omega_M \left(\sum_{o\in\mathcal{O}} d_{LR}^o + \sum_{o\in\mathcal{O}\setminus\{|\mathcal{O}|\}} d_{RL}^o \right)
\end{aligned} \tag{Multitarget $-$ MDRP}$$

$$\begin{aligned}
\text{s.t.} \quad & (1)-(6), \\
& (\mathcal{P}\text{-C}), (\alpha\text{-}\mathcal{P}), \\
& (DCW), \\
& (Endurance), \\
& (d\mathcal{P}), \\
& (DIST1)-(DIST6), \\
& (ORIG1)-(DEST2).
\end{aligned}$$

The objective function describes the weighted distances traveled by the drone and the mothership, respectively. Constraints (1)–(6) model the path made by the drone; ($\mathcal{P}$-C) and ($\alpha - \mathcal{P}$) describe the location of the arrival and departure points and the requirement of visiting an α fraction for the polygonal chain $p \in \mathcal{P}$, respectively. Constraint (d$\mathcal{P}$) computes the distance between each pair of segments in the polygonal chain. Constraints (DIST$_1$)–(DIST$_6$) set the variables d_L^{to}, $d_{\text{out}}^{tt'}$, d_R^{to} , d_{LR}^o, d_{RL}^o, and d_{in}^t, defined in Table 3, which set the Euclidean distances required in the model.

Note that, to handle the bilinear terms that appear in the objective function and in the (DCW) constraint, we use McCormick's envelope to linearize these terms by including variables $q \geq 0$

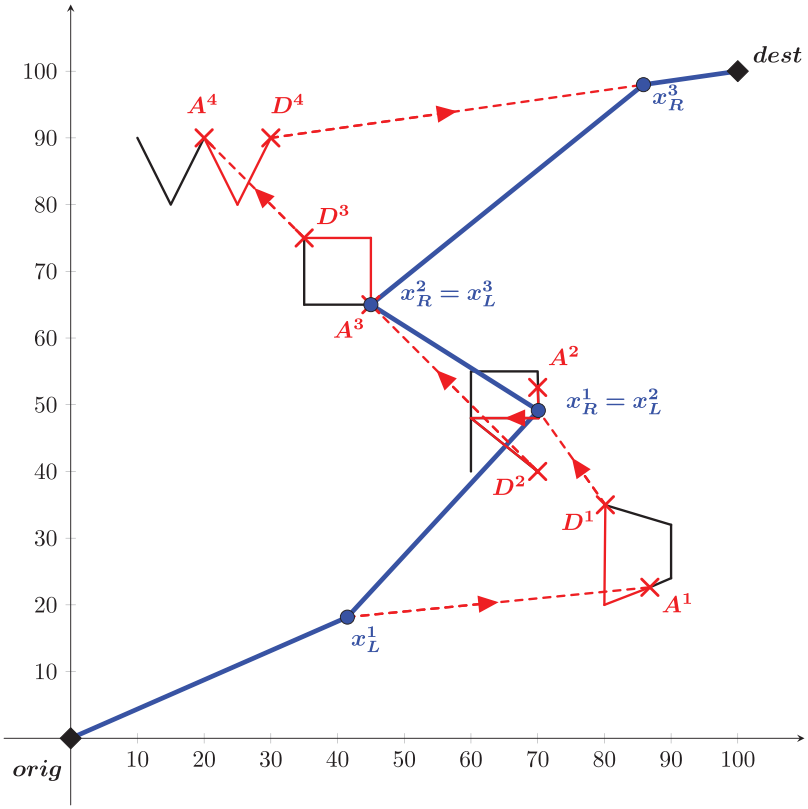


Fig. 2. Optimal solution obtained for the data of the example.

representing the products and introducing the following constraints:

$$q \leq Mz,$$
$$q \leq d,$$
$$q \geq mz,$$
$$q \geq d - M(1 - z),$$

where m and M are, respectively, the lower and upper bounds of the distance variable d. These bounds will be tightened for each bilinear term in Section 3.

Figure 2 shows an example of the solution obtained by means of the exact method solving the formulation. We run the model on the same example of Fig. 1 and we obtained the optimal solution consisting in the mothership tour, represented by the bold polygonal chain starting from the origin *orig*, ending at the destination *dest*, and with drone movements represented with the dotted segments. The mothership launches the drone for its first operation from x_L^1. The drone flies to the arrival point A^1 for visiting a portion (50%) of the first target. It leaves the first target at point D^1 and meets the mothership at point x_R^1. This point is also the launching point from where the drone

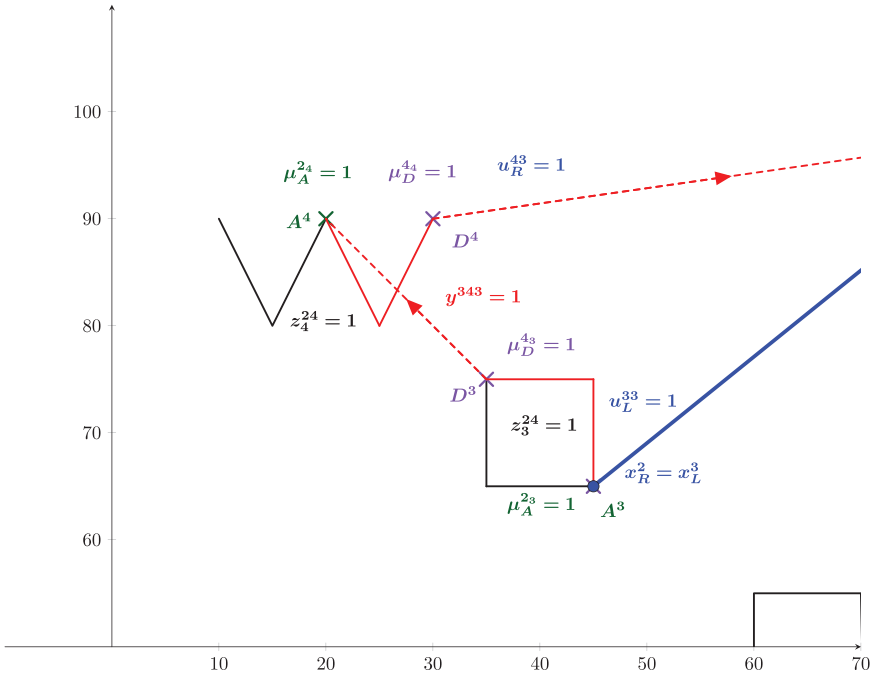


Fig. 3. Binary decision variables associated to each target.

starts its second operation by visiting the second target. From that point, it flies to point A^2 and traverses the portion of the second target from A^2 to D^2. Then, the drone flies to point x_R^2 where it meets again the mothership. From this point, the drone starts its last operation by visiting the third and the fourth targets. Indeed, after visiting the third target from point A^3 to point D^3, it directly flies to point A^4 of the last target. It visits the required portion of this last target from point A^4 to D^4 and then meets the mothership at point x_R^3. The mothership and the drone complete their service at the destination *dest*.

Figure 3 shows a zoom on the last two targets of the example of the solution reported in Fig. 2. We can visualize in detail the values of the different binary variables introduced in the formulation to define the order of visit of the targets, the launching and rendezvous points and thus the mothership and drone tours. In particular, variable $u_L^{33}=1$ indicates that the third operation of the drone starts in the third target from point A^3. This point is located on the line segment 2 of the polygonal target 3, $\mu_A^{2_3}=1$, and it is also the rendezvous point x_R^2 of the second operation. The visit on the third target ends at the departure point D^3 that is located on the line segment 4 of the polygonal target 3, $\mu_A^{4_3}=1$. Indeed, the visit of target 3 starts in segment 2 and ends in segment 4, $z_3^{24}=1$. From this point, always during the third operation, the drone directly flies to the last target, that is, target 4, $y^{343}=1$. The visit of the target 4 starts from the point A^4, located on the line segment 2, $\mu_A^{2_4}=1$, and ends at the departure point D^4, located on the line segment 4, $\mu_A^{4_4}=1$. Eventually, the third and last operations of the drone finish on target 4, $u_R^{43}=1$.

3. Strengthening the formulation of Multitarget-MDRP

3.1. Preprocessing

In this subsection, we explore the nature of the problem to fix *a priori* some variables and to increase the efficiency of the model. Particularly, the following proposition allows to fix $y^{tt'o}$ binary variables to zero.

Proposition 1. *Let $t, t' \in \mathcal{T}$ be two targets. Let $d_{min}(t, t')$ denote the minimum distance between them and $length(t)$, the length of the target t. If t, t' verify that*

$$\alpha^t\, length(t) + d_{min}(t, t') + \alpha^{t'}\, length(t') > \frac{v_\delta}{v_M} N, \tag{7}$$

then the drone cannot visit in a single operation t and t'.

Proof. Let us assume that the drone can go from t to t' in the operation o. Then it must satisfy (Endurance) and (DCW) constraints:

$$\sum_{t\in\mathcal{T}} u_L^{to} d_L^{to} + \sum_{t,t'\in\mathcal{T}} y^{tt'o} d_{\text{out}}^{tt'} + \sum_{t\in\mathcal{T}} \chi^{to} d_{\text{in}}^{t} + \sum_{t\in\mathcal{T}} u_R^{to} d_R^{to} \le \frac{v_\delta}{v_M} N.$$

Since $d_L^{to}, d_R^{to} \ge 0$, the left-hand side of this inequality can be bounded from below by the left-hand side of (7):

$$\alpha^t\, length(t) + d_{\min}(t, t') + \alpha^{t'}\, length(t') \le \sum_{t\in\mathcal{T}} u_L^{to} d_L^{to} + \sum_{t,t'\in\mathcal{T}} y^{tt'o} d_{\text{out}}^{tt'} + \sum_{t\in\mathcal{T}} \chi^{to} d_{\text{in}}^{t} + \sum_{t\in\mathcal{T}} u_R^{to} d_R^{to},$$

which is impossible because each side is lower (resp., upper) bounded by $\frac{v_\delta}{v_M} N$. □

3.2. Valid inequalities for the Multitarget-MDRP

In this subsection, we introduce some valid inequalities for Multitarget-MDRP that strengthen the formulation presented in Section 2.2. In addition, the constraint that coordinates the movement of the drone and the mothership, namely (DCW), and the objective function of the model hold the products of binary and continuous variables. Each of these products generates Big-M constants that must be tightened when they are linearized. The present section provides some bounds for these constants.

For this problem, we are assuming that the drone endurance suffices to visit more than one target in the same operation because, otherwise, the problem is similar to (AMDRPG) that has been already considered in Amorosi et al. (2021a). Hence, provided that there exists an operation in which the drone visits two or more targets, the mothership does not need to perform $|\mathcal{T}|$ different operations. This idea can be used to compactify all operations made by the drone for the first operations, avoiding void tasks in $\mathcal{O}$.

Let β^o be a binary variable that attains the value 1 if the entire set of targets is visited when the operation o starts, and 0, otherwise. Observe that, if the drone has traversed all targets before the

operation o then they are also traversed before starting the operation $o+1$. Therefore, β variables must fulfill the following constraints:

$$\beta^o \leq \beta^{o+1}, \text{ for all } o = 1, \ldots, |\mathcal{O}| - 1. \tag{Monotonicity}$$

Let k^o represent the number of targets that are visited at the operation o. χ variables can be used to compute this number because χ^{to} attains the value one when the target t is visited in the operation o. Thus:

$$k^o = \sum_{t \in \mathcal{T}} \chi^{to}.$$

Thus, if β^o is one, the full set $\mathcal{T}$ must have been visited before the operation o:

$$\sum_{o'=1}^{o-1} k^{o'} \geq |\mathcal{T}|\beta^o, \tag{VI-1}$$

where $|\mathcal{T}|$ stands for the cardinality of $\mathcal{T}$.

To reduce the space of feasible solutions, it is possible to assume, without loss of generality, that it is not allowed to have an operation o without any visit if the drone still has to visit some targets. To enforce that, we can set the following constraints:

$$k^o \geq 1 - \beta^o. \tag{VI-2}$$

Following the idea given in Proposition 1 of the previous subsection, we can set some valid inequalities that indicate that the drone cannot visit a subset $S \subset \mathcal{T}$ of targets because of the (Endurance) constraint. Let $\mathcal{S}$ be the collection of subsets of $\mathcal{T}$ that do not verify (Endurance), then:

$$\sum_{t \in S} \chi^{to} \leq |S| - 1, \qquad \forall S \in \mathcal{S}, \quad \forall o \in \mathcal{O}. \tag{VI-3}$$

We can construct $\mathcal{S}$ by fixing the χ variables and partially solving (Multitarget-MDRP) for each subset of $\mathcal{T}$, which is computationally expensive. However, it is sufficient to find sets of targets of minimum cardinality such that the drone endurance does not permit to visit all their elements. Indeed, any other subset of targets that contains these minimal sets cannot be visited in a single drone operation.

The different models that we have proposed include in one way or another Big-M constants. We have defined different Big-M constants along this work. To strengthen formulations, we provide tight upper and lower bounds for those constants.

3.2.1. Big M constants bounding the distance from the launching/rendezvous point on the path followed by the mothership to the arrival/departure point on the target $t \in \mathcal{T}$

To linearize the first addend of the objective function in Multitarget-MDRP, we set the auxiliary nonnegative continuous variables q_L^{to} (resp. q_R^{to}) to model the product by inserting the following

inequalities:

$$q_L^{to} \geq m_L^{to} u_L^{to},$$
$$q_L^{to} \geq d_L^{to} - M_L^{to}(1 - u_L^{to}).$$

The best upper bound M_L^{to} or M_R^{to} is the full diameter of the data, that is, the maximum distance between every pair of vertices of the targets in $\mathcal{T}$, that is, every point that must be determined is inside the circle whose diametrically opposite points are explained below:

$$M_L^{to} = \max_{\{v, v' \in V\}} \|v - v'\| = M_R^{to}.$$

Conversely, the minimum distance in this problem can be zero. This bound is attainable whenever the launching or rendezvous points of the mothership are the same that the arrival or departure point on a given target.

3.2.2. *Bounds on the big M constants for the distance from the departure to the arrival points on the operation $o \in \mathcal{O}$*

When the drone travels in the operation o, it has to go from one target t to another target t' depending on the order given by $y^{tt'o}$. This fact produces another product of variables linearized by the following constraints:

$$q^{tt'o} \geq m^{tt'} y^{tt'o},$$
$$q^{tt'o} \geq d_{\text{out}}^{tt'} - M^{tt'}(1 - y^{tt'o}).$$

The evaluation of the bounds appearing in these constraints presents three cases depending on the structure of the targets:

- If both targets t, t' are points, then the distance is fixed and we can set

$$M^{tt'} = \|A^t - A^{t'}\| = m^{tt'}.$$

- If one target t is a point and the other t' is a polygonal chain, we can compute the minimum distance as a minimum distance point-to-set problem:

$$m^{tt'} = \min \|A^t - x\|$$
$$\text{s.t.} x \text{ verifies } (P - C).$$

On the other hand, the maximum distance between these targets can be obtained by taking the maximum of the distance between the point t and the breakpoints of the polygonal chain t':

$$M^{tt'} = \max_{v \in V_{t'}} \|v - A^t\|.$$

- If both targets t, t' are polygonal chains, it is also possible to compute exactly the minimum distance:

$$m^{tt'} = \min \|x' - x\|$$
$$\text{s.t.} x, x' \text{ verifies } (P - C).$$

On the other hand, to estimate the maximum distance we can repeat the procedure described in the previous case, but now for each breakpoint of the first polygonal chain. Then, taking the maximum of the maximum distances for each breakpoint, we get an upper bound for $M^{tt'}$:

$$M^{tt'} = \max_{v \in V_t, v' \in V_{t'}} \|v - v'\|.$$

4. A matheuristic for the Multitarget-MDRP

This section is devoted to present our matheuristic approach to provide good feasible solutions of the Multitarget-MDRP. Our motivation comes from the fact that the exact method based on the mathematical programming formulation presented in the previous section can be highly time demanding. Alternatively, the matheuristic provides a good quality solution in limited computing times.

Assuming that the drone has enough endurance to visit every target, the basic idea of the algorithm is to associate each target to one operation by solving a crossing postman problem with neighbors (XPPN) (see Puerto and Valverde, 2021) for the targets including *orig* and *dest*. Recall that the XPPN consists in finding a minimum total route that visits all the neighborhoods and traverses some fractions of the polygonal chains considered in the problem. In Puerto and Valverde (2021), a mathematical programming formulation that uses connectivity properties is used to develop a branch-and-bound procedure to solve this problem. The motivation of this approach comes from the results in Puerto and Valverde (2021), which show that the XPPN is easily solvable for medium-sized instances provided that the neighbors are points or polygonal chains. In the following, we present the pseudocode of this algorithm:

STEP 1 (Order of visit targets)
Compute the order of visits by solving the XPPN for the targets of the problem including *orig* as the first point in the tour and *dest* as the last one and associate each target $t \in \mathcal{T}$ to one operation $o \in \mathcal{O}$ in the given order.

STEP 2 (Solution of the Multitarget-MDRP model by fixing an initial partial solution)
Set the values of the binary variables u_L^{to}, u_R^{to}, χ^{to} and $y^{tt'o}$ provided by the solution of STEP 1 and solve the resulting Multitarget-MDRP model to obtain a complete feasible solution.

It is possible to refine the previous algorithm by slightly modifying STEP 2. Indeed, after STEP 1, starting from the first visited target, according to the order provided by the XPPN solution, we can iteratively append the next target to the same operation, if the drone endurance allows it. In this way, the number of operations can be reduced and a better initial partial solution can be provided to start STEP 2.

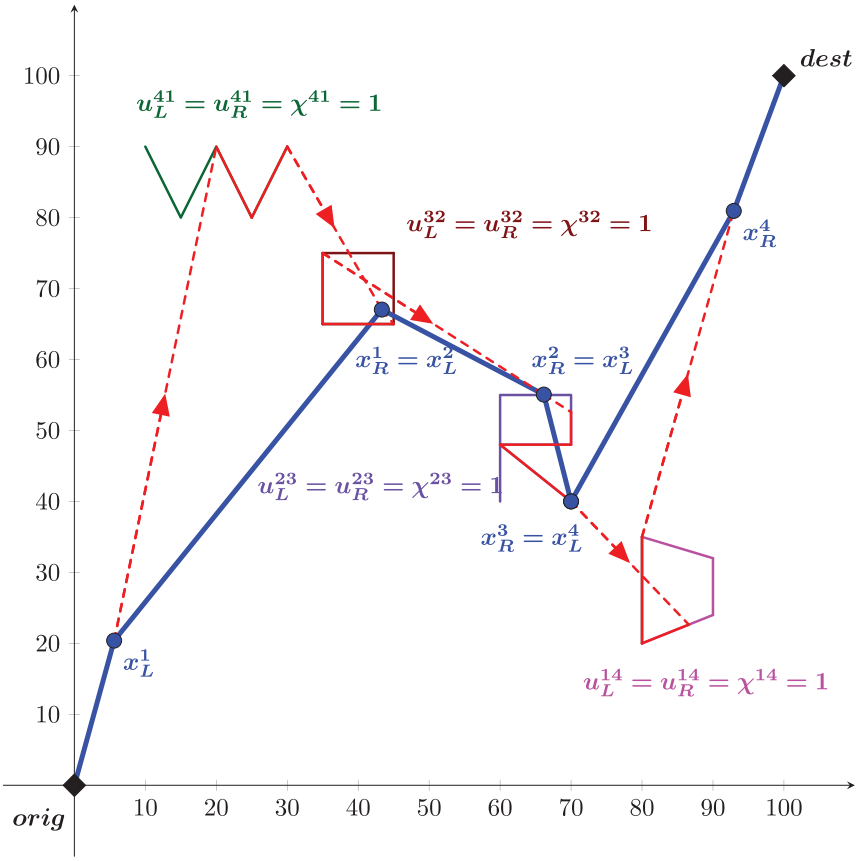


Fig. 4. Solution provided by the matheuristic.

Figure 4 shows the solution obtained by means of the matheuristic, for the same example of Fig. 1. In particular, the upper subfigure reports the solution after STEP 2 in its original form. We can observe that the solution consists in the mothership tour, represented with the bold polygonal chain starting from the origin and ending at the destination, and in four operations of the drone. Indeed, the drone first visits the 50% of the target with label 4 ($u_L^{41} = u_R^{41} = 1$), then flies to meet the mothership at the retrieve point x_R^1 and from there starts its second operation to visit a 50% of the target with label 3 ($u_L^{32} = u_R^{32} = 1$). After that, the drone flies to the retrieve point x_R^2, it visits another 50% of the target with label 2 and then, from the launching point x_L^4 it starts its last operation to visit also another 50% of the target with label 1 ($u_L^{14} = u_R^{14} = 1$). Then, it flies to the last retrieve point x_R^4 and moves to the destination point together with the mothership. In the lower subfigure of Fig. 4, we can observe the solution obtained with the modified version of STEP 2. Different from the upper one, the number of drone operations is equal to 2. Indeed, thanks to the refinement of STEP 2, the drone endurance permits to visit the two targets with labels 4 and 3 in the first operation ($u_L^{41} = u_R^{31} = 1$ and $y^{431} = 1$). Similarly, the drone can also visit two targets, namely 2 and 1, in its second operation ($u_L^{22} = u_R^{12} = 1$ and $y^{212} = 1$).

In terms of objective function values, in this example, the refinement of STEP 2 does not provide an improved solution. Indeed, its value is equal to 888.01 without refinement and equal to 920.4 with the refinement of STEP 2. However, the length of the mothership tour, when STEP 2 is implemented in its original form, is equal to 189.72, while with the refinement of STEP 2 it is equal to 180.39. Moreover, we point out that the total time associated with the mothership tour is shorter in the solution without refinement of STEP 2. This is due to the different number of stops performed by the mothership in the two solutions. Indeed, in the solution obtained by the refinement, the number of mothership stops is 4 instead of 5 and, in some of them, the mothership waits for the drone. Summing up, in this example, the refinement of STEP 2 generates a solution with a shorter tour of the mothership but with a weighted sum of the distances traveled by both the drone and mothership, which is worse than the one obtained without refinement. In general, depending on the instance and the weighting factor of the two terms in the objective function, the refinement of STEP 2 can provide better solutions. For this reason, in the implementation we compared the solutions obtained with and without this refinement, and we select the best one to be provided as the initial solution for the exact model.

5. Experimental results

In this section, we discuss the experimental results obtained testing the formulation presented in Section 2.2 and the matheuristic procedure proposed in Section 4 on a testbed of instances.

In particular, we consider instances of three typologies: the first one in which the targets to be visited are represented by points randomly located in a square of side 100 units, the second one where the targets are represented by polygonals and the last one with heterogeneous targets, represented by points and polygonals.

More in details, to build polygonals, we set the cardinality of the set of points of the polygonal equal to 4 (this is for guaranteeing that the drone endurance is sufficient for traversing the whole polygonal) and we generate a random value r in the interval (5, 10). This value is used to generate a point $P = (P_x, P_y)$ inside the subsquare $[r, 100 - r]^2$.

Next, sequentially, we generate a random angle $\alpha \in (0, 2\pi)$ and from the current point P, we generate a new point $Q = (P_x + r * \cos(\alpha), P_y + r * \sin(\alpha))$. If point Q belongs to the square $[r, 100 - r]^2$, then we connect point Q to point P with a segment. Otherwise, we update α by adding $\pi/6$ to it until we obtain a point $Q' = (P_x + r * \cos(\alpha), P_y + r * \sin(\alpha))$ that belongs to the square $[r, 100 - r]^2$. Then, the same procedure is repeated to generate the remaining break points of the polygonal.

For each of the three typologies, we generate instances of increasing size (number of targets) ranging between 5 and 15. For each size, we consider the values of the drone endurance in the set {30, 40, 50, 60, 70}. For each combination of size and endurance value, we generate five instances.

We run the formulation on these instances by adopting the commercial solver Cplex, setting a time limit of two hours. Moreover, we run the matheuristic on the same sets of instances and we use the solutions obtained to initialize the exact solution method provided by the mathematical programming formulation. Specifically, in Table 4, for each size (column 1), for each endurance value (column 2) and for each typology of instances (Type 1 for points, Type 2 for polygonals, and Type 3 for points and polygonals), we report the average gaps. The first one is the average

Table 4
Average gaps

		Type 1				Type 2				Type 3			
		Gap (wi)	Gap (i)	Rel. Gap (LB)	Rel. Gap	Gap (wi)	Gap (i)	Rel. Gap (LB)	Rel. Gap	Gap (wi)	Gap (i)	Rel. Gap (LB)	Rel. Gap
5	30	0	0	0.2	0.2	0	0	0.14	0.14	0	0	0.14	0.14
	40	0	0	0.22	0.22	0	0	0.17	0.17	0	0	0.16	0.16
	50	0	0	0.23	0.23	0	0	0.17	0.17	0	0	0.17	0.17
	60	0	0	0.23	0.23	0	0	0.17	0.17	0	0	0.17	0.17
	70	0	0	0.23	0.23	0	0	0.17	0.17	0	0	0.17	0.17
6	30	0	0	0.16	0.16	0	0	0.19	0.19	0	0	0.12	0.12
	40	0	0	0.21	0.21	0	0	0.22	0.22	0	0	0.14	0.14
	50	0	0	0.21	0.21	0	0	0.24	0.24	0	0	0.15	0.15
	60	0	0	0.21	0.21	0	0	0.24	0.24	0	0	0.15	0.15
	70	0	0	0.21	0.21	0	0	0.24	0.24	0	0	0.15	0.15
7	30	0	0	0.2	0.2	0	0	0.19	0.19	0	0	0.15	0.15
	40	0	0	0.24	0.24	0.06	0.03	0.24	0.22	0	0	0.19	0.19
	50	0	0	0.25	0.25	0.05	0.15	0.34	0.23	0	0	0.2	0.2
	60	0	0	0.25	0.25	0	0.12	0.33	0.24	0	0	0.2	0.2
	70	0	0	0.25	0.25	0	0.06	0.29	0.24	0	0	0.2	0.2
8	30	0	0	0.23	0.23	0.42	0.5	0.59	0.17	0.12	0.04	0.16	0.13
	40	0	0	0.26	0.26	0.48	0.52	0.61	0.18	0.12	0.08	0.22	0.16
	50	0	0	0.27	0.27	0.51	0.5	0.6	0.19	0.29	0.17	0.3	0.16
	60	0	0	0.28	0.28	0.44	0.47	0.59	0.21	0.3	0.25	0.37	0.16
	70	0	0	0.28	0.28	0.49	0.49	0.6	0.21	0.28	0.2	0.34	0.17
9	30	0.36	0.27	0.43	0.22	0.68	0.65	0.69	0.11	0.62	0.54	0.59	0.1
	40	0.35	0.33	0.5	0.24	0.66	0.66	0.71	0.14	0.63	0.58	0.62	0.11
	50	0.37	0.29	0.48	0.27	0.67	0.66	0.71	0.12	0.62	0.54	0.61	0.13
	60	0.36	0.36	0.52	0.26	0.66	0.66	0.71	0.14	0.59	0.55	0.61	0.14
	70	0.33	0.31	0.5	0.27	0.64	0.63	0.69	0.15	0.58	0.55	0.61	0.13
10	30	0.5	0.45	0.54	0.17	0.69	0.66	0.69	0.08	0.72	0.7	0.71	0.06
	40	0.48	0.48	0.56	0.16	0.7	0.64	0.68	0.11	0.76	0.7	0.71	0.07
	50	0.63	0.44	0.55	0.2	0.7	0.68	0.7	0.08	0.74	0.73	0.75	0.07
	60	0.51	0.49	0.59	0.2	0.69	0.66	0.69	0.09	0.75	0.7	0.73	0.1
	70	0.47	0.46	0.57	0.21	0.71	0.67	0.69	0.08	0.74	0.72	0.74	0.08
11	30	0.61	0.54	0.62	0.17	0.71	0.67	0.68	0.01	0.81	0.78	0.78	0.01
	40	0.65	0.55	0.64	0.19	0.7	0.67	0.68	0.01	0.82	0.77	0.77	0.02
	50	0.65	0.54	0.64	0.22	0.69	0.68	0.68	0.01	0.79	0.75	0.76	0.08
	60	0.64	0.58	0.67	0.21	0.71	0.68	0.68	0.02	0.81	0.79	0.79	0.03
	70	0.6	0.56	0.65	0.2	0.72	0.68	0.69	0.01	0.78	0.77	0.78	0.03
12	30	0.76	0.66	0.71	0.14	0.72	0.68	0.69	0.05	0.81	0.77	0.77	0.01
	40	0.75	0.67	0.72	0.17	0.72	0.68	0.69	0.03	0.82	0.77	0.77	0
	50	0.72	0.74	0.78	0.18	0.72	0.69	0.7	0.03	0.81	0.78	0.78	0
	60	0.74	0.72	0.77	0.19	0.71	0.68	0.69	0.03	0.82	0.79	0.8	0.01
	70	0.71	0.71	0.77	0.23	0.71	0.68	0.69	0.04	0.81	0.78	0.78	0

Continued

Table 4
(Continued)

		Type 1				Type 2				Type 3			
		Gap (wi)	Gap (i)	Rel. Gap (LB)	Rel. Gap	Gap (wi)	Gap (i)	Rel. Gap (LB)	Rel. Gap	Gap (wi)	Gap (i)	Rel. Gap (LB)	Rel. Gap
	30	0.85	0.81	0.83	0.11	0.76	0.72	0.72	0	0.84	0.79	0.79	0
	40	0.84	0.8	0.83	0.16	0.75	0.72	0.73	0.02	0.84	0.81	0.81	0
13	50	0.82	0.83	0.86	0.16	0.75	0.72	0.72	0	0.84	0.81	0.81	0
	60	0.82	0.81	0.84	0.14	0.74	0.72	0.72	0	0.84	0.81	0.81	0
	70	0.83	0.82	0.86	0.18	0.75	0.72	0.72	0.02	0.84	0.81	0.81	0
	30	0.91	0.85	0.85	0.02	0.73	0.67	0.67	0.01	0.83	0.79	0.79	0
	40	0.9	0.84	0.85	0.07	0.72	0.67	0.67	0.01	0.83	0.79	0.79	0
14	50	0.89	0.87	0.87	0.03	0.72	0.67	0.67	0	0.83	0.79	0.79	0
	60	0.89	0.85	0.87	0.07	0.73	0.67	0.67	0	0.83	0.79	0.79	0
	70	0.91	0.84	0.86	0.1	0.71	0.67	0.67	0.01	0.84	0.79	0.79	0
	30	0.92	0.87	0.87	0	0.72	0.64	0.64	0	0.83	0.78	0.78	0
	40	0.9	0.87	0.88	0.01	0.7	0.64	0.64	0	0.83	0.78	0.78	0
15	50	0.91	0.88	0.88	0.02	0.72	0.63	0.63	0	0.83	0.78	0.78	0
	60	0.92	0.88	0.88	0.02	0.73	0.63	0.63	0	0.83	0.78	0.78	0
	70	0.93	0.88	0.88	0	0.71	0.64	0.64	0	0.83	0.78	0.78	0

gap, provided by the solver, computed as the relative gap between the best solution and the best lower bound found within the time limit (Gap (wi)), the second one is the average gap, provided by the solver, with initialization with the solution found by the matheuristic (Gap (i)), the third one is the average relative gap between the solution found by the matheuristic and the best lower bound provided by the solver (Rel.Gap (LB)), and the last one is the average relative gap between the solution found by the matheuristic and the best solution provided by the solver (Rel. Gap). From this table, we can see that Cplex is always able to find feasible solutions for all instances of all types and sizes. The average gap associated with the solutions provided by the solver, without initialization, ranges between 0 and 0.93, and it increases with the instance size.

Moreover, we can observe that Cplex is able to optimally solve instances of Type 1 up to size 8, instances of Type 2 up to size 6, and instances of Type 3 up to size 7, within the time limit. The values of the gaps associated with the exact solution method without initialization show that instances with polygonal targets up to size 9 appear to be more challenging than the ones with only point targets. The instances with mixed targets are in between and thus they represent an intermediate level of difficulty. However, when the size is between 10 and 12, the greatest values of the gap are associated with the instances consisting of mixed targets. Moreover, for size greater than or equal to 12, the average gap associated with instances of Type 1 is always greater than the one associated with instances of Type 2, while the instances of Type 3 have again values of the average gap in between the ones related to the other two typologies. Moreover, when the solution provided by the matheuristic algorithm is used to initialize the solver, in most of the cases the final average gap slightly decreases after the time limit is reached. The most significant reduction in the average gap can be observed for the largest sized instances of Type 2, for which the average gap with initialization is up to 13% lower than the one without initialization. As regards the relative

gap associated with the solution provided by the matheuristic with respect to the best lower bound found by the solver, we can observe that, for instances that are not solved at optimality by the solver, its values range between 0.43 and 0.88. For instances of Type 1, its values become much closer to the average gap associated with the exact resolution with initialization, when the size increases. For instances of Types 2 and 3, the values of both gaps are equal almost for all sizes. This shows that, in most of the cases, the solver is not able to improve the quality of the solution provided by the matheuristic. Moreover, for the biggest sized instances of all typologies we can see that the average relative gap of the solution provided by the matheuristic is lower than the one associated with the exact resolution without initialization. Thus, in these cases, the solution provided by the matheuristic, in few seconds or minutes, is even of better quality than the solution provided by the solver after two hours. Considering the average relative gap associated with the solution provided by the matheuristic with respect to the best solution found by the solver, we can see that it is always less than 0.3 and that it decreases with the size for all typologies of instances. In particular, it is equal to 0 or very close to 0, for instances of Types 2 and 3 with a number of targets greater than or equal to 12.

Table 5 reports for each typology of instance, for each size and for each endurance value, the average running times related to the exact solution with the solver (Time (wi)), the solution via the matheuristic (Time_h) and the exact solution via the solver with initialization provided by the matheuristic (Time (i)). In particular, we can observe that Cplex is able to solve, in average, in less than 6 seconds instances of size 5, in at most 15 minutes instances of size 6, and in at most 1 hour and a half instances of size 7. The average solution time increases by an order of magnitude with respect to the instance size, up to size 8 for Type 1, up to size 7 for Types 2 and 3. For larger sized instances, it reaches the time limit of two hours.

We can also observe that the matheuristic provides a solution of the problem in less than 1 second for instances with point targets (Type 1) up to size 9 and in less than 10 seconds for sizes between 10 and 15. The polygonal instances (Type 2) are more challenging to be solved with solution times of the matheuristic ranging between 1.42 seconds and 2.4 minutes. As for the instances of Type 3 (mixed targets), the solution time of the matheuristic is between a minimum of 1.82 seconds and a maximum of 4 minutes. However, comparing these times with those required by the exact solution of the formulation and considering the small values of the average relative gaps reported in Table 4, we can conclude that the matheuristic algorithm is a very good alternative to the exact solver.

As regards the computation times for the exact solution method initialized with the solution provided by the matheuristic, we can also observe that the convergence to the optimal solution for instances of Type 1 up to eight targets is significantly improved.

In Fig. 5, we show the boxplots representing the gap values for the different instance sizes, distinguishing between the three typologies. In particular, the left subfigure refers to the gap values obtained by solving the formulation without initialization, while the right subfigure, refers to the ones obtained by providing to the solver the initial solution generated by the matheuristic algorithm. Specifically, we can visualize that the gap values for instances with point targets increase with the problem size in the exact solution both without and with initialization. As regards the polygonal instances, we can observe a different behavior with respect to the gap, which increases with the instances size up to nine targets. For instances of size greater than or equal to 10, its gap, excluding outliers, always ranges in the interval [0.6, 0.8]. This can be still observed also by initializing the exact solution of the formulation even if, as already mentioned, in this latter case the

Table 5
Average running times (in seconds)

		Type 1			Type 2			Type 3		
		Time (wi)	Time_h	Time (i)	Time (wi)	Time_h	Time (i)	Time (wi)	Time_h	Time (i)
	30	5.33	0.29	4.04	13.41	1.42	16.83	4.86	1.9	4.51
	40	5.08	0.21	3.88	10.9	1.47	17.17	5.5	1.95	5.26
5	50	4.73	0.25	4.05	13.05	1.45	20.31	5.4	1.91	6.19
	60	5.27	0.32	4.72	19.07	1.42	20.75	5.96	2.05	6.38
	70	5.27	0.25	4.13	16.36	1.46	25.54	5.65	1.82	5.91
	30	21.95	0.2	18.66	148.69	2.41	286.12	49.68	2.17	51.97
	40	30.5	0.17	17.92	238.77	2.55	394.29	71.12	2.54	52.68
6	50	34.08	0.12	19.35	193.8	2.35	383.94	69.35	2.34	86.55
	60	33.39	0.16	19.72	185.92	2.24	322.92	85.89	2.55	82.53
	70	36.3	0.18	23.35	264.39	2.69	413.62	69.9	2.41	61.44
	30	229.75	0.21	201.92	3063.24	4.25	2806.77	709.57	4.22	621.96
	40	268.47	0.46	204.39	3420.54	4.07	4651.65	1127.66	3.81	1139.36
7	50	255.35	0.35	213.41	4034.04	3.87	5409.26	901.86	4.3	960.31
	60	348.62	0.46	208.24	4787.98	4.58	6281.02	986.83	4.61	940.26
	70	301.47	0.39	243.37	3986.24	4.48	6103.81	1053.51	4.16	790.63
	30	2265.56	0.7	1619.37	7200	7.61	7200	5549.54	5.3	4800.82
	40	2820.11	0.47	2333.97	7200	7.65	7200	6876.95	5.61	6487.69
8	50	2877.69	0.48	2020.92	7200	7.45	7200	7200	5.67	6769.41
	60	3159.62	0.63	2484.49	7200	7.02	7200	7200	5.68	7176.09
	70	3188.16	0.62	2724.14	7200	7.8	7200	7200	5.46	7123.6
	30	7200	0.94	7200	7200	13.64	7200	7200	12.08	7200
	40	7200	0.91	7200	7200	12.31	7200	7200	12.28	7200
9	50	7200	0.9	7200	7200	14.88	7200	7200	10.23	7200
	60	7200	0.72	7200	7200	14.63	7200	7200	12.53	7200
	70	7200	0.68	7200	7200	13.37	7200	7200	10.18	7200
	30	7200	1.43	7200	7200	22.58	7200	7200	18.01	7200
	40	7200	1.68	7200	7200	20.89	7200	7200	16.95	7200
10	50	7200	1.61	7200	7200	21.31	7200	7200	16.4	7200
	60	7200	1.74	7200	7200	22.1	7200	7200	17.25	7200
	70	7200	1.8	7200	7200	23.62	7200	7200	20.59	7200
	30	7200	1.8	7200	7200	34.72	7200	7200	31.71	7200
	40	7200	2.1	7200	7200	34.9	7200	7200	26.86	7200
11	50	7200	2.12	7200	7200	34.31	7200	7200	24.99	7200
	60	7200	2	7200	7200	34.9	7200	7200	27.88	7200
	70	7200	2.11	7200	7200	35.02	7200	7200	27.19	7200
	30	7200	3.14	7200	7200	51.52	7200	7200	34.51	7200
	40	7200	3.43	7200	7200	53.85	7200	7200	52.46	7200
12	50	7200	3.04	7200	7200	53.02	7200	7200	32.36	7200
	60	7200	2.97	7200	7200	53.83	7200	7200	41.66	7200
	70	7200	3.12	7200	7200	58.29	7200	7200	27.77	7200
	30	7200	5.42	7200	7200	78.7	7200	7200	120.74	7200
	40	7200	4.3	7200	7200	83.12	7200	7200	138.35	7200
13	50	7200	5.46	7200	7200	79.63	7200	7200	119.08	7200
	60	7200	4.16	7200	7200	86.79	7200	7200	151.91	7200
	70	7200	4.28	7200	7200	83.36	7200	7200	120.16	7200

Continued

Table 5
(Continued)

		Type 1			Type 2			Type 3		
		Time (wi)	Time_h	Time (i)	Time (wi)	Time_h	Time (i)	Time (wi)	Time_h	Time (i)
	30	7200	6.75	7200	7200	107.31	7200	7200	143.91	7200
	40	7200	5.88	7200	7200	110.38	7200	7200	163.42	7200
14	50	7200	6.07	7200	7200	107.14	7200	7200	132.36	7200
	60	7200	5.71	7200	7200	112.05	7200	7200	111.87	7200
	70	7200	6.4	7200	7200	111.62	7200	7200	99.63	7200
	30	7200	7.1	7200	7200	141.05	7200	7200	185.87	7200
	40	7200	7	7200	7200	146.34	7200	7200	237.79	7200
15	50	7200	9.47	7200	7200	132.54	7200	7200	130.69	7200
	60	7200	6.31	7200	7200	144.69	7200	7200	158.09	7200
	70	7200	7.15	7200	7200	140.05	7200	7200	143.95	7200

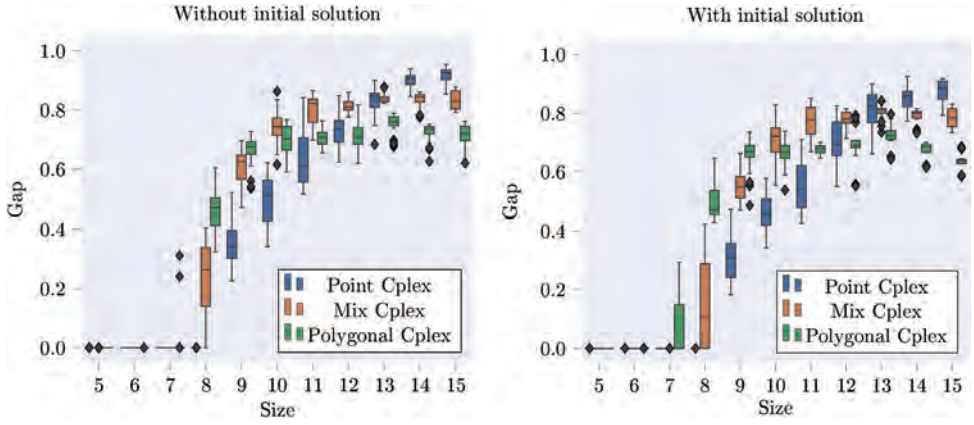


Fig. 5. Final gap with and without initialization by using Cplex.

gap values decrease and in most of the cases they belong to the interval [0.6,0.7], always excluding outliers. Thus, the polygonal instances are more difficult than the point instances for sizes up to 10. For bigger sized instances, the gap value for polygonal instances consistently belongs to a fixed interval, while the gap for point instances increases with the size. As regards the mixed instances, we can see that the behavior of the gap is similar to the one of the instances of Type 2. It increases with the size up to 11 targets. Then, always excluding outliers, it belongs to the interval [0.8, 0.9], without initialization, and to the interval [0.7, 0.8] with initialization.

To further investigate this behavior, we run the mathematical programming formulation with Cplex on one of the biggest sized instances with 15 targets and endurance equal to 40, for both Types 1 and 2 and setting a time limit of 24 hours. Figure 6 reports the objective function value and the lower bound value obtained over time, respectively, for the point and polygonal target versions of this instance. We can observe that the initial value of the lower bound for the point targets instance is equal to 0 while the one for the polygonal targets instance is equal to 331. This difference is due to the different structure of the targets. Indeed, for the polygonal targets, the formulation

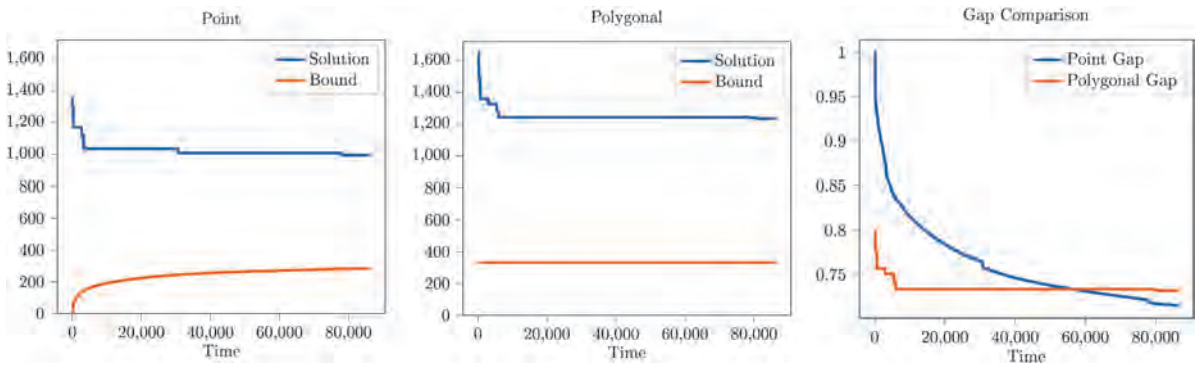


Fig. 6. Evolution of objective function and lower bound values of one instance with 15 pointwise targets and drone endurance equal to 40 in 24 hours of CPU time.

includes the constraint imposing the visit of a given minimum fraction of each polygonal. This implies that the first value of the lower bound is greater than 0. Thus, while the solution process for the point targets instance starts from a gap equal to 1, the one for the polygonal targets instance starts from a gap equal to 0.80, as we can also observe in Fig. 6. This different initial gap allows the solver to slowly improve it in the first case but not in the second case. Actually, only after around 15 hours of CPU, we can observe that the two gaps (for the instance of Type 1 and 2) are equal and then the value of the gap for the point targets instance continues to decrease, whereas the one related to the polygonal targets instance does not change in the remaining time. This is explained because the lower bound of the point targets instance improves over time, see Fig. 6, but it does not change at all for the polygonal targets instance.

This behavior shows that the instances of Type 2 (polygonal targets) are still more difficult to be solved than the instances of Type 1 (point targets). However, the different structure of the targets provides an initial advantage in the lower bound of the polygonal target instances, due to the constraint related to the minimum fraction of each polygonal to be visited, which can be exploited by the solver for the biggest sizes. For the same reason, also the instances of Type 3, consisting of mixed targets, have an initial advantage in the lower bound that explains lower values of the gap after two hours, with respect to the one related to the instances of Type 1, when the number of targets is greater than 12. Indeed, among the mixed targets, the polygonal ones imply the presence of the constraint imposing the visit of a given minimum fraction of each of them. This makes the formulation more restrictive and the solver is able to find an initial lower bound greater than 0, from which it takes an advantage, with respect to instances of Type 1.

6. Concluding remarks

This paper has analyzed a coordination problem that arises between a mothership vehicle and a drone that is allowed to visit more than one target per operation, synchronizing their displacements to minimize the travel distance. The mothership can move freely in a continuous space so that the drone can be launched and recovered at any point that is convenient to optimize the problem

goal. We present an exact model that can be adapted to deal with pointwise and graph-like targets. This model is a mixed integer second-order cone problem and it can be solved by most of the current nowadays solvers. Our computational results show that the considered problem is rather hard and only small- to medium-sized problems can be solved to optimality. For that reason, we also developed a matheuristic algorithm as a fast alternative to exact methods, which provides acceptable feasible solutions in short computing time.

An interesting problem related to the one in this paper is the coordination of the operations of one or several motherships with several drones, each one of them allowed to visit more than one target per operation. This is a realistic, challenging problem that models actual situations in drone's delivery situations, but although it is very interesting and deserves to be studied, it is beyond the scope of this paper and will be the topic of a follow-up paper.

Acknowledgments

This research has been partially supported by the Agencia Estatal de Investigación (AEI) and the European Regional Development's fund (ERDF): PID2020-114594GB-C21; Regional Government of Andalusia: projects CEI-3-FQM331, FEDER-US-1256951, and P18-FR-1422; Fundación BBVA: project NetmeetData (Ayudas Fundación BBVA a equipos de investigación científica 2019) and University of Rome, Sapienza grant number RM11916B7F962975.

References

AltiGator, 2015. A drone to rescue immigrants. Available at https://altigator.com/en/rescue-immigrants-with-a-drone/.

Amorosi, L., Chiaraviglio, L., Galan-Jimenez, J., 2019. Optimal energy management of UAV-based cellular networks powered by solar panels and batteries: formulation and solutions. *IEEE Access* 7, 53698–53717.

Amorosi, L., Puerto, J., Valverde, C., 2021a. Coordinating drones with mothership vehicles: the mothership and drone routing problem with graphs. *Computers and Operations Research* 136. https://doi.org/10.1016/j.cor.2021.105445.

Amorosi, L., Puerto, J., Valverde, C., 2021b. Coordinating drones with mothership vehicles: the mothership and multiple drones routing problem with graphs. https://doi.org/10.48550/arXiv.2109.01447.

Blanco, V., Fernández, E., Puerto, J., 2017. Minimum spanning trees with neighborhoods: mathematical programming formulations and solution methods. *European Journal of Operational Research* 262, 3, 863–878.

Cavani, S., Iori, M., Roberti, R., 2021. Exact methods for the traveling salesman problem with multiple drones. *Transportation Research Part C: Emerging Technologies* 130. https://doi.org/10.1016/j.trc.2021.103280.

Chiaraviglio, L., Amorosi, L., Blefari-Melazzi, N., Dell'olmo, P., Lo Mastro, A., Natalino, C., Monti, P., 2019. Minimum cost design of cellular networks in rural areas with UAVs, optical rings, solar panels, and batteries. *IEEE Transactions on Green Communications and Networking* 3, 4, 901–918.

Chui, M., Manyika, J., Miremadi, M., 2016. Where machines could replace humans—and where they can't (yet). *McKinsey Quarterly* 7, 1–6.

Chung, S.H., Sah, B., Lee, J., 2020. Optimization for drone and drone-truck combined operations: a review of the state of the art and future directions. *Computers & Operations Research* 123, 105004.

Coindreau, M., Gallay, O., Zufferey, N., 2021. Parcel delivery cost minimization with time window constraints using trucks and drones. *Networks*. https://doi.org/10.1002/net.22019.

Dell'Amico, M., Montemanni, R., Novellani, S., 2022. Exact models for the flying sidekick traveling salesman problem. *International Transactions in Operational Research* 29, 1360–1393.

Di Puglia Pugliese, L., Guerriero, F., 2017. *Last-Mile Deliveries by Using Drones and Classical Vehicles BT—Optimization and Decision Science: Methodologies and Applications*. Springer International Publishing, Cham, pp. 557–565.

Macrina, G., Di Puglia Pugliese, L., Guerriero, F., Laporte, G., 2020. Drone-aided routing: a literature review. *Transportation Research Part C: Emerging Technologies* 120. https://doi.org/10.1016/j.trc.2020.102762.

Mbiadou Saleu, R.G., Deroussi, L., Feillet, D., Grangeon, N., Quilliot, A., 2021. The parallel drone scheduling problem with multiple drones and vehicles. *European Journal of Operational Research* 300, 571–589.

Otto, A., Agatz, N., Campbell, J., Golden, B., Pesch, E., 2018. Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: a survey. *Networks* 72, 1–48.

Puerto, J., Valverde, C., 2021. Routing for unmanned aerial vehicles: touring dimensional sets. *European Journal of Operational Research* 298, 118–136.

Roberti, R., Ruthmair, M., 2021. Exact methods for the traveling salesman problem with drone. *Transportation Science* 55, 315–335.

Rojas Viloria, D., Solano-Charris, E.L., Muñoz-Villamizar, A., Montoya-Torres, J.R., 2021. Unmanned aerial vehicles/drones in vehicle routing problems: a literature review. *International Transactions in Operational Research* 28, 4, 1626–1657.

Vidal, T., Laporte, G., Matl, P., 2020. A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research* 286, 2, 401–416.

Wang, K., Pesch, E., Kress, D., Fridman, I., Boysen, N., 2022. The piggyback transportation problem: transporting drones launched from a flying warehouse. *European Journal of Operational Research* 296, 2, 504–519.

Appendix

The special case of polygonal chains with equal-length segments

This section shows a simplification of the constraints modeling traversing a given fraction of the polygonal targets provided that all their segments are of the same length. In that case, we can model the arrival point A^p and the departure point D^p associated with the polygonal chain $p \in \mathcal{P}$ by means of a parameter $\rho^p \in [1, |V_p|]$ (resp. $\lambda^p \in [1, |V_p|]$) that determines the absolute position in p.

To compute the value of these parameters, we can link them to the variables that model the location of the points inside the polygonal by including the following inequalities for each $s_p \in S_p$:

$$\rho^p - s_p \geq \gamma_A^{s_p} - |V_p|(1 - \mu_A^{s_p}),$$
$$\rho^p - s_p \leq \gamma_A^{s_p} + |V_p|(1 - \mu_A^{s_p}),$$
$$\lambda^p - s_p \geq \gamma_D^{s_p} - |V_p|(1 - \mu_D^{s_p}),$$
$$\lambda^p - s_p \leq \gamma_D^{s_p} + |V_p|(1 - \mu_D^{s_p}).$$

The first and second inequalities determine the upper and lower limits for the parameterization of each segment of p. If $\mu_A^{s_p} = 0$ the inequalities are always fulfilled and there is no arrival point in the s_pth segment of the polygonal. Conversely, if $\mu_A^{s_p} = 1$ then $\rho^p = \gamma_A^{s_p} + s_p$ meaning that the corresponding arrival or departure point is in the s_pth segment of the polygonal and its value is equals to the number of segments plus the part of the segment s_p that has been already traversed. The same idea is applied in the third and fourth inequalities for the λ^p parameter.

Finally, we can model the condition of traversing a fraction α^p of the polygonal p by the standard trick of the absolute value constraint:

$$|\rho^p - \lambda^p| \geq \alpha^p |S_p| \Longleftrightarrow \begin{cases} \rho^p - \lambda^p = \nu^p_{\max} - \nu^p_{\min} \\ \nu^p_{\max} \leq 1 - \text{entry}^p, \\ \nu^p_{\min} \leq \text{entry}^p, \\ \nu^p_{\max} + \nu^p_{\min} \geq \alpha^p |S_p|. \end{cases} \qquad (\alpha - \mathcal{P})$$

Here $\nu^p_{\max}$ and $\nu^p_{\min}$ are auxiliary variables used for linearizing the absolute value and the binary variable entry^p represents the traveling direction in the polygonal chain p.

Chapter 11

A multiple-drone arc routing and mothership coordination problem

Computers & Operations Research 159 (2023) 106322

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

A multiple-drone arc routing and mothership coordination problem

Lavinia Amorosi [a,*,1], Justo Puerto [b,1], Carlos Valverde [b,1]

[a] Department of Statistical Sciences, Sapienza University of Rome, Italy
[b] Department of Statistical Sciences and Operational Research, University of Seville, Spain

ARTICLE INFO

Keywords:
Arc routing problems
Networks
Drones
Conic programming

ABSTRACT

This paper considers the optimisation problems that arise in coordinating a tandem between a mothership vehicle and a fleet of drones. Each drone can be launched from the mothership to perform a task. After completing their tasks, the drones return to the mothership to recharge their batteries and be ready for a new task. Tasks consist of (partially) visiting graphs of a given length to provide some services or to carry out a surveillance/inspection activity. The goal is to minimise the overall time of travelling carried out by the mothership (makespan) while satisfying some requirements in terms of fractions of visits to the target graphs. In all cases, we develop exact formulations resorting to mixed-integer second-order cone programmes that are compared on a testbed of instances to assess their performance. We also develop a matheuristic algorithm that provides reasonable solutions. Computational experiments show the usefulness of our methodology in different scenarios.

1. Introduction

In recent years, the growth of potential business opportunities related to the use of drone technology has motivated the appearance of an interesting body of methodological literature on optimising the use of this technology. Examples may be found in many different sectors, such as telecommunications, where drones can be adopted in place of traditional infrastructures to provide connectivity (see, for example, Amorosi et al. (2018), Chiaraviglio et al. (2018), Jiménez et al. (2018), Amorosi et al. (2019), and Chiaraviglio et al. (2019a)), or to temporarily deal with damage caused by a disaster (Chiaraviglio et al., 2019b; Dönmez et al., 2021), deliveries (see, for example, Mathew et al. (2015), Ferrandez et al. (2016), Poikonen and Golden (2020b), Amorosi et al. (2020), and Pei et al. (2021)), also in emergency (Wen et al., 2016), inspection (Trotta et al., 2018) and other contexts. The reader is referred to the recent surveys (Otto et al., 2018; Chung et al., 2020; Dönmez et al., 2021) for further details. In this context, depending on the specific application, we can distinguish three main different systems: one or multiple drones that, starting from a depot, provide a given service, one or multiple drones supported by a traditional vehicle that works only as a mobile depot (and/or recharging station), and one or multiple drones that cooperate with one or more traditional vehicles. In the latter case, the vehicles are also responsible for providing the service.

In the rest of this section, we review the related literature, limiting ourselves to articles that focus on truck-and-drone systems. After the seminal paper (Murray and Chu, 2015) that introduces the *Flying Sidekick Travelling Salesman Problem* (FSTSP), in which a truck and a drone cooperate to make deliveries, Ulmer and Thomas (2018) considers another model in which a fleet of trucks and drones is dispatched when orders are placed and analyses the effect of different policies to decide whether an order must be delivered by a drone or by a vehicle. Other articles, such as Campbell et al. (2017) and Carlsson and Song (2018), also study hybrid truck-and-drone models to mitigate the limited delivery range of drones. Dayarian et al. (2020) focuses on a delivery system in which traditional vehicles are resupplied by drones. Dell'Amico et al. (2021) presents a branch-and-bound algorithm and heuristics based on it to solve large-sized instances of the FSTSP. In Poikonen and Golden (2020b), the authors study the *k-Multi-Visit Drone Routing Problem* (k-MVDRP) in which a truck acts as a mobile depot. The truck is allowed to stop at a predefined set of points and launches drones that can deliver more than one package to their designated destination points (customers). The model also includes a drone energy drain function that takes into account each package weight.

Many of the articles cited assume that the set of allowable locations to launch/retrieve a drone is fixed and known a priori, the operations carried out by the drone consist of delivering to a single point, and coordination is between a truck and a single drone. These assumptions may be appropriate in some contexts, but in other cases it would be better to relax them. In Poikonen and Golden (2020a) the authors progress on the coordination problem by introducing the *Mothership*

* Corresponding author.
E-mail addresses: lavinia.amorosi@uniroma1.it (L. Amorosi), puerto@us.es (J. Puerto), cvalverde@us.es (C. Valverde).
[1] Equally contributing authors.

https://doi.org/10.1016/j.cor.2023.106322
Received 10 August 2022; Received in revised form 6 April 2023; Accepted 18 June 2023
Available online 1 July 2023

and Drone Routing Problem (MDRP) in which a two-vehicle tandem is used to design a route that visits a set of points that allow the mothership (which may be a ship or an aircraft) to launch and recover the drone in a continuous space. However, only a few articles in the literature focus on drone operations that consist of traversing graphs rather than visiting single points. In Campbell et al. (2018) the authors introduce the *Drone Rural Postman Problem* (DRPP). The paper presents a solution algorithm based on the approximation of curves in the plane by polygonal chains that iteratively increases the number of points in the polygonal chain where the UAV can enter or leave. Thus, the problem is solved as a discrete optimisation problem trying to better define the curve by increasing the number of points. The authors also consider the case in which the drone has limited endurance and thus cannot serve all lines. To deal with the latter case, they assume to have a fleet of drones and the problem consists of finding a set of routes, each of limited length. In Campbell et al. (2021) this problem is defined as the *Length Constrained K-Drones Rural Postman Problem* (LC K-DRPP), a continuous optimisation problem in which a fleet of homogeneous drones has to jointly service (traverse) a set of (curved or straight) lines of a network. The authors design and implement a branch-and-cut algorithm for its solution and a matheuristic algorithm capable of providing good solutions for large-scale instances of the problem.

Scanning the literature on arc routing problems involving hybrid systems consisting of one vehicle and one or multiple drones, the number of contributions is rather limited. In Tokekar et al. (2016) the authors study the path planning problem of a system composed of a ground robot and a drone in precision agriculture and solve it by applying orienteering algorithms. Furthermore, the paper Garone et al. (2010) studies the problem of path planning for systems consisting of a carrier vehicle and a carried one to visit a set of target points and assumes that the carrier vehicle moves in continuous space.

To the best of our knowledge, the papers Amorosi et al. (2021, 2022) are the only that deal with the coordination of a mothership with a drone to visit targets represented by graphs. In particular, in Amorosi et al. (2021) the authors make different assumptions on the route followed by the mothership: (i) it can move on the Euclidean plane, (ii) on a connected piecewise linear polygonal chain, or (iii) on a general graph. In all cases, the authors develop exact formulations using mixed-integer second-order cone programmes and propose a matheuristic algorithm capable of obtaining high-quality solutions in short computing time. The set of target graphs to be visited permits one to model real situations like monitoring or inspection activities on fractions of networks (roads or wires) where traditional vehicles cannot arrive, due to, for example, the presence of narrow streets, or because of a natural disaster or a terrorist attack that causes damage to the network. In all these cases, drone inspection or monitoring consists of traversing the edges of the network to perform a reconnaissance activity. For this reason, the targets that the drone will visit are modelled as graphs. Note that, in general, the shape of the graph edges may be a straight line or a curve. In this paper, we limit the discussion to the case of straight lines. However, as shown in Campbell et al. (2018), it is possible to deal with curved edges by approximating them through polygonal chains. Thus, this paper represents a first building block in the study of coordination models between a mothership and a fleet of drones that embed drone arc routing problems. The investigation on different edges shape is out of the scope of this paper and it is left for future research. The action of visiting a graph can be of two different types: (i) traversing a given fraction of the length of each one of its edges or (ii) visiting a fraction of the total length of the network. Other types of inspection activities, such as, for example, video surveillance of urban areas in large cities, can also be modelled by adopting the formulations presented in this paper. In this context, the request to visit a certain fraction of the target graphs (e.g., borders of a neighbourhood) may be due to the necessity of "covering" different areas in a limited time interval. Another example that we can mention is traffic flow monitoring. In this case, to verify whether traffic progression is not disrupted, only inspecting a fraction of the edge provides valuable information.

In this article, we deal with an extension of the problem studied in Amorosi et al. (2021), for which we propose a novel mothership-and-multi-drone coordination model. We consider a system in which a base vehicle (mothership) travels in continuous space and must support the launch/retrieval of several drones that must visit graphs. Indeed, depending on the specific application, the tandem system may require the adoption of multiple drones to perform surveillance/monitoring activities, for example, to accelerate data collection in emergency contexts. The presence of several drones opens up different possible working principles of the tandem system that can be more or less appropriate depending on the specific application. In particular, we focus on two different versions.

The contributions of this article to the existing literature can be summarised as follows:

(i) it extends the mothership-drone coordination problem to the more cumbersome case of several drones;
(ii) it focuses on drone arc routing problems in which drone operations consist of traversing graphs rather than visiting single points;
(iii) it studies two versions of the problem that make the presented mathematical programming approach flexible with respect to different working principles of the tandem system;
(iv) it presents matheuristic algorithms able to handle large-sized instances;
(v) it includes in the Appendix the model extension for dealing also with a non-homogeneous fleet of drones.

The rest of the paper is structured as follows. Section 2 provides a detailed description of the problem under consideration. Section 3 develops valid Mixed-Integer Non-Linear Programming (MINLP) formulations for the two versions of the problem considered. Here, we also show the relationship between the two models and prove that the second model is a relaxation of the first. Section 4 provides some valid inequalities that strengthen the formulations and derive upper and lower bounds on the bigM constants introduced in the proposed formulations. Section 5 presents details of the matheuristic algorithms designed to handle large-sized instances. In Section 6, we report the results obtained by testing the formulations and the matheuristic algorithm on different classes of planar graphs to assess their effectiveness. In Section 7, we present an illustrative example that applies the coordination models for the Cordoba Courtyard Festival. Section 8 concludes the paper. Finally, for the sake of completeness, we include an Appendix with an extension of the model of complete overlapping where the drones are not assumed to be homogeneous.

2. Problem description

In the *All Terrain Mothership and Multiple-Drone Routing Problem with Graphs* (AMMDRPG), there is one mothership (the base vehicle) and a fleet of homogeneous drones D that have to coordinate between each other and with the mothership to perform a number of operations consisting of visiting given fractions of the length of a set of graphs $\mathcal{G}$. The mothership and the drones travel at constant speeds v_M and v_D, respectively. We also assume that it is not necessary for the mothership to be stopped to launch and retrieve the drones and that the time spent by the mothership to launch and retrieve them is negligible. Furthermore, it is assumed that each drone has a limited endurance N_D, so once launched, it must complete the operation and return to the base vehicle to recharge its batteries before the time limit. In addition, the base vehicle freely moves in the continuous space. This assumption can model the case where the base vehicle is a helicopter or a boat, so that there are no obstacles or restrictions on its movement. Nowadays, this type of system consisting of a boat and a fleet of drones is used, for example, by coast guards to carry out surveillance activities to identify

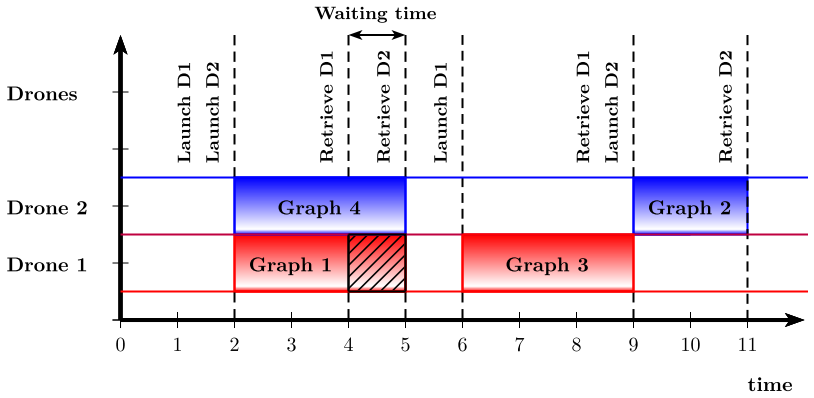


Fig. 1. Model with complete overlapping.

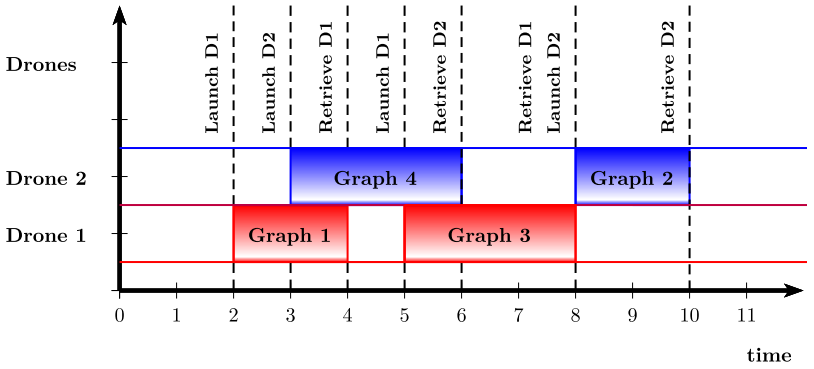


Fig. 2. Model with partial overlapping.

immigrants who need help on the sea (see AltiGator (2015)). The mothership starts at a known location, denoted *orig* where the entire system is ready to depart. Once all operations are completed, the mothership and drones must return together to a final location called *dest*. Moreover, without loss of generality, we assume that the endurance of the drone does not allow it to visit all target graphs in a single trip, starting from the origin and ending at the destination. Otherwise, the problem becomes trivial, and coordination is not required.

In this problem, it is assumed that each graph must be visited by one drone: once the drone is assigned to this action, it visits the graph and has to complete the entire action of traversing this target before returning to the base. We assume that the time each drone spends visiting the graph must be less than or equal to the time the mothership takes to move from the launching point to the retrieval point. Note also that each drone in the fleet cannot be launched from the same base vehicle location to carry out all tasks due to its limited endurance. Additionally, the costs induced by the drone trips are negligible compared to those incurred by the base vehicle. Therefore, the goal is to minimise the makespan, which in this case coincides with the overall time travelled by the mothership. Despite that, the reader may note that from a theoretical point of view, the extension to include in the objective function the times travelled by drones is straightforward and does not increase the complexity of the models or formulations.

The goal of the AMMDRPG is to find the launch and retrieval points of the drone fleet D that meet the visit requirements of the graphs in $\mathcal{G}$ and minimise the makespan (total time travelled by the mothership).

In this paper, we focus on two different versions of AMMDRPG. In the first, called AMMDRPG with *complete overlapping model* (AMMDRPG-CO), operations consisting of the launch and retrieval of a set of drones are carried out sequentially so that no two consecutive launches are possible without the retrieval of previously launched drones (see Fig. 1). The second version, called AMMDRPG with *partial overlapping model* (AMMDRPG-PO), allows consecutive launch or retrieval actions so that the visits of several drones to their target graphs are allowed to partially overlap over time (see Fig. 2). This second version is more difficult to model. Indeed, the possibility for the mothership to retrieve one drone in a different phase from that in which it has been launched implies the introduction of additional concepts and decision variables to properly formulate the problem. Moreover, additional constraints must be included to model the mothership route and its relationship with one of the drones. For both variants of the problem, we present mathematical programming formulations, valid inequalities to strengthen them, and ad hoc matheuristics to deal with medium-sized instances of the problem.

Note that the partial overlapping variant is an extension of the complete overlapping case. In fact, we allow one drone to be launched and retrieved before another different drone is launched to visit another target graph. Therefore, when applicable, the partial overlapping version can provide smaller values of the makespan. This can be observed in Figs. 1 and 2, showing the Gantt diagram of a feasible solution for the two versions of the problem, involving two drones and four target graphs. In particular, in Fig. 1, the visit of Graph 3 starts at 6 and the makespan is equal to 11. This fact is due to the constraint that the mothership must wait for the retrieval of both drones before launching Drone 1 again. In Fig. 2, the possibility to launch both drones in an asynchronous way allows us to move the visit of Graph 3 up, with a makespan equal to 10.

3. Mixed-integer non-linear programming formulations

In this section, we present a MINLP formulation for the AMMDRPG that can be used to solve medium-sized instances of this problem. As mentioned in Section 2, we assume that the mothership is allowed to move freely in a continuous space that, for the sake of presentation, we assume to be $\mathbb{R}^2$. Here, distances are measured by the Euclidean

Table 1
Nomenclature for AMMDRPG.

Problem parameters
$orig$: coordinates of the point defining the origin of the mothership path (or tour).
$dest$: coordinates of the point defining the destination of the mothership path (or tour).
$\mathcal{G}$: set of the target graphs.
$g = (V_g, E_g)$: set of nodes and edges of each target graph $g \in \mathcal{G}$.
$\mathcal{L}(e_g)$: length of edge e of graph $g \in \mathcal{G}$.
$\mathcal{L}(g) = \sum_{e_g \in E_g} \mathcal{L}(e_g)$: total length of the graph $g \in \mathcal{G}$.
B^{e_g}, C^{e_g}: coordinates of the endpoints of edge e of graph $g \in \mathcal{G}$.
α^{e_g}: fraction of length of edge e of graph $g \in \mathcal{G}$ that must be visited. It ranges from 0 to 1.
α^g: fraction of length of graph $g \in \mathcal{G}$ that must be visited. It ranges from 0 to 1.
v_M: mothership speed.
$\lvert D\rvert$: number of drones.
v_D: drone speed.
N_D: drone endurance.
$\mathcal{O}$: set of drone operations to perform visits to the target graphs. $\mathcal{O} = \{1, \dots, \lvert\mathcal{O}\rvert\}$.
M: big-M constant.

Table 2
Decision variables for AMMDRPG-CO.

Binary decision variables
$\mu^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G})$: equal to 1 if edge e of graph g (or a fraction of it) is visited by the drone, 0 otherwise.
$\text{entry}^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G})$: auxiliary binary variable used for linearising expressions.
$u^{e_g o} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall o \in \mathcal{O}$: equal to 1 if one drone enters graph g through the edge e_g at operation o, 0 otherwise.
$z^{e_g e'_g} \in \{0,1\},\ \forall e_g, e'_g \in E_g\ (g \in \mathcal{G})$: equal to 1 if one drone goes from e_g to e'_g, 0 otherwise.
$v^{e_g o} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall o \in \mathcal{O}$: equal to 1 if one drone exits graph g by e_g at operation o, 0 otherwise.
Continuous decision variables
$s^{e_g} \in [0, \lvert E_g\rvert - 1],\ \forall e_g \in E_g\ (g \in \mathcal{G})$: continuous non-negative variable representing the order of visits to the edge e of graph g.
$\rho^{e_g} \in [0,1]$ and $\lambda^{e_g} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathcal{G})$: defining the entry and exit points on e_g.
$v^{e_g}_{\min}$ and $v^{e_g}_{\max} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathcal{G})$: auxiliary variables used for linearising expressions.
$x^o_L \in \mathbb{R}^2,\ \forall o \in \mathcal{O}$: coordinates representing the point where the mothership launches the drones at operation o.
$x^o_R \in \mathbb{R}^2,\ \forall o \in \mathcal{O}$: coordinates representing the point where the mothership retrieves the drones at operation o.
$R^{e_g} \in \mathbb{R}^2,\ \forall e_g \in E_g\ (g \in \mathcal{G})$: coordinates representing the entry point on edge e_g of graph g.
$L^{e_g} \in \mathbb{R}^2,\ \forall e_g \in E_g\ (g \in \mathcal{G})$: coordinates representing the exit point on edge e_g of graph g.
$d_{orig} \geq 0$: that represents distance from the origin $orig$ to the first launching point x^1_L.
$d^{e_g o}_L \geq 0,\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall o \in \mathcal{O}$: representing the distance travelled by one drone from the launching point x^o_L on the mothership at operation o to the first visiting point R^{e_g} on e_g.
$p^{e_g o}_L \geq 0,\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall o \in \mathcal{O}$: auxiliary variable used for modelling the product of $d^{e_g o}_L$ and $u^{e_g o}$.
$d^{e_g} \geq 0,\ \forall e_g \in E_g\ (g \in \mathcal{G})$: representing the distance travelled by the drone from the retrieval point R^{e_g} to the launching point L^{e_g} on e_g.
$p^{e_g} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathcal{G})$: auxiliary variable used for modelling the product of μ^{e_g} and $\lvert\lambda^{e_g} - \rho^{e_g}\rvert$.
$d^{e_g e'_g} \geq 0,\ \forall e_g, e'_g \in E_g\ (g \in \mathcal{G})$: representing the distance travelled by the drone from the launching point L^{e_g} on e_g to the retrieval point $R^{e'_g}$ on e'_g.
$p^{e_g e'_g} \geq 0,\ \forall e_g, e'_g \in E_g\ (g \in \mathcal{G})$: auxiliary variable used for modelling the product of $d^{e_g e'_g}$ and $z^{e_g e'_g}$.
$d^{e_g o}_R \geq 0,\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall o \in \mathcal{O}$: representing the distance travelled by one drone from the last visiting point L^{e_g} on e_g to the retrieval point x^o_R on the mothership at operation o.
$p^{e_g o}_R \geq 0,\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall o \in \mathcal{O}$: auxiliary variable used for modelling the product of $d^{e_g o}_R$ and $v^{e_g o}$.
$d^o_{LR} \geq 0,\ \forall o \in \mathcal{O}$: representing the distance travelled by the mothership from the launching point x^o_L to the retrieval point x^o_R at operation o.
$d^o_{RL} \geq 0,\ \forall o \in \mathcal{O} \setminus \lvert\mathcal{O}\rvert$: representing the distance travelled by the mothership from the retrieval point x^o_R at operation o to the launching point $x^{(o+1)}_L$ at operation $o+1$.
$d_{dest} \geq 0$: that represents distance from the last retrieval point $x^{\lvert\mathcal{O}\rvert}_R$ to the destination $dest$.
$time^o_D \geq 0,\ \forall o \in \mathcal{O}$: maximum time spent by a drone during operation o.
$time^o_M \geq 0,\ \forall o \in \mathcal{O}$: time spent by the mothership to go from the launching point x^o_L to the retrieval point x^o_R of operation o.
$time_M \geq 0$: total time spent by the mothership to go from the origin to the destination (makespan).

norm, $\|\cdot\|_2$, although this assumption can be extended to any l_p norm, $1 \leq p \leq \infty$ (see Blanco et al. (2017)).

Table 1 summarises the parameters or input data that formally describe the problem.

In the following, we describe all the constraints required to formulate the AMMDRPG-CO model. Table 2 summarises the set of decision variables that appear in that formulation.

Visits to graphs

To represent the movement of the drone within a graph $g \in \mathcal{G}$, we now introduce some notations related to g. Let $g = (V_g, E_g)$ be a graph in $\mathcal{G}$ whose total length is denoted by $\mathcal{L}(g)$. Here, V_g denotes the set of nodes and E_g denotes the set of edges connecting pairs of nodes. Let e_g be the edge e of graph $g \in G$ and let $\mathcal{L}(e_g)$ be its length. Each edge e_g is parameterised by its endpoints $B^{e_g} = (B^{e_g}(x_1), B^{e_g}(x_2))$ and $C^{e_g} = (C^{e_g}(x_1), C^{e_g}(x_2))$ and we can compute its length $\mathcal{L}(e_g) = \|C^{e_g} - B^{e_g}\|$.

For each edge e_g an indicator binary variable μ^{e_g} is associated, assuming the value one if the drone visits the segment e_g. Furthermore, we define the entry and exit points $R^{e_g} = (B^{e_g}, C^{e_g}, \rho^{e_g})$ and $L^{e_g} = (B^{e_g}, C^{e_g}, \lambda^{e_g})$ that determine the fraction of the edge visited by the drone. The coordinates of the points R^{e_g} and L^{e_g} are given, respectively by

$$R^{e_g} = \rho^{e_g} B^{e_g} + (1 - \rho^{e_g}) C^{e_g} \quad \text{and} \quad L^{e_g} = \lambda^{e_g} B^{e_g} + (1 - \lambda^{e_g}) C^{e_g},$$

where $\rho^{e_g} \in [0,1]$ and $\lambda^{e_g} \in [0,1]$ are variables to determine the position of the points in the segment.

As discussed in Section 2, we consider two modes of visit to the target graphs $g \in \mathcal{G}$:

(i) Visiting a fraction α^{e_g} of each edge e_g which can be modelled by using the following constraints:

$$|\lambda^{e_g} - \rho^{e_g}| \geq \alpha^{e_g}, \quad \forall e_g \in E_g. \tag{α-E}$$

These inequalities state that the difference between the parameterisations of the entry and exit points associated with each edge e_g must be greater than or equal to the fraction of the length of e_g required to be traversed.

(ii) Visiting a fraction α^g of the total length of the graph:

$$\sum_{e_g \in E_g} \mu^{e_g} |\lambda^{e_g} - \rho^{e_g}| \mathcal{L}(e_g) \geq \alpha^g \mathcal{L}(g). \tag{α-G}$$

This constraint ensures that the sum of the length fractions of the edges chosen to be crossed must be greater than or equal to the length fraction of g required to be traversed.

In both cases, the corresponding constraints are non-linear. To linearise them, we need to introduce a binary variable entry^{e_g} that determines the direction of travel on the edge e_g and the definition of the auxiliary variables $v^{e_g}_{\min}$ and $v^{e_g}_{\max}$ of the access and exit points on that segment. Then, for each edge e_g, the absolute value constraint (α-E) can be represented by:

$$|\rho^{e_g} - \lambda^{e_g}| \geq \alpha^{e_g} \iff \begin{cases} \rho^{e_g} - \lambda^{e_g} & = & v^{e_g}_{\max} - v^{e_g}_{\min}, \\ v^{e_g}_{\max} & \leq & 1 - \text{entry}^{e_g}, \\ v^{e_g}_{\min} & \leq & \text{entry}^{e_g}, \\ v^{e_g}_{\min},\ v^{e_g}_{\max} & \geq & 0, \\ v^{e_g}_{\max} + v^{e_g}_{\min} & \geq & \alpha^{e_g}. \end{cases} \tag{α-E}$$

The first four inequalities model the standard trick of linearisation of the absolute value. The last constraint ensures that the value of the linear expression of the absolute value is higher than the required fraction α^{e_g}.

Similarly, (α-G) can be linearised as follows:

$$\sum_{e_g \in E_g} \mu^{e_g} |\rho^{e_g} - \lambda^{e_g}| \mathcal{L}(e_g) \geq \alpha^g \mathcal{L}(g).$$

$$\iff \begin{cases} \rho^{e_g} - \lambda^{e_g} & = & v^{e_g}_{\max} - v^{e_g}_{\min}, \\ v^{e_g}_{\max} & \leq & 1 - \text{entry}^{e_g}, \\ v^{e_g}_{\min} & \leq & \text{entry}^{e_g}, \\ v^{e_g}_{\min},\ v^{e_g}_{\max} & \geq & 0, \\ p^{e_g} & \leq & v^{e_g}_{\max} + v^{e_g}_{\min}, \\ p^{e_g} & \leq & \mu^{e_g}, \\ p^{e_g} & \geq & v^{e_g}_{\max} + v^{e_g}_{\min} + \mu^{e_g} - 1, \\ \sum_{e_g \in E_g} p^{e_g} \mathcal{L}(e_g) & \geq & \alpha^g \mathcal{L}(g), \end{cases} \tag{α-G}$$

where p^{e_g} is the auxiliary variable that represents the product of the binary variable μ^{e_g} and the difference in absolute value $|\rho^{e_g} - \lambda^{e_g}|$. The first four inequalities linearise the expression of the absolute value. The following three constraints model the product of the expression of the absolute value and the binary variable μ^{e_g}. The last inequality ensures that the fraction of the length of those edges chosen to be crossed must be greater than the fraction of the length of g required to be traversed.

Elimination of subtours

To represent the actual routes of drones on the target graph, subtours are not allowed. The reader may note that the subtour elimination constraints are needed to avoid the presence of disconnected paths on the edges of the graph. To prevent the existence of subtours within each graph $g \in \mathcal{G}$ that the drone must visit, we can include, among others, the compact formulation that uses Miller-Tucker-Zemlin constraints (MTZ) or subtour elimination constraints (SEC).

For the MTZ formulation, we use the continuous variables s^{e_g}, defined in Table 2, which state the order of visit to the edge e_g and set the following constraints for each $g \in \mathcal{G}$:

$$s^{e_g} - s^{e'_g} + |E_g| z^{e_g e'_g} \leq |E_g| - 1, \qquad \forall e_g \neq e'_g \in E_g, \tag{MTZ_1}$$

$$0 \leq s^{e_g} \leq |E_g| - 1, \qquad \forall e_g \in E_g. \tag{MTZ_2}$$

Alternatively, we can also use the family of subtour elimination constraints for each $g \in \mathcal{G}$:

$$\sum_{e_g, e'_g \in S} z^{e_g e'_g}_g \leq |S| - 1, \quad \forall S \subset E_g. \tag{SEC}$$

Since there is an exponential number of SEC constraints, when we implement this formulation, we need to perform a row generation procedure including constraints whenever they are required by a separation oracle. To find SEC inequalities, as usual, we search for disconnected components in the current solution. Among them, we choose the shortest subtour found in the solution to be added as a lazy constraint to the model.

3.1. AMMDRPG *with complete overlapping*

To model this problem, we use operations identified with the order in which the different target graphs in the problem are visited. Let us denote by $\mathcal{O}$ the set of operations that the mothership and the drone fleet have to perform. These operations are visits to different graphs in $\mathcal{G}$ with the required constraints. An *operation* $o \in \mathcal{O}$ is referred to as the actions in which the mothership launches some drones from a take-off location, denoted by x^o_L and then takes them back to a retrieval location x^o_R. Here, it is important to realise that both the locations x^o_L and x^o_R must be determined in the continuous space in which the mothership is assumed to move. Note that $|\mathcal{O}| \leq |\mathcal{G}|$, since it is assumed that, for each operation, at least one drone must be launched.

For each operation $o \in \mathcal{O}$, each of the drones launched from the mothership must follow a path starting from and returning to the mothership, while visiting the required edges of one of the graphs $g \in \mathcal{G}$. According to the notation introduced above, we write this generic path in the following form:

$$x^o_L \to R^{e_g} \to L^{e_g} \to \ldots \to R^{e'_g} \to L^{e'_g} \to \ldots \to R^{e''_g} \to L^{e''_g} \to x^o_R.$$

Fig. 3 shows an example of the notation in a configuration with four target graphs that have four nodes and four edges. Here, it is assumed that the number of drones available is equal to two. In particular, Fig. 3(a) represents a feasible solution to the problem for this configuration. The mothership, whose path is represented in black, begins at its starting point *orig*, which coincides with the first launch point x^1_L, where two drones are launched to visit two graphs. There, each drone follows a route (represented by the orange and the green paths) that ensures coverage of one-half of the length of each edge of the graph. The smaller red dots in the visited graphs are the intermediate points R^{e_g} and L^{e_g} used by the drones in their visit to the edges of the different graphs. After finishing the visit of the first two graphs, the drones return to the point x^1_R. The mothership moves from this point to the second launch point x^2_L from where only one drone is launched to visit the third graph. Once this graph has been visited, the drone returns to the mothership at the retrieval point x^2_R. Finally, the mothership moves to the point x^3_L from where a drone is launched for the last visit to the fourth graph. The drone is then retrieved by the mothership at the point x^3_R, and then the mothership ends its route at the destination point *dest*.

Fig. 3(b) represents an optimal solution for the same instance of the problem. We can observe that, in this case, from the first launch point x^1_L only one drone is launched, while from the second x^2_L two drones are launched to visit the second and third graphs. The different position in space of this last point, with respect to the feasible solution reported in Fig. 3(a) whose makespan is 158.36, ensures that the makespan of the optimal solution is equal to 152.39, which is shorter.

To include the definition of these paths in our mathematical programming formulation, we need to make decisions to choose:

(i) The optimal assignment of drones to visit graphs in a given operation o.

(ii) The order to visit the edges of each graph in its corresponding operation.

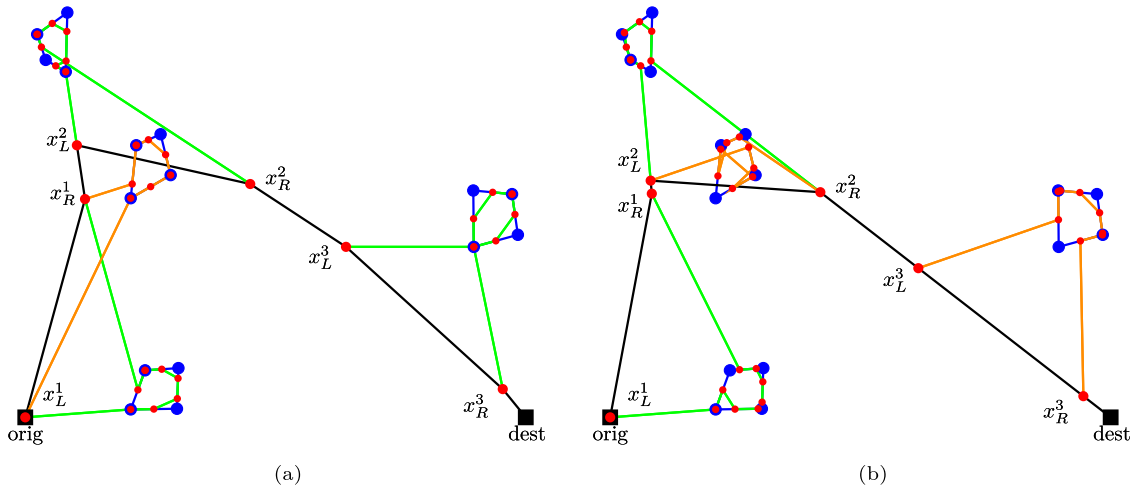


Fig. 3. Problem instance with 4 graphs and 2 drones to visit 50% of each target graph: (a) A feasible solution; and (b) an optimal solution for this case..

Drone constraints

We model the route that the drone follows using the binary variables $u^{e_g o}$, $z^{e_g e'_g}$ and $v^{e_g o}$ defined in Table 2.

$$\sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g o} \le |D|, \qquad \forall o \in \mathcal{O}, \tag{Drone ROUTE$_1$-CO}$$

$$\sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g o} \le |D|, \qquad \forall o \in \mathcal{O}, \tag{Drone ROUTE$_2$-CO}$$

$$\sum_{e_g\in E_g}\sum_{o\in\mathcal{O}} u^{e_g o} = 1, \qquad \forall g \in \mathcal{G}, \tag{Drone ROUTE$_3$-CO}$$

$$\sum_{e_g\in E_g}\sum_{o\in\mathcal{O}} v^{e_g o} = 1, \qquad \forall g \in \mathcal{G}, \tag{Drone ROUTE$_4$-CO}$$

$$\sum_{e_g\in E_g} u^{e_g o} = \sum_{e_g\in E_g} v^{e_g o}, \qquad \forall g \in \mathcal{G}, \forall o \in \mathcal{O}, \tag{Drone ROUTE$_5$-CO}$$

$$\sum_{o\in\mathcal{O}} u^{e_g o} + \sum_{e'_g\in E_g} z_g^{e'_g e_g} = \mu^{e_g}, \qquad \forall e_g \in E_g : g \in \mathcal{G}, \tag{Drone ROUTE$_6$-CO}$$

$$\sum_{o\in\mathcal{O}} v^{e_g o} + \sum_{e'_g\in E_g} z_g^{e_g e'_g} = \mu^{e_g}, \qquad \forall e_g \in E_g : g \in \mathcal{G}. \tag{Drone ROUTE$_7$-CO}$$

The inequalities (Drone ROUTE$_1$-CO) and (Drone ROUTE$_2$-CO) state that it is not possible to use a number of drones larger than the one available in each operation o. Constraints (Drone ROUTE$_3$-CO) and (Drone ROUTE$_4$-CO) ensure that each graph is visited by a drone in an operation o. Eqs. (Drone ROUTE$_5$-CO) ensure that the action of entering and exiting the graph g occurs in the same operation o. Constraints (Drone ROUTE$_6$-CO) state that if a drone visits an edge e of the graph g, one of two alternative situations must occur: e is the first edge of the graph g visited by the drone during operation o, or the edge e is visited by the drone after visiting another edge e' of the graph g. Similarly, the constraints (Drone ROUTE$_7$-CO) state that if a drone visits an edge e of the graph g, e is the last edge of the graph g visited by the drone during operation o, or the drone must move to another edge e' of the graph g after visiting the edge e.

Distance and time constraints

The goal of the AMMDRPG-CO is to find a feasible solution that minimises the makespan. To account for the different distances between the decision variables of the model, we need to set the continuous variables $d_L^{e_g o}$, d^{e_g}, $d^{e_g e'_g}$, $d_R^{e_g o}$, d_{orig}, d_{RL}^o, d_{LR}^o and d_{dest} defined in Table 2 (Blanco et al., 2013). This can be done by means of the following constraints:

$$\|x_L^o - R^{e_g}\| \le d_L^{e_g o}, \qquad \forall e_g \in E_g : g \in \mathcal{G},\ \forall o \in \mathcal{O}, \tag{Drone DIST$_1$-CO}$$

$$\|R^{e_g} - L^{e_g}\| \le d^{e_g}, \qquad \forall e_g \in E_g : g \in \mathcal{G}, \tag{Drone DIST$_2$-CO}$$

$$\|R^{e_g} - L^{e'_g}\| \le d^{e_g e'_g}, \qquad \forall e_g \ne e'_g \in E_g : g \in \mathcal{G}, \tag{Drone DIST$_3$-CO}$$

$$\|L^{e_g} - x_R^o\| \le d_R^{e_o}, \qquad \forall e_g \in E_g : g \in \mathcal{G},\ \forall o \in \mathcal{O}. \tag{Drone DIST$_4$-CO}$$

$$\|orig - x_L^1\| \le d_{orig}, \tag{Mothership DIST$_1$-CO}$$

$$\|x_L^o - x_R^o\| \le d_{LR}^o, \qquad \forall o \in \mathcal{O}, \tag{Mothership DIST$_2$-CO}$$

$$\|x_R^o - x_L^{o+1}\| \le d_{RL}^o, \qquad \forall o \in \mathcal{O} : o < |\mathcal{O}|, \tag{Mothership DIST$_3$-CO}$$

$$\|x_R^{|\mathcal{O}|} - dest\| \le d_{dest}. \tag{Mothership DIST$_4$-CO}$$

All variables that model the distances covered by drones, namely $d_L^{e_g o}$, d^{e_g}, $d^{e'_g e_g}$ and $d_R^{e_g o}$, as well as those modelling the distance travelled by the mothership, namely d_{orig}, d_{LR}^o, d_{RL}^o and d_{dest}, are defined in Table 2.

In order to compute the maximum time a drone spends visiting a graph $g \in \mathcal{G}$ associated with the operation $o, \forall o \in \mathcal{O}$, we introduce the following constraints:

$$time_D^o \ge \frac{1}{v_D}\left(\sum_{e_g\in E_g} u^{e_g o} d_L^{e_g o} + \sum_{e_g, e'_g\in E_g} z^{e_g e'_g} d^{e_g e'_g} + \sum_{e_g\in E_g} \mu^{e_g} d^{e_g} + \sum_{e_g\in E_g} v^{e_g o} d_R^{e_g o}\right) - N_D\left(1 - \sum_{e_g\in E_g} u^{e_g o}\right). \tag{Drone TIME$_o$-CO}$$

The first addend in the brackets represents the time that the drone spends transferring from the launch point x_L^o to the first retrieval point in the graph R^{e_g}. The second addend considers the time consumed by the drone to go from edge e_g to edge e'_g on graph g. The third computes the time required to traverse the required edges in g. The fourth measures the time taken to travel from the last launching point $L^{e''_g}$ to the retrieval point x_R^o. Note that, in the special case where all edges must be visited, the third sum on the right-hand side of the constraint (Drone TIME$_o$-CO) reduces to $\sum_{e_g\in E_g} d^{e_g}$ setting all variables μ^{e_g} equal to one.

The endurance term in the constraint (Drone TIME$_o$-CO) ensures that the constraint becomes active only when a graph g is visited during

the operation o. The reader may observe that the endurance constraint (Endurance-CO) restricts the time the drone spends performing the operation o to be less than the endurance N_D. Therefore, the constant N_D can be taken as the bigM term in the constraint (Drone TIME$_o$-CO).

Note that, to deal with the bilinear terms of the constraint (Drone TIME$_o$-CO), we use McCormick's envelope to linearise them by adding variables $p \geq 0$ representing the products and introducing the following constraints:

$$p \leq Mz,$$
$$p \leq d,$$
$$p \geq mz,$$
$$p \geq d - M(1-z),$$

where m and M are, respectively, the lower and upper bounds of the distance variable d. These bounds will be adjusted for each bilinear term in Section 4.

The constraint (Mothership TIME$_o$-CO) defines the time the mothership must spend to go from the launch point x^o_L to the retrieval point x^o_R associated with the operation o:

$$time^o_M = \frac{d^o_{LR}}{v_M}, \quad \forall o \in \mathcal{O}. \qquad \text{(Mothership TIME}_o\text{-CO)}$$

Thus, the overall time spent by the mothership to move from the origin to the destination (makespan) can be expressed as follows:

$$time_M = \frac{1}{v_M}(d_{orig} + \sum_{o\in\mathcal{O}} d^o_{LR} + \sum_{o\in\mathcal{O}:o<|\mathcal{O}|} d^o_{RL} + d_{dest}).$$

(Mothership TIME-CO)

Coordination and endurance constraints

The coordination between the drones and the mothership must ensure that the maximum time $time^o_D$ spent by a drone to visit a graph g at operation o is less than or equal to the time that the mothership needs to move from the launching point to the retrieval point during operation o. To this end, we need to define the following coordination constraint for each operation $o \in \mathcal{O}$:

$$time^o_D \leq time^o_M. \qquad \text{(DCW-CO)}$$

We can model the time endurance constraint for a particular operation $o \in \mathcal{O}$ by limiting the time travelled by the drone for this operation o:

$$time^o_D \leq N_D. \qquad \text{(Endurance-CO)}$$

AMMDRPG-complete overlapping formulation

Combining all the constraints introduced hitherto, the following formulation minimises the makespan, ensuring coordination with the drone fleet while guaranteeing the required coverage of the target graphs.

$$\min \quad time_M \qquad \text{(AMMDRPG-CO)}$$

s.t. (α-E) or (α-G),
(MTZ_1)–(MTZ_2) or (SEC),
(Drone ROUTE$_1$-CO)–(Drone ROUTE$_7$-CO),
(Drone DIST$_1$-CO)–(Drone DIST$_4$-CO),
(Mothership DIST$_1$-CO)–(Mothership DIST$_4$-CO),
(Drone TIME$_o$-CO), (Mothership TIME$_o$-CO), (Mothership TIME-CO),
(DCW-CO), (Endurance-CO).

The objective function accounts for the makespan. Constraints (Drone ROUTE$_1$-CO)–(Drone ROUTE$_7$-CO) model the route followed by the drones, (MTZ_1)–(MTZ_2) or (SEC) ensure that the displacement of a drone assigned to the target graph $g \in \mathcal{G}$ is a route, (α-E) or (α-G) define what is required in each visit to a target graph. Constraints (Drone DIST$_1$-CO)–(Drone DIST$_4$-CO) set the variables $d^{e_g o}_L$, d^{e_g}, $d^{e_g e'_g}$, $d^{e_g o}_R$. The mothership distances d^o_{RL} and d^o_{LR}, are defined by means of constraints (Mothership DIST$_1$-CO)–(Mothership DIST$_4$-CO). Constraints (Drone TIME$_o$-CO), (Mothership TIME$_o$-CO) and (Mothership TIME-CO) define times travelled by the drones and the mothership. Finally, constraints (DCW-CO)–(Endurance-CO) guarantee that coordination and drone endurance are satisfied.

3.2. The AMMDRPG with partial overlapping

In the AMMDRPG-CO version of the problem, we assume that every drone is launched and retrieved in the same operation. In this subsection, we show how this assumption can be relaxed. We consider a variant of the model presented in Section 3.1, in which we assume that the mothership can retrieve one drone in a different phase from that in which it has been launched. That is, the mothership can move to another point to launch a new drone without having retrieved all the drones that were launched previously.

In the following formulation, we use the concept of *stage* to refer to the action of launching or receiving a drone by the mothership. Each graph must be visited by a drone so that each operation gives rise to two stages: one when the drone is launched and another one, once the same drone has been retrieved by the mothership. We denote by $\mathcal{T}$ the set of stages. It is clear that $|\mathcal{T}| = 2|\mathcal{G}|$. Using the concept of stage, we can substitute the set of operations with the set of stages to model the coordination between drones and mothership in the partial overlapping version of the problem. Indeed, in this case, different from the complete overlapping version of the problem, the launch of a drone is not necessarily followed by its retrieval but, for example, by the launch of a different drone to visit another target graph, as shown in Fig. 5. We notice that when the fleet of drones consists of only one drone, the two versions of the problem coincide. Table 3 summarises all the variables used in our formulation for the AMMDRPG-PO model.

Drone constraints

Similarly to the complete overlapping version of the problem, we model the route followed by the drone using the binary variables $u^{e_g t}$, $v^{e_g t}$ and $z^{e_g e'_g}$. However, in this case, the variables $u^{e_g t}$ and $v^{e_g t}$ are associated with the stage t and due to the assumptions of the problem, we need to introduce additional binary variables γ^{gt}. Thus, the following constraints model the route followed by the drone while operating on a graph $g \in \mathcal{G}$:

$$\sum_{t\in\mathcal{T}} \sum_{e_g\in E_g} u^{e_g t} = 1, \qquad \forall g \in \mathcal{G},$$

(Drone ROUTE$_1$)

$$\sum_{t\in\mathcal{T}} \sum_{e_g\in E_g} v^{e_g t} = 1, \qquad \forall g \in \mathcal{G},$$

(Drone ROUTE$_2$)

$$\sum_{g\in\mathcal{G}} \sum_{e_g\in E_g} u^{e_g t} \leq \mathcal{K}(t), \qquad \forall t \in \mathcal{T},$$

(Drone ROUTE$_3$)

$$\sum_{g\in\mathcal{G}} \sum_{e_g\in E_g} (u^{e_g t} + v^{e_g t}) \leq 1, \qquad \forall t \in \mathcal{T},$$

(Drone ROUTE$_4$)

$$\sum_{e_g\in E_g} u^{e_g t} \leq \sum_{e_g\in E_g} \sum_{t'\in\mathcal{T}:t'>t} v^{e_g t'}, \qquad \forall g \in \mathcal{G},\ \forall t \in \mathcal{T},$$

(Drone ROUTE$_5$)

$$\sum_{t\in\mathcal{T}} u^{e_g t} + \sum_{e'_g\in E_g} z^{e'_g e_g} = \mu^{e_g}, \qquad \forall e_g \in E_g :\ g \in \mathcal{G},$$

(Drone ROUTE$_6$)

$$\sum_{t\in\mathcal{T}} v^{e_g t} + \sum_{e'_g\in E_g} z^{e_g e'_g} = \mu^{e_g}, \qquad \forall e_g \in E_g :\ g \in \mathcal{G},$$

(Drone ROUTE$_7$)

Table 3
Decision variables for AMMDRPG-PO.

Binary and integer decision variables
$\mu^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G})$: equal to 1 if edge e of graph g (or a portion of it) is visited by the drone, 0 otherwise.
$\text{entry}^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G})$: auxiliary binary variable used for linearising expressions.
$u^{e_g t} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall t \in \mathcal{T}$: equal to 1 if the visit of graph g starts in stage t from edge e_g, 0 otherwise.
$z^{e_g e'_g} \in \{0,1\},\ \forall e_g, e'_g \in E_g\ (g \in \mathcal{G})$: equal to 1 if the drone goes from e_g to e'_g, 0 otherwise.
$\gamma^{gt} \in \{0,1\},\ \forall g \in \mathcal{G},\ \forall t \in \mathcal{T}$: equal to 1 if the operation of visiting graph g continues when stage t occurs, 0 otherwise.
$v^{e_g t} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall t \in \mathcal{T}$: equal to 1 if the visit of graph g ends in stage t on edge e_g, 0 otherwise.
$y^t_{LL} \in \{0,1\},\ \forall t \in \mathcal{T} : t < \lvert\mathcal{T}\rvert$: equal to 1 if the mothership moves from a launching point to a launching point between stage t and stage $t+1$, 0 otherwise.
$y^t_{LR} \in \{0,1\},\ \forall t \in \mathcal{T} : t < \lvert\mathcal{T}\rvert$: equal to 1 if the mothership moves from a launching point to a retrieval point between stage t and stage $t+1$, 0 otherwise.
$y^t_{RL} \in \{0,1\},\ \forall t \in \mathcal{T} : t < \lvert\mathcal{T}\rvert$: equal to 1 if the mothership moves from a retrieval point to a launching point between stage t and stage $t+1$, 0 otherwise.
$y^t_{RR} \in \{0,1\},\ \forall t \in \mathcal{T} : t < \lvert\mathcal{T}\rvert$: equal to 1 if the mothership moves from a retrieval point to a retrieval point between stage t and stage $t+1$, 0 otherwise.
$\mathcal{K}(t) \in \{0,1,2,\dots,\lvert D\rvert\},\ \forall t \in \mathcal{T}$: integer non-negative variable representing the number of available drones at stage t.
Continuous decision variables
$s^{e_g} \in [0, \lvert E_g\rvert - 1],\ \forall e_g \in E_g\ (g \in \mathcal{G})$: continuous non-negative variable representing the order of visit of the edge e of graph g.
$x^t_L \in \mathbb{R}^2,\ \forall t \in \mathcal{T}$: coordinates representing the launching point visited by the mothership at stage t.
$x^t_R \in \mathbb{R}^2,\ \forall t \in \mathcal{T}$: coordinates representing the retrieval point visited by the mothership at stage t.
$R^{e_g} \in \mathbb{R}^2,\ \forall e_g \in E_g\ (g \in \mathcal{G})$: coordinates representing the entry point on edge e_g of graph g.
$L^{e_g} \in \mathbb{R}^2,\ \forall e_g \in E_g\ (g \in \mathcal{G})$: coordinates representing the exit point on edge e_g of graph g.
$d^{e_g t}_L \geq 0,\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall t \in \mathcal{T}$: representing the distance travelled by the drone from the launching point x^t_L on the mothership at stage t to the first visiting point R^{e_g} on e_g.
$d^{e_g} \geq 0,\ \forall e_g \in E_g\ (g \in \mathcal{G})$: representing the distance travelled by the drone from the retrieval point R^{e_g} to the launching point L^{e_g} on e_g.
$d^{e_g e'_g} \geq 0,\ \forall e_g, e'_g \in E_g\ (g \in \mathcal{G})$: representing the distance travelled by the drone from the launching point L^{e_g} on e_g to the retrieval point $R^{e'_g}$ on e'_g.
$d^{e_g t}_R \geq 0,\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall t \in \mathcal{T}$: representing the distance travelled by the drone from the last visiting point L^{e_g} on e_g to the retrieval point x^o_R on the mothership at stage t.
$d_{orig} \geq 0$: distance from the origin $orig$ to the first launching point x^1_L.
$d^t_{LL} \geq 0,\ \forall t \in \mathcal{T} : t < \lvert\mathcal{T}\rvert$: distance from the launching point x^t_L to the launching point x^{t+1}_L.
$d^t_{LR} \geq 0,\ \forall t \in \mathcal{T} : t < \lvert\mathcal{T}\rvert$: distance from the launching point x^t_L to the retrieval point x^{t+1}_R.
$d^t_{RL} \geq 0,\ \forall t \in \mathcal{T} : t < \lvert\mathcal{T}\rvert$: distance from the retrieval point x^t_R to the launching point x^{t+1}_L.
$d^t_{RR} \geq 0,\ \forall t \in \mathcal{T} : t < \lvert\mathcal{T}\rvert$: distance from the retrieval point x^t_R to the retrieval point x^{t+1}_R.
$d_{dest} \geq 0$: distance from the last retrieval point $x^{\lvert\mathcal{T}\rvert}_R$ to the destination $dest$.
$d^g_{LR} \geq 0,\ \forall g \in \mathcal{G}$: representing the distance travelled by the mothership from the launching point x^t_L to the retrieval point $x^{t'}_R$ associated with graph g for some $t, t' \in \mathcal{T}$.
$time^g_M \geq 0,\ \forall g \in \mathcal{G}$: time spent by the mothership while graph g is visited by a drone.
$time^g_D \geq 0,\ \forall g \in \mathcal{G}$: time spent by a drone to visit graph g.
$time_M \geq 0$: total time spent by the mothership to go from the origin to the destination (makespan).

$$\gamma^{gt} \geq \sum_{e_g \in E_g} u^{e_g t}, \qquad \forall g \in \mathcal{G},\ \forall t \in \mathcal{T}, \tag{Drone ROUTE$_8$}$$

$$\gamma^{g(t+1)} \geq \gamma^{gt} - \sum_{e_g \in E_g} v^{e_g (t+1)}, \qquad \forall g \in \mathcal{G},\ \forall t \in \mathcal{T} : t < \lvert\mathcal{T}\rvert, \tag{Drone ROUTE$_9$}$$

$$\sum_{t' \in \mathcal{T} : t' < t} \gamma^{gt'} \leq (t-1)\Big(1 - \sum_{e_g \in E_g} u^{e_g t}\Big), \qquad \forall g \in \mathcal{G},\ \forall t \in \mathcal{T}, \tag{Drone ROUTE$_{10}$}$$

$$\sum_{t' \in \mathcal{T} : t' \geq t} \gamma^{gt'} \leq (\lvert\mathcal{T}\rvert - t + 1)\Big(1 - \sum_{e_g \in E_g} v^{e_g t}\Big), \qquad \forall g \in \mathcal{G},\ \forall t \in \mathcal{T}, \tag{Drone ROUTE$_{11}$}$$

$$\mathcal{K}(1) = \lvert D\rvert, \tag{Drone ROUTE$_{12}$}$$

$$\mathcal{K}(t+1) = \mathcal{K}(t) + \sum_{g \in \mathcal{G}} \sum_{e_g \in E_g} v^{e_g t} - \sum_{g \in \mathcal{G}} \sum_{e_g \in E_g} u^{e_g t}, \qquad \forall t \in \mathcal{T} : t < \lvert\mathcal{T}\rvert. \tag{Drone ROUTE$_{13}$}$$

The constraints (Drone ROUTE$_1$) and (Drone ROUTE$_2$) ensure that a launch point and a retrieval point are associated with each graph g. The restrictions (Drone ROUTE$_3$) allow the mothership to launch a drone in stage t only if a drone is available when stage t occurs. Constraints (Drone ROUTE$_4$) guarantee that a launch or a retrieval occurs at each stage $t \in \mathcal{T}$. Constraints (Drone ROUTE$_5$) indicate that the retrieval stage associated with the graph g occurs after the launch stage associated with the same graph g. Eqs. (Drone ROUTE$_6$) state that if a drone visits an edge e of the graph g, either e is the first edge of the graph g visited by the drone at stage t, or the drone visits the edge e after visiting another edge e' of the graph g. Similarly, constraints (Drone ROUTE$_7$) state that if a drone visits an edge e of the graph g, e is the last edge of the graph g visited by the drone at stage t, or the drone must move to another edge e' of the graph g after visiting the edge e. Constraints (Drone ROUTE$_8$) ensure that the operation associated with graph g starts when the drone is launched during the stage t. Inequalities (Drone ROUTE$_9$) state that the drone is still operating in the graph g for successive stages until it is retrieved at the stage t. Constraints (Drone ROUTE$_{10}$) ensure that drone does not operate in g until launch stage occurs. Constraints (Drone ROUTE$_{11}$) guarantee that the drone finishes operating in graph g when retrieval stage occurs. Finally, the constraints (Drone ROUTE$_{12}$) and (Drone ROUTE$_{13}$) model the number of drones available at the stage t.

Mothership constraints

This subsection models all possible sequences of stages in terms of launching and retrieval that can be followed by the mothership: launching-launching, launching-retrieval, retrieval-launching, and retrieval-retrieval.

$$y^1_{LL} + y^1_{LR} = 1, \tag{Mothership ROUTE$_1$}$$

$$y^{t+1}_{LL} + y^{t+1}_{LR} \geq y^t_{RL} + y^t_{LL}, \qquad \forall t \in \mathcal{T} : t < \lvert\mathcal{T}\rvert, \tag{Mothership ROUTE$_2$}$$

$$y^{t+1}_{RR} + y^{t+1}_{RL} \geq y^t_{LR} + y^t_{RR}, \qquad \forall t \in \mathcal{T} : t < \lvert\mathcal{T}\rvert, \tag{Mothership ROUTE$_3$}$$

$$y_{LR}^{|\mathcal{T}|-1} + y_{RR}^{|\mathcal{T}|-1} = 1. \qquad \text{(Mothership ROUTE}_4\text{)}$$

The constraints (Mothership ROUTE$_1$) state that at stage 1 the mothership must depart from the launch point x_L^1. Constraints (Mothership ROUTE$_2$) (resp. (Mothership ROUTE$_3$)) ensure that if the mothership goes to the launching point (resp. retrieval) x_L^{t+1} (resp. x_R^{t+1}) then in the next stage it must depart from x_L^{t+1} (resp. x_R^{t+1}). The constraint (Mothership ROUTE$_4$) guarantees that the path followed by the mothership ends at the retrieval point $x_R^{|\mathcal{T}|}$.

Distance and time constraints

This subsection considers the second-order cone constraints that model the distances covered by the drones and the mothership:

$$\|x_L^t - R^{e_g}\| \le d_L^{e_g t}, \qquad \forall e_g \in E_g : g \in \mathcal{G},\ \forall t \in \mathcal{T}, \qquad \text{(Drone DIST}_1\text{)}$$

$$\|R^{e_g} - L^{e_g}\| \le d^{e_g}, \qquad \forall e_g \in E_g : g \in \mathcal{G}, \qquad \text{(Drone DIST}_2\text{)}$$

$$\|R^{e_g} - L^{e'_g}\| \le d^{e_g e'_g}, \qquad \forall e_g \ne e'_g \in E_g : g \in \mathcal{G}, \qquad \text{(Drone DIST}_3\text{)}$$

$$\|L^{e_g} - x_R^t\| \le d_R^{e_g t}, \qquad \forall e_g \in E_g : g \in \mathcal{G},\ \forall t \in \mathcal{T}. \qquad \text{(Drone DIST}_4\text{)}$$

$$\|orig - x_L^1\| \le d_{orig}, \qquad \text{(Mothership DIST}_1\text{)}$$

$$\|x_L^t - x_L^{t+1}\| \le d_{LL}^t, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Mothership DIST}_2\text{)}$$

$$\|x_L^t - x_R^{t+1}\| \le d_{LR}^t, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Mothership DIST}_3\text{)}$$

$$\|x_R^t - x_L^{t+1}\| \le d_{RL}^t, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Mothership DIST}_4\text{)}$$

$$\|x_R^t - x_R^{t+1}\| \le d_{RR}^t, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Mothership DIST}_5\text{)}$$

$$\|x_R^{|\mathcal{T}|} - dest\| \le d_{dest}. \qquad \text{(Mothership DIST}_6\text{)}$$

All variables that model the distances covered by drones, namely $d_L^{e_g t}$, d^{e_g}, $d^{e'_g e_g}$, and $d_R^{e_g t}$, as well as those that model the distances travelled by the mothership, namely d_{orig}, d_{LL}^t, d_{LR}^t, d_{RL}^t, d_{RR}^t and d_{dest}, are defined in Table 3.

The time that the drone spends performing the operation of visiting the graph g is given by:

$$time_D^g = \frac{1}{v_D}\left(\sum_{t\in\mathcal{T}}\sum_{e_g\in E_g} u^{e_g t} d_L^{e_g t} + \sum_{e_g, e'_g \in E_g} z^{e_g e'_g} d^{e_g e'_g} + \sum_{e_g \in E_g} \mu^{e_g} d^{e_g} + \sum_{t\in\mathcal{T}}\sum_{e_g\in E_g} v^{e_g t} d_R^{e_g t}\right).$$

(Drone TIME$_g$)

The time spent by the mothership while the drone is operating in graph g is given by:

$$time_M^g = \frac{1}{v_M} d_{LR}^g = \frac{1}{v_M} \sum_{t\in\mathcal{T}: t<|\mathcal{T}|} (\|x_L^t - x_L^{t+1}\| y_{LL}^t + \|x_L^t - x_R^{t+1}\| y_{LR}^t + \|x_R^t - x_L^{t+1}\| y_{RL}^t + \|x_R^t - x_R^{t+1}\| y_{RR}^t)\gamma^{gt}, \quad \forall g \in \mathcal{G}.$$

(Mothership TIME$_g$)

Finally, the overall time spent by the mothership (makespan) can be described as follows:

$$time_M = \frac{1}{v_M}\left(d_{orig} + \sum_{t\in\mathcal{T}: t<|\mathcal{T}|} \left(\|x_L^t - x_L^{t+1}\| y_{LL}^t + \|x_L^t - x_R^{t+1}\| y_{LR}^t + \|x_R^t - x_L^{t+1}\| y_{RL}^t + \|x_R^t - x_R^{t+1}\| y_{RR}^t\right) + d_{dest}\right).$$

(Mothership TIME)

Coordination and endurance constraints

Once having defined the time the drone spends to visit g and the time spent by the mothership while the drone is visiting this graph g, we can model the coordination constraint simply as follows:

$$time_D^g \le time_M^g, \quad \forall g \in \mathcal{G}. \qquad \text{(DCW)}$$

In addition, the time the drone spends to operate on the graph g must not exceed its endurance:

$$time_D^g \le N_D \qquad \text{(Endurance)}$$

Linearisation constraints

This subsection is devoted to linearising the relationship between the decision variables that model the route of the mothership and the drones. The relationship of these variables is given by the following non-linear expressions:

$$y_{LL}^t = \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g t} \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g (t+1)}, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|,$$

$$y_{LR}^t = \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g t} \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g (t+1)}, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|,$$

$$y_{RL}^t = \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g t} \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g (t+1)}, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|,$$

$$y_{RR}^t = \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g t} \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g (t+1)} \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|.$$

The products above can be linearised, respectively, by means of the following constraints:

$$y_{LL}^t \le \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g t}, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Linearization}_1\text{)}$$

$$y_{LL}^t \le \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g (t+1)}, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Linearization}_2\text{)}$$

$$y_{LL}^t \ge \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g t} + \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g (t+1)} - 1, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Linearization}_3\text{)}$$

$$y_{LR}^t \le \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g t}, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Linearization}_4\text{)}$$

$$y_{LR}^t \le \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g (t+1)}, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Linearization}_5\text{)}$$

$$y_{LR}^t \ge \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g t} + \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g (t+1)} - 1, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Linearization}_6\text{)}$$

$$y_{RL}^t \le \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g t}, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Linearization}_7\text{)}$$

$$y_{RL}^t \le \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g (t+1)}, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Linearization}_8\text{)}$$

$$y_{RL}^t \ge \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g t} + \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g (t+1)} - 1, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Linearization}_9\text{)}$$

$$y_{RR}^t \le \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g t}, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Linearization}_{10}\text{)}$$

$$y_{RR}^t \le \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g (t+1)}, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|, \qquad \text{(Linearization}_{11}\text{)}$$

$$y^t_{RR} \geq \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g t} + \sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g (t+1)} - 1, \qquad \forall t \in \mathcal{T} : t < |\mathcal{T}|.$$

(Linearization$_{12}$)

AMMDRPG-Partial overlapping formulation

Hence, the formulation of the AMMDRPG with partial overlapping operations is as follows:

min $time_M$ (AMMDRPG-PO)

s.t. (α-E) or (α-G),
(MTZ_1)–(MTZ_2) or (SEC),
(Drone ROUTE$_1$)–(Drone ROUTE$_{13}$),
(Mothership ROUTE$_1$)–(Mothership ROUTE$_4$),
(Drone DIST$_1$)–(Drone DIST$_1$),
(Mothership DIST$_1$)–(Mothership DIST$_6$),
(Drone TIME$_g$), (Mothership TIME$_g$), (Mothership TIME),
(DCW), (Endurance),
(Linearization$_1$)–(Linearization$_{12}$)

3.3. Relationship between problem variants

In this section, we present two results that link the two models presented above. Note that the only difference between the solutions of these models is that, for the partial overlapping case, the mothership can launch a second drone sequentially before retrieving those that were launched previously. Fig. 5 shows a solution that is not possible for the model with complete overlapping. In fact, we can see that the first drone is launched at x^1_L to visit P_1 and retrieved at x^1_R. However, the mothership has launched another drone at x^2_L that goes to visit P_2 before retrieved the first drone. Clearly, this solution does not satisfy the assumptions of the complete overlapping model.

Theorem 3.1. *Let X_{CO} be the feasible set of the* AMMDRPG *with complete overlapping operations, and let X_{PO} be the feasible set of the* AMMDRPG *with partial overlapping operations, then:*

$$X_{CO} \subsetneq X_{PO}.$$

Proof. To prove the theorem, we first show that a feasible solution $\bar{\omega} \in X_{CO}$ is also feasible for the AMMDRPG-PO. We can notice that all the discrete decision variables of the AMMDRPG-PO model can be obtained directly once the variables $\hat{u}^{e_g t}$ and $\hat{v}^{e_g t}$ are set through the constraints (Drone ROUTE$_8$)–(Drone ROUTE$_{13}$). Thus, we can limit ourselves to showing how their values can be derived from those of $\bar{\omega}$ to obtain a feasible solution $\hat{\omega} \in X_{PO}$. We consider $\bar{u}^{e_g o}$ and $\bar{v}^{e_g o}$ equal to 1. Let $\bar{\mathcal{O}} = \{o \in \mathcal{O} : \bar{u}^{e_g o} = 1\}$. Let $\bar{\mathcal{G}}(o)$ be the set of graphs visited in operation $o \in \bar{\mathcal{O}}$. We can compute for each $o \in \bar{\mathcal{O}}$ the corresponding set $\mathcal{T}(o)$, that is, the set of stages that define the launch and retrieval actions that occur in operation o. More in detail, we can identify the first element $t(o)$ of this set as follows:

$$t(0) = 1;$$

$$t(o+1) = t(o) + \sum_{g\in\bar{\mathcal{G}}(o)}\sum_{e_g\in E_g} (\bar{u}^{e_g o} + \bar{v}^{e_g o}).$$

Given its first element $t(o)$, we can split $\mathcal{T}(o)$ into two subsets of the indexes $\mathcal{T}_u(o)$ and $\mathcal{T}_v(o)$ as follows:

$$\mathcal{T}_u(o) = \{t \in \mathcal{T} : t(o) \leq t \leq t(o) + |\bar{\mathcal{G}}(o)| - 1\},$$

$$\mathcal{T}_v(o) = \{t \in \mathcal{T} : t(o) \leq t - |\bar{\mathcal{G}}(o)| \leq t(o) + |\bar{\mathcal{G}}(o)| - 1\}.$$

Since the cardinality of the set $\mathcal{T}_u(o)$ is equal to the cardinality of the set $\bar{\mathcal{G}}(o)$, we can define a bijective function $\bar{\varphi}_{u(o)} : \mathcal{T}_u(o) \to \bar{\mathcal{G}}(o)$ and similarly we can define a bijective function $\bar{\varphi}_{v(o)} : \mathcal{T}_v(o) \to \bar{\mathcal{G}}(o)$. These functions define the assignment between the graphs and stages. Note that any assignment defined by the functions $\bar{\varphi}_{u(o)}$ and $\bar{\varphi}_{v(o)}$ is feasible.

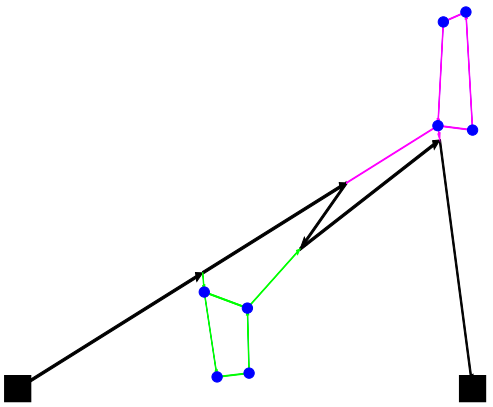


Fig. 4. Feasible solution with partial overlapping that is not feasible for the complete overlapping model.

Using these two functions, we can set the values of the $\hat{u}^{e_g t}$ and $\hat{v}^{e_g t}$ variables. Indeed, by resorting to the Graph of the functions $\bar{\varphi}_{u(o)}$ and $\bar{\varphi}_{v(o)}$ we can define, respectively, the variables $\hat{u}^{e_g t}$ and $\hat{v}^{e_g t}$ that must be equal to 1:

$$\hat{u}^{e_g t} = 1, \quad (t, g) \in \text{Graph}(\bar{\varphi}_{u(o)}) \wedge (\bar{u}^{e_g o} = 1)$$

$$\hat{v}^{e_g t} = 1, \quad (t, g) \in \text{Graph}(\bar{\varphi}_{v(o)}) \wedge (\bar{v}^{e_g o} = 1)$$

The remaining $\hat{u}^{e_g t}$ and $\hat{v}^{e_g t}$ variables are set equal to 0. To show that binary variables $\hat{u}^{e_g t}$ and $\hat{v}^{e_g t}$ are feasible for the AMMDRPG-PO model, it can be easily checked they satisfy the constraints (Drone ROUTE$_1$)–(Drone ROUTE$_7$).

Moreover, it is easy to show that the mothership constraints are also satisfied by the variables $\hat{y}^t = (\hat{y}^t_{LL}, \hat{y}^t_{LR}, \hat{y}^t_{RL}, \hat{y}^t_{RR})$, induced by the variables $\hat{u}^{e_g t}$ and $\hat{v}^{e_g t}$.

With regard to continuous variables, they can be directly derived from the setting of variables $\hat{x}^t_L$ and $\hat{x}^t_R$ which can be obtained as follows:

$$\hat{x}^t_L = \bar{x}^o_L, \quad \forall t \in \mathcal{T}_u(o) : o \in \bar{\mathcal{O}},$$

$$\hat{x}^t_R = \bar{x}^o_R, \quad \forall t \in \mathcal{T}_v(o) : o \in \bar{\mathcal{O}}.$$

We can see that the distances between two consecutive launch points or two consecutive retrieval points are equal to 0 by the definition of the variables $\hat{x}^t_L$ and $\hat{x}^t_R$. Consequently, the time $\widehat{time}^g_M$ spent by the mothership while the drone visits the graph $g \in \bar{\mathcal{G}}(o) : o \in \bar{\mathcal{O}}$ is equal to $\overline{time}^o_M$.

To complete the proof, it suffices to notice that, in contrast, there exist feasible solutions of the AMMDRPG-PO characterised by partial overlaps between operations, as shown, for example, in Fig. 4, which are not feasible for the AMMDRPG-CO. □

To present our next result, wlog, we restrict ourselves to the degenerate case where the graphs are reduced to points. The reader may note that it is possible to reduce the visit of graphs to the visit of points by assuming that the drone is stopped at the point which is at the same time as the one required to traverse the edges of the graph. We simplify the proof by considering a generic solution between two consecutive target points.

Theorem 3.2. *Let x^1_L, x^2_L (resp. x^1_R, x^2_R) be the launch (resp. retrieval) points associated with the visit to the target points P_1 and P_2. If there exist*

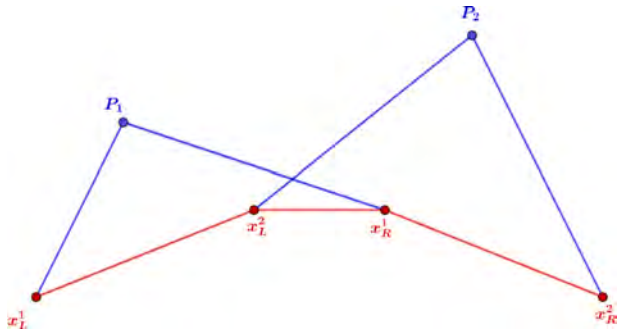


Fig. 5. The mothership launches two drones sequentially.

two points x_L and x_R verifying

$$\begin{cases} \frac{\|x_L - x_R\|}{v_M} & \leq & \frac{\|x_L - P_1\| + \|P_1 - x_R\|}{v_D}, \\ \frac{\|x_L - x_R\|}{v_M} & \leq & \frac{\|x_L - P_2\| + \|P_2 - x_R\|}{v_D}, \\ \frac{\|x_L - x_R\|}{v_M} & \leq & N_D, \\ \|x_L - x_R\| & \leq & \|x_L^1 - x_L^2\| + \|x_L^2 - x_R^1\| + \|x_R^1 - x_R^2\|, \end{cases}$$

then the contribution of this partial route to the optimal objective value will be the same in both models.

Proof. Note that, in the configuration considered, the order of visits to the points P_1 and P_2 is fixed, and then the binary variables in the model are fixed in this case. Thus, the only differences that the two models can have are the location of the launch and retrieval points. Therefore, the only constraints involved are those related to these points. These are the conditions in the following statement: The first two are the (DCW-CO) inequalities. The third is the constraint (Endurance-CO) and the last ensures that the distance travelled by the mothership in the complete overlapping model is smaller than or equal to the distance assumed in the partial overlapping solution described in the statement. Therefore, the conclusion follows. □

Note that this result states sufficient conditions to obtain the same solution for the two models.

4. Strengthening the formulations

In this section, we present some valid inequalities for (AMMDRPG-CO) that reinforce the formulation given in Section 3.1. Moreover, constraints (DCW-CO) and (DCW) have products of binary and continuous variables that, when linearised, produce bigM constants that must be tightened. This section also provides some bounds for these constants when it is possible.

4.1. Valid inequalities for the (AMMDRPG-CO)

In this problem, we assume that the fleet has more than one drone, since otherwise the problem reduces to the *All Terrain Mothership and Drone Routing Problem with Graphs* that was already studied in Amorosi et al. (2021). Therefore, if there exists an operation in which more than one drone is launched, the mothership does not need to perform $|\mathcal{G}|$ different operations. Therefore, most likely, the model does not need to deal with those operations that are numbered at the end. By exploiting this idea, it is possible to concentrate all drone activities on the first operations, avoiding empty operations in $\mathcal{O}$.

Let β^o be a binary variable that assumes the value of 1 if all target graphs are visited when the operation o begins, and zero otherwise. Note that if all graphs have already been visited before operation o then they were also completed before operation $o+1$. Therefore, the variables β must satisfy the following constraints:

$$\beta^o \leq \beta^{o+1}, \text{ for all } o = 1, \dots, |\mathcal{G}| - 1. \qquad \text{(Monotonicity)}$$

Let k^o denote the number of graphs visited in operation o. This number can be computed using variables u since $u^{e_g o}$ takes the value of 1 if the graph g is visited in operation o. Thus:

$$k^o = \sum_{g \in \mathcal{G}} \sum_{e_g \in E_g} u^{e_g o}.$$

Hence, if β^o is equal to one, the entire set of graphs in $\mathcal{G}$ must have been visited before operation o:

$$\sum_{o'=1}^{o-1} k^{o'} \geq |\mathcal{G}|\beta^o, \quad \forall o \in \mathcal{O}, \qquad \text{(VI-1)}$$

where $|\mathcal{G}|$ denotes the number of graphs of $\mathcal{G}$.

To reduce the space of feasible solutions, we can assume without loss of generality that it is not allowed to have an operation o without visiting graphs if some of them are still to be visited. This can be enforced by the following constraints:

$$k^o \geq 1 - \beta^o, \quad \forall o \in \mathcal{O}. \qquad \text{(VI-2)}$$

The model we propose includes bigM constants. In this work, we define different bigM constants. To strengthen the formulations, we provide tight upper and lower bounds for these constants. In this section, we present some results that adjust them for each of the models. The reader may note that the same bounds can be used for both models. Therefore, wlog, we focus on the bigM constants that appear in (AMMDRPG-CO).

Big M constants bounding the distance from the launch/retrieval point on the path followed by the mothership to the retrieval/launch point on the target graph $g \in \mathcal{G}$

To linearise the first addend in (DCW-CO), we define the non-negative continuous auxiliary variables $p_L^{e_g o}$ (resp. $p_R^{e_g o}$) and we model the product by including the following constraints:

$$\begin{aligned} & p_L^{e_g o} \leq M_L^{e_g o} u^{e_g o}, \\ & p_L^{e_g o} \leq d_L^{e_g o}, \\ & p_L^{e_g o} \geq m_L^{e_g o} u^{e_g o}, \\ & p_L^{e_g o} \geq d_L^{e_g o} - M_L^{e_g o}(1 - u^{e_g o}). \end{aligned}$$

Note that, among all graph nodes and the origin and destination points, it is possible to identify the pair of points at the maximum distance. From this pair of points, we can build a circle whose diameter is the segment that joins them. Hence, because we minimise the distance travelled by the mothership, every launch or retrieval point is inside this circle, and the best upper bound $M_L^{e_g o}$ or $M_R^{e_g o}$ can be described as:

$$M_R^{e_g o} = \max_{\{v \in V_g \cup \{\text{orig,dest}\}, v' \in V_{g'} \cup \{\text{orig,dest}\} : g, g' \in \mathcal{G}\}} \|v - v'\| = M_L^{e_g o}.$$

On the other hand, the minimum distance in this case can be zero. This bound is attainable whenever the launch or the retrieval points of the mothership are the same as the retrieval or launching point on the target graph $g \in \mathcal{G}$.

Bounds on the bigM constants for the distance from the launch to the retrieval points on the target graph $g \in \mathcal{G}$

When the drone visits a graph g, it has to go from one edge e_g to another edge e'_g depending on the order given by $z^{e_g e'_g}$. This fact produces a product of variables linearised by the following constraints:

$$\begin{aligned} & p^{e_g e'_g} \leq M^{e_g e'_g} z^{e_g e'_g}, \\ & p^{e_g e'_g} \leq d^{e_g e'_g}, \end{aligned}$$

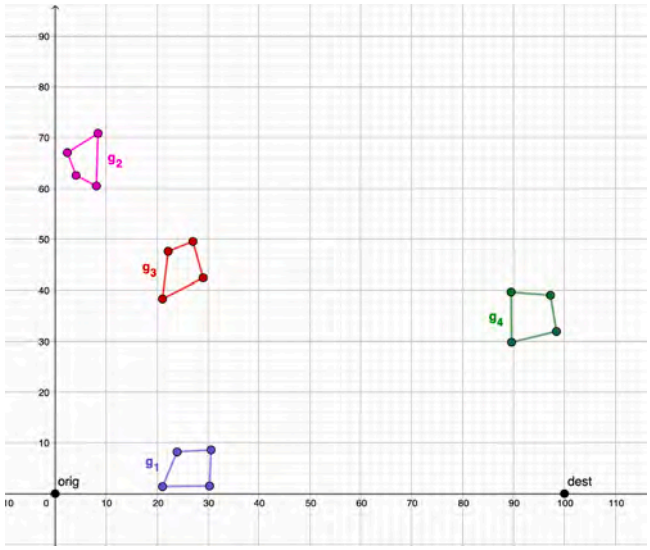


Fig. 6. Illustrative example.

$$p^{e_g e'_g} \geq m^{e_g e'_g} d^{e_g e'_g},$$
$$p^{e_g e'_g} \geq d^{e_g e'_g} - M^{e_g e'_g}(1 - z^{e_g e'_g}).$$

Since we take into account the distance between two edges $e_g = (B^{e_g}, C^{e_g})$, $e'_g = (B^{e'_g}, C^{e'_g}) \in E_g$, the maximum distance between their vertices gives us the upper bound:

$$M^{e_g e'_g} = \max\{\|B^{e_g} - C^{e'_g}\|, \|B^{e_g} - B^{e'_g}\|, \|C^{e_g} - B^{e'_g}\|, \|C^{e_g} - C^{e'_g}\|\}.$$

We observe that the minimum distance between edges $m^{e_g e'_g}$ can easily be obtained by computing the minimum distance between two edges, which results in a simple second-order cone program.

5. A matheuristic for the All Terrain Mothership and Multiple-Drone Routing Problem with Graphs

This section is devoted to presenting our matheuristic approach to address the solution of the AMMDRPG. Our motivation comes from the fact that the exact solution of the models presented in Section 3 is time-consuming. Alternatively, the matheuristic provides a good quality solution in limited computing times.

The basic idea of the algorithm is to determine the route that a drone should take to visit each graph $g \in \mathcal{G}$, and thus the entry and exit points L^{e_g} and $R^{e'_g}$ for each graph. Sequentially, a clustering procedure on the target graphs is applied to compute the route of the mothership via their reference points and the origin/destination points. The clustering procedure is based on a random selection of the initial target graphs, and, for this reason, it is repeated several times to consider different cluster structures. At each iteration, the new clusters are evaluated by computing the cost of the route that visits their reference points and the origin/destination points. The route of minimum length, computed on the reference points of the cluster generated by this iterative procedure, is used to set the values of the binary variables $u^{e_g o}$ and $v^{e_g o}$ which determine the order of visits to the graphs. Finally, these variables are provided as an initial partial solution to the AMMDRPG-CO model to produce a complete feasible solution.

Algorithm 1 reports the pseudocode of this algorithm.

Fig. 6 shows an illustrative example consisting of four target planar graphs (g_1, g_2, g_3 and g_4) to be visited. We assume that their visits must be carried out by a fleet of two drones supported by a mothership whose path starts from the origin $(0, 0)$ and ends at the destination point $(100, 0)$.

Algorithm 1 Matheuristic algorithm for AMMDRPG-CO

Data: $\mathcal{G}$, $|D|$, N_D, v_D, *maxit* (maximum number of iterations to perform the clustering procedure), *maxseed* (maximum number of the clustering procedure repetitions)

STEP 1 (First entry and last exit points for each target graph)

For each target graph $g \in \mathcal{G}$, compute the route:
$L^{e_g} \leftarrow$ entry point on g closest to the origin
$R^{e'_g} \leftarrow$ exit point from g closest to the origin
$\mathcal{L}(e_g, e'_g) \leftarrow$ route length

STEP 2 (Clustering procedure)

$it \leftarrow 1$
$nit \leftarrow 1$
For each target graph $g \in \mathcal{G}$: $K_g \leftarrow g$ ▷ one cluster for each target graph
while $nit < maxit$ **do**
  Select randomly two clusters K_i and K_j $(i < j)$
  if $|K_i \cup K_j| < |D|$ **then**
    Search for point P satisfying the following endurance constraint:

$$\frac{d(P, R^{e_g}) + \mathcal{L}(e_g, e'_g) + d(L^{e'_g}, P)}{v_D} \leq N_D, \quad \forall R^{e_g}, L^{e'_g} \in K_i, K_j. \quad (1)$$

    if $P \exists$ **then**
      $K_i \leftarrow K_i \cup K_j$
    end
  end
  $nit \leftarrow nit + 1$
end
$\mathcal{K} \leftarrow$ set of clusters

STEP 3 (Computation of Reference Points)

For each cluster $K_i \in \mathcal{K}$ compute a reference point P_i by solving the following minimisation problem:

$$\min \sum_{K_i \in \mathcal{K}} (\|P_i - orig\| + \|P_i - dest\|) + \sum_{g \in K_i : K_i \in \mathcal{K}} (\|P_i - R^{e_g}\| + \|P_i - L^{e'_g}\|) + \sum_{K_i, K_j \in \mathcal{K} : i \neq j} \|P_i - P_j\|$$

subject to (1).

STEP 4 (Order of visits to the graphs: route via the reference points and the origin/destination points)

Compute the TSP of the mothership among the reference points P_i of the clusters

$\mathcal{L}(TSP) \leftarrow$ TSP length [This update is performed only if $\mathcal{L}(TSP)$ decreases with respect to the previous iteration $it - 1$]
$it \leftarrow it + 1$
if $it < maxseed$ **then**
  go to STEP 2
else
  go to STEP 5
end

STEP 5 (Solution of the AMMDRPG model by fixing an initial partial solution)

Set the initial values of the binary variables $u^{e_g o}$ and $v^{e_g o}$ and solve the model AMMDRPG to obtain a feasible solution.

Result: Feasible solution for AMMDRPG-CO

Fig. 7 illustrates a zoom in on each target graph, showing the tours generated by STEP 1 of the matheuristic procedure. A pair of points representing the launch and retrieval points, together with an arrow

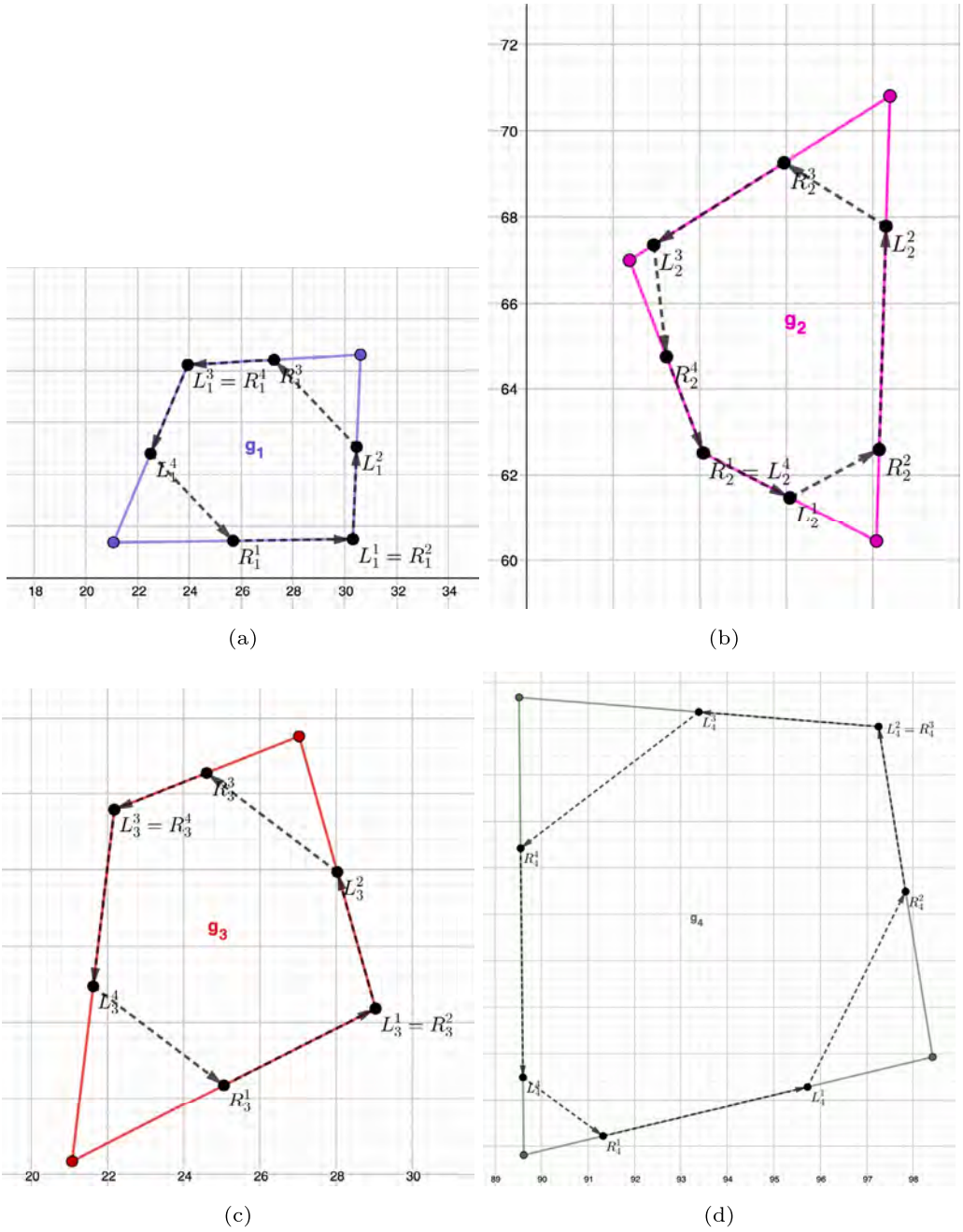


Fig. 7. STEP 1 for the illustrative example.

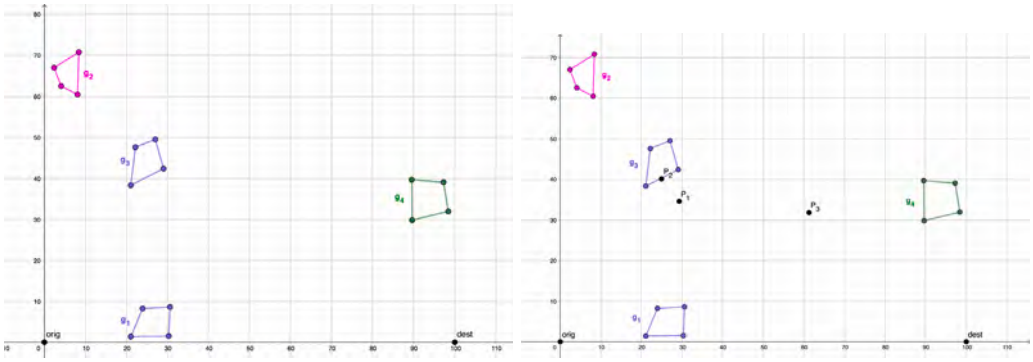
that points the direction followed by the drone according to the order in which the edges are visited, is depicted on each edge.

Applying STEP 2 to this illustrative example, we obtain three clusters, as shown in Fig. 8(a). One cluster contains graphs g_1 and g_3 (in lilac), while graphs g_2 and g_4 represent distinct clusters. The computation of the reference points of these clusters, according to STEP 3, produces the points P_1, P_2 and P_3, as shown in Fig. 8(b).

STEP 4 of the matheuristic procedure generates the tour of the mothership along the origin point, P_1, P_2, P_3 and the destination point, as shown in Fig. 9(a). This tour also returns the order in which the clusters are visited (and thus also the order of visits to the target graphs), and this allows us to set the values of the variables $u^{e_{g^o}}$ and $v^{e_{g^o}}$ of the AMMDRPG-CO model.

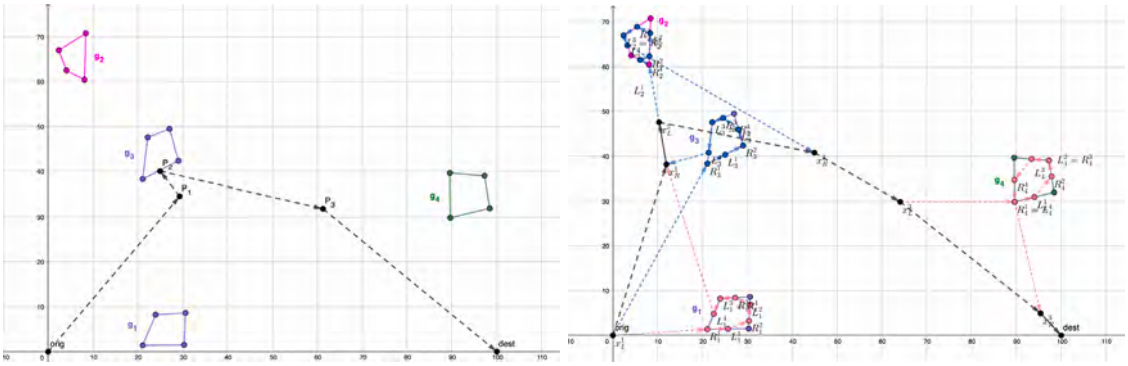
By providing the initial partial solution obtained from the values of the variables $u^{e_{g^o}}$ and $v^{e_{g^o}}$, STEP 5 solves the AMMDRPG-CO model and returns the final feasible solution shown in Fig. 9(b). From it, we can observe that the sequence of visits of the target graphs does not change with respect to that provided by STEP 4. The two-drone fleet first visits the graphs g_1 and g_3 starting from the launch point x_L^1. Then, both drones are retrieved by the mothership at point x_R^1. The mothership moves to the point x_L^2 where a drone is launched to visit graph g_2. The mothership then reaches the point x_R^2 to retrieve the drone, and from the same point it launches the other drone to visit graph g_4. Then, the mothership retrieves this drone at the point x_R^3 before moving to the final destination point.

Focussing on each target graph, Fig. 10 shows the zoom in on the tours followed by the drones. For example, Fig. 10[a] reports the



(a) (b)

Fig. 8. (a) STEP 2, (b) STEP 3 for the illustrative example.



(a) (b)

Fig. 9. (a) STEP 4, (b) STEP 5 for the illustrative example.

tour performed by the drone that visits the graph g_1 (the dashed pink segments) and the drone that visits the graph g_3 (the dashed light blue segments). Both drones start from the mothership at the point x_L^1, that is, the *orig*. One drone first visits the segment $\overline{R_1^1 L_1^1}$ of the graph g_1, while the other starts the visit to graph g_3 by traversing the segment $\overline{R_1^3 L_1^3}$. From point L_1^1 the first drone moves to the second visited edge of the graph g_1 traversing the segment $\overline{R_1^2 L_1^2}$. Then, it moves to the third visited edge of the graph g_1, flying over the segment $\overline{R_1^3 L_1^3}$. From point R_1^4 the drone starts the visit to the last edge of the graph g_1 up to point L_1^4. Finally, the drone leaves the graph g_1 at this last point and is retrieved by the mothership at the point x_R^1. Similarly, the second drone, which visits the graph g_3, after traversing the segment $\overline{R_3^1 L_3^1}$, moves to the second visited edge of the same graph and traverses the segment $\overline{R_3^2 L_3^2}$. Then, it flies to the third visited edge, traversing the segment $\overline{R_3^3 L_3^3}$. Finally, it moves to the last visited edge of the graph g_3, flying over the segment $\overline{R_3^4 L_3^4}$. The drone leaves the graph g_3 at the point L_3^4 and reaches the mothership at the point x_R^1. Note that, in this example, the drones do not visit the full 100% of each graph edge, but only half of each of them.

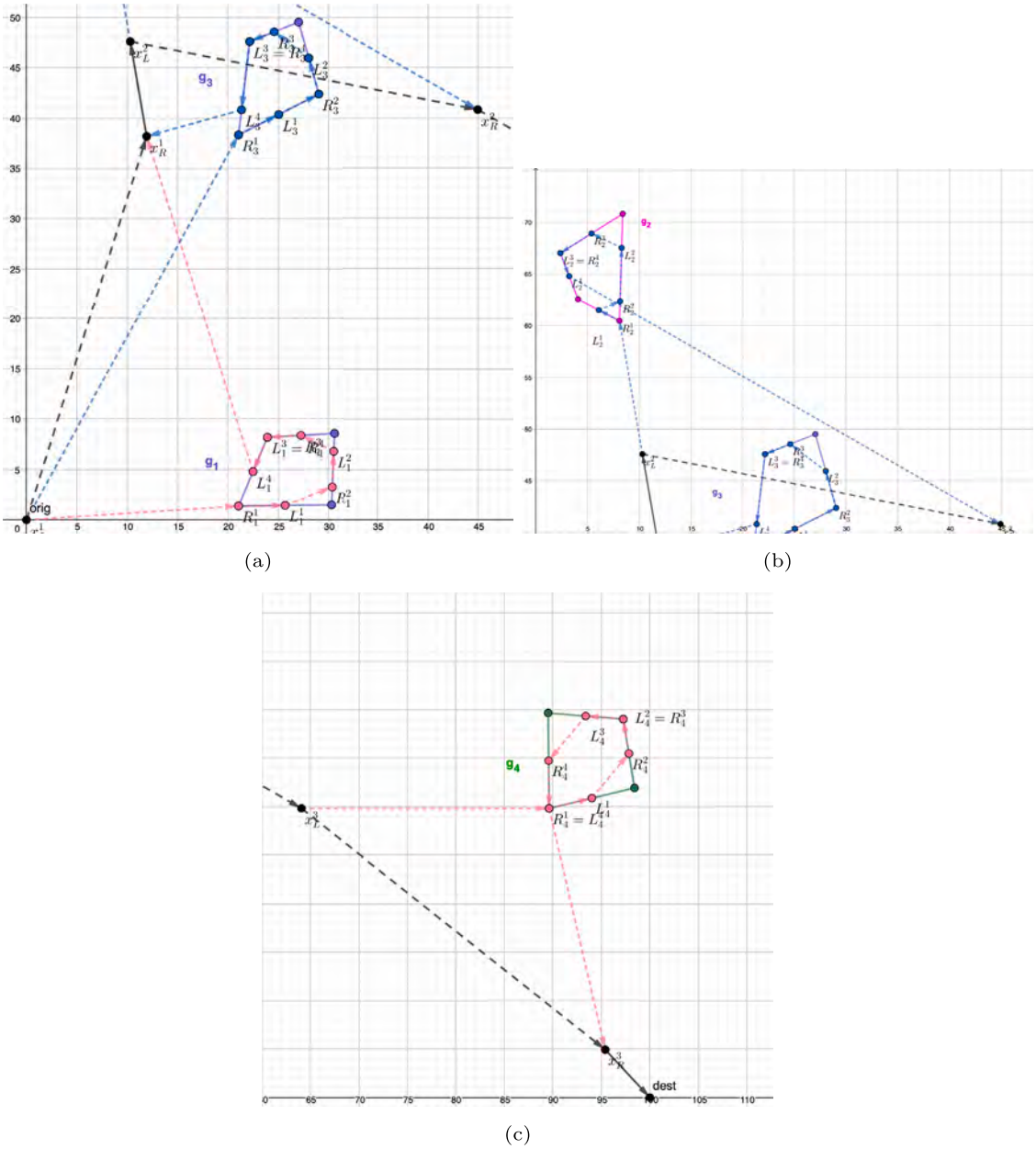
The reader may notice that the algorithm above can also be used to generate solutions for the partial overlapping model presented in Section 3.2 as any solution of the model AMMDRPG-CO is also feasible for the model AMMDRPG-PO one, as shown in Theorem 3.1.

6. Experimental results

In this section, we discuss the experimental results obtained testing the formulations presented in Section 3 and the matheuristic procedure proposed in Section 5 on a testbed of instances. In particular, we consider instances such as those used in Amorosi et al. (2021), where the targets to be visited are represented by grid graphs. More precisely, we generate a set of five instances each with five graphs having 4, 6, 8, 10, and 12 nodes, respectively. Similarly, we generate a set of five instances each with ten graphs, equally distributed with respect to the number of nodes (which always ranges between 4 and 12).

In order to place the graphs of a single instance, we follow the same procedure described in Amorosi et al. (2021). This is based on the division of an initial square of side 100 units in sub-squares and on the random selection of the graph locations among them. Then, in order to build the single graph, each of the randomly selected sub-squares is further partitioned in a number of sub-squares equal to the cardinality n of the set of nodes of the graph to build. Successively, two opposite corner sub-squares are considered to randomly select two points and build a rectangle whose diagonal joins these two points. Finally, a grid of n points is identified by locating $\frac{n}{m}$ equally spaced points on the two sides square, where m is randomly selected in the set of divisors of n. A perturbation on the coordinates of the points so obtained is applied, imposing that the perturbed points still belong to the original square. The resulting grid graph is obtained connecting each point to its adjacent ones lying on the same side and with the one located on the opposite side of the square. The reader can find and download all the instances used in this paper from Puerto and Valverde (2021). Moreover, we assume that the speed of the drones is twice that of the mothership and that the fleet of drones must visit a random fraction of each target graph or of each of its edges. These fractions are uniformly randomly sampled in the interval $(0, 1)$.

In our experiments, we consider that the number of drones varies between 1 and 3 and that the endurance of the drone (expressed as the maximum time the drone can operate when fully recharged) ranges between 20 and 60. Note that the case of a single drone is also included



(a) (b) (c)

Fig. 10. Zoom on the tour on each target graph provided by STEP 5.

Table 4
Instance parameter values.

$\|\mathcal{G}\|$	(5,10)
$\|D\|$	(1,2,3)
$\|V_g\|$	(4,6,8,10,12)
N_D	(20,30,40,50,60)
Fraction target (edge)	Uniformly randomly sampled in $(0, 1)$.

in our experiments to compare the results and complexity of using one or more than one drone. The interested reader is referred to Amorosi et al. (2021) to analyse the complexity in terms of the gap of the model with a single drone. Table 4 reports a summary of the characteristics of our instances.

We code the matheuristic and the exact resolution of the model in Python 3.8.10. The mathematical programming formulation is implemented in Gurobi 9.1.2. All tests are run on an AMD® Epyc 7402p with 24-core processor × 8. Table 5 reports the results obtained by solving both variants of the AMMDRPG model on instances described above, by adopting Gurobi commercial solver. We consider the exact solution, setting a time limit of one hour, providing and not providing to the solver an initial solution computed by the matheuristic described in Section 5. More precisely, the first row of Table 5 indicates the variant of the model, and the second row reports the number of target graphs to be visited by the drone fleet (5 or 10). From the third row, we split each column into three sub-columns. The first three subcolumns report, respectively, the endurance of the drones, the size of the fleet of drones, and an indication of the visit of a fraction of each edge (e) or a fraction of each target graph (g). From subcolumn 4 to subcolumn 15, we report, for each combination of the listed parameters characterising the instances, respectively, the average gap of the best solution found by Gurobi in one hour without initialisation by the solution provided by the matheuristic (wi), the average gap of the best solution found by Gurobi in one hour with initialisation by the solution obtained by the matheuristic (i), and the solution time, in seconds, of the matheuristic (TimeH).

Table 5
Comparison between partial and complete overlapping models.

Model			Complete overlapping						Partial overlapping					
$\|\mathcal{G}\|$			5			10			5			10		
N_D	$\|D\|$	α	Gap (wi)	Gap (i)	TimeH	Gap (wi)	Gap (i)	TimeH	Gap (wi)	Gap (i)	TimeH	Gap (wi)	Gap (i)	TimeH
20	1	g	0.78	0.79	6.01	0.91 (2)	0.86	177.69	0.65	0.63	16.53	1	1	215.59
		e	0.81	0.81	15.41	0.89 (2)	0.84	148.95	0.84	0.83	52.36	0.88 (3)	0.87	440.93
	2	g	0.81	0.87	5.76	0.96 (3)	0.96	139.24	0.97	0.96	13.91	1 (3)	1	76.77
		e	0.93	0.92	33.99	0.97 (3)	0.97	163.41	0.86 (2)	0.85	66.38	0.89 (4)	0.85	578.31
	3	g	0.88	0.89	4.83	0.95 (3)	0.94	67.76	0.97	0.97	17.87	1 (2)	1	18.88
		e	0.92	0.91	14.08	0.97 (2)	0.97	125.89	0.81 (3)	0.84	61.83	– (5)	0.82	237.33
30	1	g	0.71	0.7	9.66	0.82 (4)	0.82	87.4	0.77	0.75	15.43	1	1	39.83
		e	0.79	0.8	14.16	0.8 (4)	0.83	122.23	0.84	0.82	38.94	0.83 (4)	0.81	289.74
	2	g	0.82	0.82	4.98	0.95 (3)	0.92	174.64	0.97	0.96	12.94	1	1	45.37
		e	0.84	0.84	14.73	0.96 (3)	0.97	133.75	0.78	0.79	31.82	0.82	0.77	171.16
	3	g	0.82	0.81	4.63	0.93 (3)	0.95	105.54	0.96	0.96	16.22	1	1	33.95
		e	0.88	0.89	12.08	– (5)	0.97	127.78	0.83	0.82	35.38	0.79 (3)	0.8	213.06
40	1	g	0.68	0.68	5.79	0.81 (2)	0.82	93.21	0.73	0.71	11.46	1	1	48.85
		e	0.76	0.77	37.55	0.78 (4)	0.81	160.24	0.8	0.79	57.28	0.79 (1)	0.8	403.72
	2	g	0.72	0.66	5.14	0.91 (2)	0.92	131.26	0.96	0.95	11.48	1	1	35.71
		e	0.83	0.78	19.46	0.91 (2)	0.95	141.6	0.79	0.79	35.79	0.79 (1)	0.79	576.75
	3	g	0.61	0.62	3.91	0.91	0.91	115.48	0.95	0.95	15.13	1	1	17.98
		e	0.85	0.83	15.36	0.93	0.94	85.9	0.81	0.81	40.37	0.81 (1)	0.8	309.09
50	1	g	0.65	0.64	5.52	0.82 (3)	0.84	101.24	0.82	0.78	9.53	1	1	32.54
		e	0.74	0.73	16.63	0.81 (3)	0.83	118.67	0.78	0.77	58.95	0.82 (2)	0.82	311.02
	2	g	0.7	0.7	6.37	0.9 (1)	0.93	206.87	0.97	0.97	14.68	1	1	39.5
		e	0.67	0.73	12.07	0.92 (2)	0.93	168.57	0.77	0.77	36.46	0.8 (1)	0.81	265.16
	3	g	0.65	0.64	4.27	0.9 (1)	0.93	26.68	0.94	0.92	19.08	1	1	15.97
		e	0.74	0.74	12.95	0.9	0.94	90.14	0.8	0.79	40.77	0.76 (3)	0.79	195.68
60	1	g	0.69	0.7	5.58	0.8 (4)	0.81	83.02	0.78	0.76	11.18	1	1	36.78
		e	0.74	0.74	16.53	0.85 (2)	0.86	145.06	0.76	0.76	37.73	0.84 (2)	0.83	359.68
	2	g	0.67	0.72	4.09	0.94 (2)	0.94	81.69	0.95	0.94	13.33	1	1	17.04
		e	0.76	0.73	15.58	0.94 (2)	0.92	108.17	0.78	0.78	33.28	0.78	0.79	237.38
	3	g	0.58	0.53	7	0.89 (2)	0.9	60.99	0.91	0.94	20.15	1	1	33.93
		e	0.72	0.7	15.39	0.91 (2)	0.96	96.52	0.78	0.78	49.39	0.81	0.81	259.34

We can observe that the value of the average gap ranges between a minimum of 0.58 and a maximum of 1. This shows that the model is difficult to solve even with small-sized instances. Moreover, we can see that for the complete overlapping version of the model, in most cases, the average gap associated with the variant of the model consisting of visiting a given fraction of each edge is greater than that associated with the variant obligating visiting a given fraction of each target graph. Another thing we can observe is that the average gap increases with the number of target graphs for both variants of the problem.

Furthermore, the reader may note that the partial overlapping version of the problem is harder to solve than the complete overlapping version by looking at the values of the average gap. This is an expected behaviour due to the fact that the feasible region of the partial overlapping variant contains that associated with the complete overlapping variant, as proven in Theorem 3.1. We can see that for both versions of the problem, by increasing the number of target graphs from 5 to 10, the exact method without initialisation of the solution obtained with the matheuristic becomes even harder. Indeed, the red entries of the table mean that some instances could not find a feasible solution within the time limit (note that in the brackets we indicate the number of these instances). Furthermore, for the minimum level of endurance, the exact solution of the partial overlapping model without initialisation provided by the matheuristic does not find any solution within the time limit for instances with 10 graphs, 3 drones and a given fraction of each edge to be visited. The same can be observed also for the exact solution of the complete overlapping model without initialisation provided by the matheuristic, for a level of endurance equal to 30, a fleet of 3 drones, and a given fraction of each edge to be visited. Taking into account the comparison with the exact method starting from the solution provided by the matheuristic, we can note that the values of the average gap are very close to those related to the exact solution method without initialisation. Thus, initialisation does not speed up the convergence of the solver. However, we can see that the matheuristic is always able to find a feasible solution to the problem, even for the cases where the solver is not.

Furthermore, the average solution times of the matheuristic range between a minimum of 4 s and a maximum of 9.5 min. In particular, we can observe that, in most cases, for the complete overlapping version of the problem, the matheuristic running time is shorter for the instances where a fraction of the length of each graph is required to be visited. The same behaviour can be observed for the partial overlapping version of the problem, for which the difference in terms of running time is even greater. Indeed, when a given fraction of the length of each edge is required, STEP 1 of the matheuristic (computation of the TSP over the graph edges) takes more time. By increasing the number of target graphs from 5 to 10, the average solution times of the matheuristic increase for both model variants. Summing up, the results obtained show that the exact solution method given by solving the formulation is very challenging even for small-sized instances. However, by exploiting this, the matheuristic is able to provide solutions for all instances quite quickly. Moreover, the quality of the solutions found by the matheuristic algorithm in a few seconds (or minutes) is comparable with the one of the solutions provided by Gurobi solver in one hour. Indeed, there are no significant differences between the average gaps associated with the solutions obtained with and without initialisation.

The boxplots in Fig. 11 represent the percentage relative gap of the solution provided by the matheuristic for the complete overlapping version of the problem, with respect to that provided by the exact solution

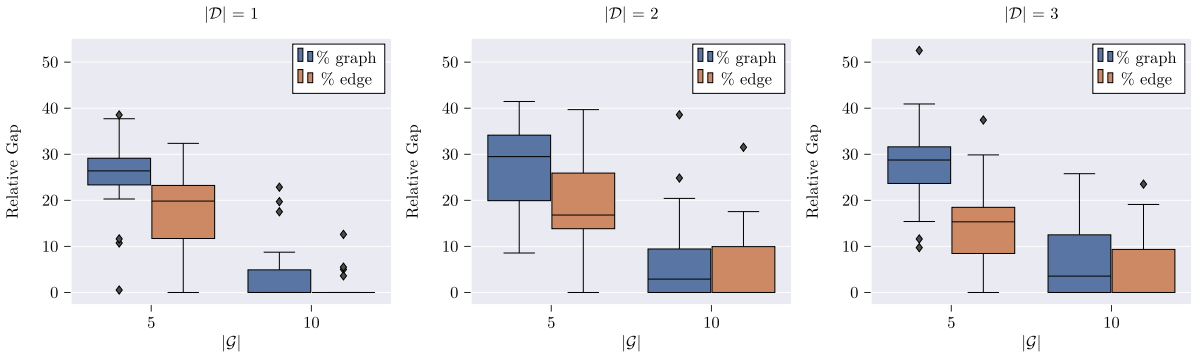


Fig. 11. Relative gap boxplots for AMMDRPG-CO.

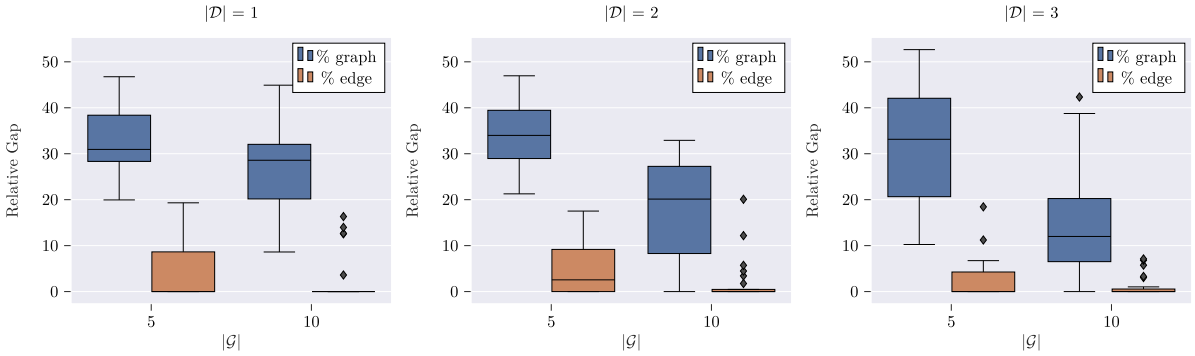


Fig. 12. Relative gap boxplots for AMMDRPG-PO.

of the mathematical programming model within the time limit, with the initialisation of the solution found by the matheuristic. Similarly, Fig. 12 reports the same information for the partial overlapping version of the problem.

From Fig. 11, we can see that for the complete overlapping version of the problem, the relative gap of the solution provided by the matheuristic tends to be greater when a given fraction of each graph must be visited, independently of the size of the drone fleet. Additionally, its values decrease with the number of target graphs. A similar behaviour can also be observed for the partial overlapping variant of the problem, from Fig. 12. In the latter case, we can notice a larger difference between the relative gap values related to the case in which a given fraction of each graph must be visited, and that in which a given fraction of each edge must be visited. Indeed, in the first case, the relative gap ranges between 0 and 50, while in the second case, between 0 and 20. Thus, we can conclude that the matheuristic provides very good quality solutions in a short computing times, especially for the version of the problem in which a given fraction of each edge must be visited.

For illustrative purposes, Fig. 13 shows one of the instances with five graphs and two drones adopted for the experimental results. Fig. 14 reports the solution of the partial overlapping version of the problem for this instance and a zoom on the mothership tour. We can observe that the origin (the black square) is the first launching point for the first drone (whose path is represented with green dotted lines) to visit the graph g_3. Then, the mothership moves to point x_L^2 for launching the second drone (whose path is represented with red dotted lines) that visits the graph g_5. Successively, the mothership moves to point x_R^3 for retrieving the same drone. From point x_L^4 the mothership launches again the second drone to visit the graph g_2. We can notice that this latter launching point coincides also with the entering point in the first visited edge of the graph g_2. From point x_R^5 the mothership retrieves the first drone that is launched again from point x_L^6 to visit the graph g_4. From the same point the mothership also retrieves the second drone ($x_L^6 = x_R^7$). Then, it moves to point x_L^8 for launching again the second drone for visiting the graph g_1. From point x_R^9 the mothership retrieves the first drone and, eventually, from point x_R^{10} it retrieves also the second drone. We can notice that the retrieving point x_R^{10} coincides with the exiting point for the drone at the end of its visit of the graph g_3. The mothership tours ends at the destination, that for this example coincides with the origin.

In Fig. 15 we show also the zoom on the tour on each target graph. The blue point represents the entering point in the graph and the yellow one represents the exiting point from the graph. The dotted arrows shows the path followed by the drones to visit each target graph.

Table 6
Instance parameter values.

$\|\mathcal{D}\|$	(1,2,3)
$\|V_g\|$	(4,6,8)
N^d	(10,20,30,40,50,60)
Fraction target (edge)	Uniformly randomly sampled in $(0,1)$.

6.1. Comparing the solutions for different configurations of the problem

In this subsection, we compare the relationship between the number of drones available and their endurance and the value of the objective function obtained with the exact algorithm of the problem. In this experiment, we have generated a single instance with three target graphs for each combination of the parameters listed in Table 6.

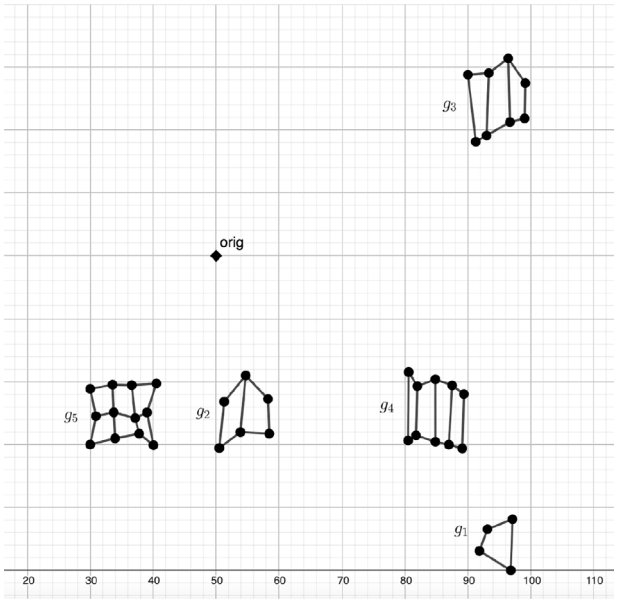


Fig. 13. Example of instance with 5 graphs.

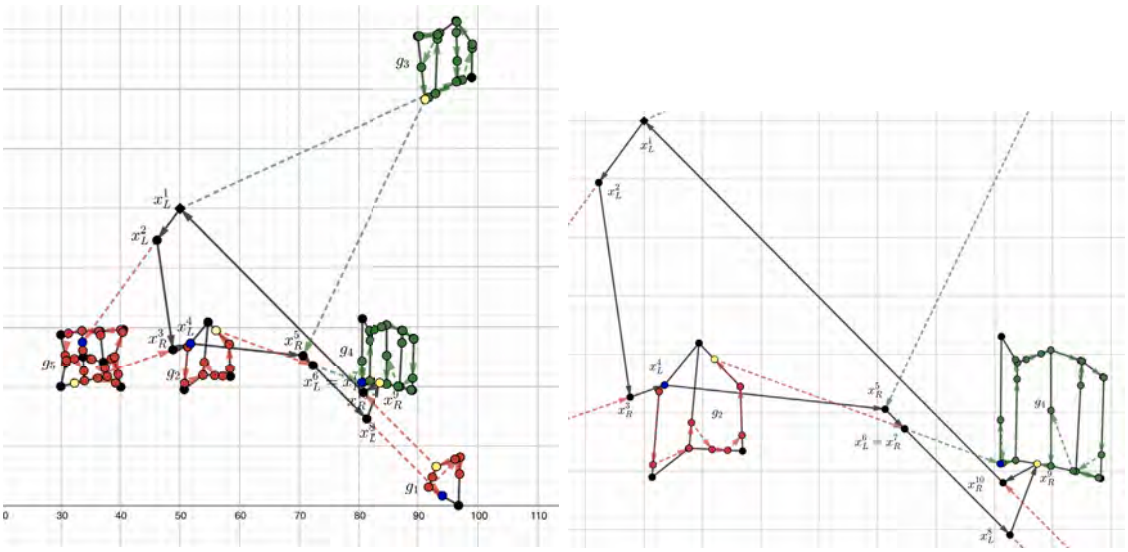


Fig. 14. Solution for the instance of Fig. 13 and zoom on the mothership tour.

Fig. 16 reports the objective value that depends on these parameters. The darker the intensity of the colour, the lower the objective value. As expected, our experiment confirms that both the larger number of drones and the greater endurance reduce the makespan of the mothership route.

7. Case study

In this section, we describe a realistic application of the system studied in this paper to perform surveillance operations. Considering the experienced COVID-19 restrictions, we focus on the problem of preventing and identifying possible concentrations of people during events such as popular or religious festivals. In particular, we consider the Cordoba Courtyards Festival (https://patios.cordoba.es/es/). This is a social event that takes place every year in the city of Cordoba, Spain, during the first two weeks of May. The owners of the courts decorate their houses with many flowers trying to win the award offered by the Municipality. During this competition, a festival is run in parallel with several artistic performances along six different paths located in different areas of the city, as shown in Fig. 17. In context of the pandemic, to monitor the situation to avoid concentration of people, we propose applying a system consisting of a helicopter and a fleet of two drones. This kind of system has been tested successfully and has already been applied in the military field by the US Army to leave the helicopter at the edge of dangerous airspace and release drones, which will then penetrate enemy territory and send back intelligence, surveillance, and reconnaissance information (see Reim (2020)). In our application, the reason for adopting a similar system is the possibility of simultaneously inspecting different paths in real-time, also reducing the risk of flying the helicopter over populated areas and the cost of moving the helicopter by minimising the total length of its tour.

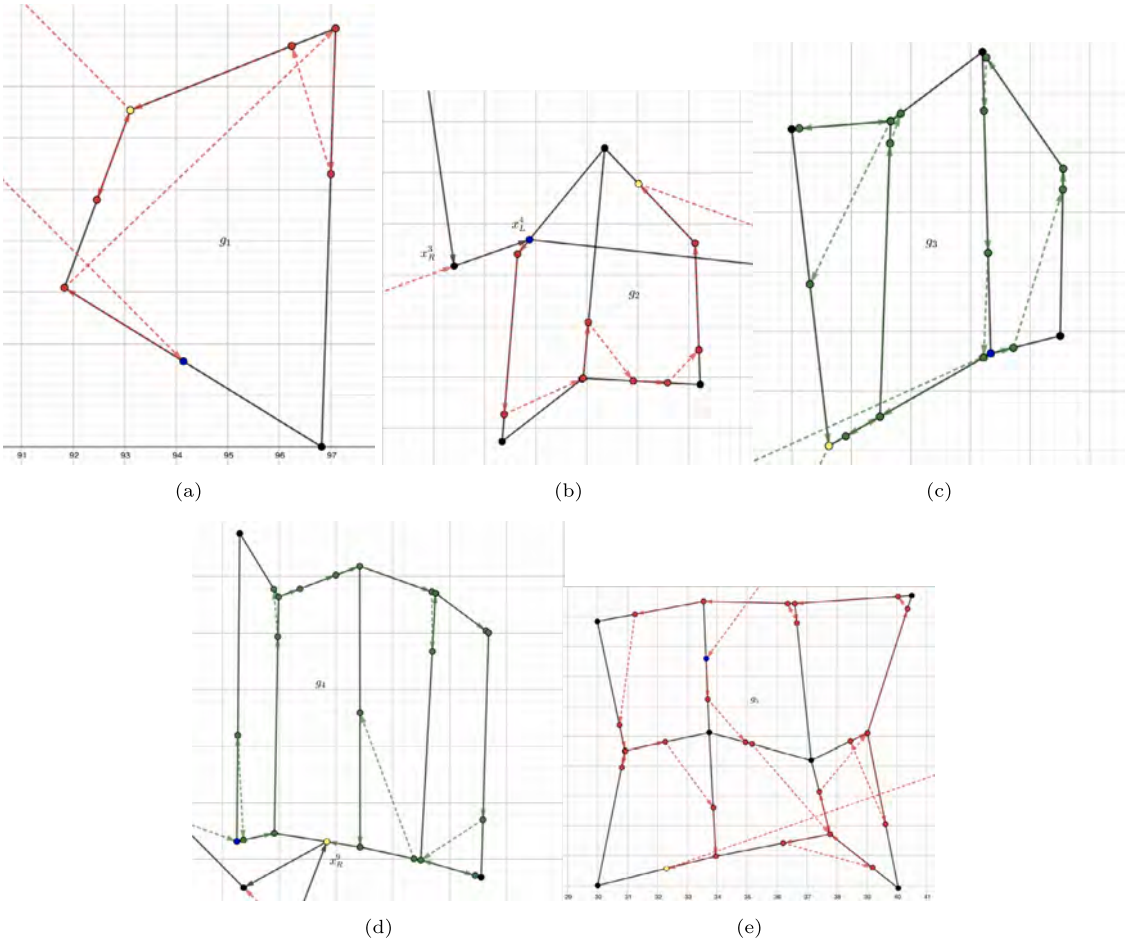


Fig. 15. Zoom on the tour on each target graphs.

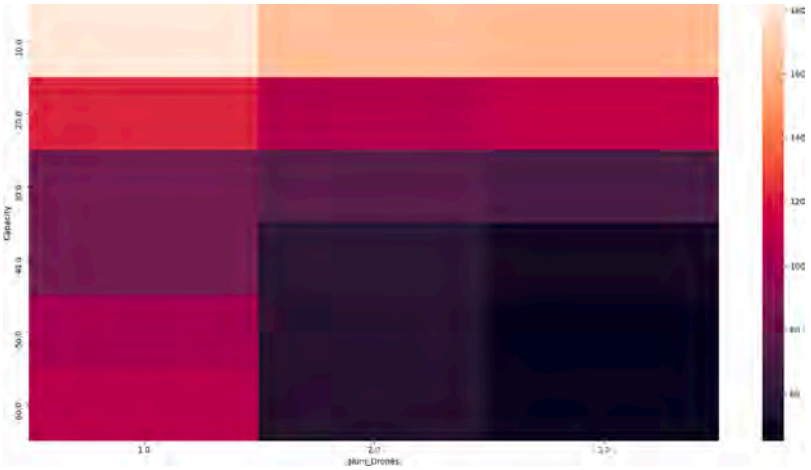


Fig. 16. Heatmap of the values of the objective function depending on the the number of drones and drone endurances. The darker the colour intensity, the lower the objective value.

We run the models presented in Section 3 in this scenario starting from the initial solution provided by the matheuristic, where the 6 coloured paths reported on the map of Fig. 17 represent the 6 target graphs to be visited, in this case, inspected, by the drone fleet. In addition, we assume that the drone speed is 43 km/h, while that of the helicopter is 30 km/h with the aim to minimise costs. Furthermore, we assume that the fleet consists of two drones with endurance equal to 7.5 min, and we impose that each target graph must be fully visited (inspected). As we can see in Figs. 18 and 19, the origin of the mothership tour coincides with the destination and is located in an area of the city where it is possible to take off and land the helicopter. Fig. 18 reports the tour followed by the helicopter in solving the complete overlapping version of the problem, after 4 h of running time. We can observe that the helicopter, starting from the origin, flies to the point x_R^1 which is the first retrieval point, coinciding with the second launching point x_L^2. Then it flies along the edge connecting x_R^1 with x_R^2, that is, the

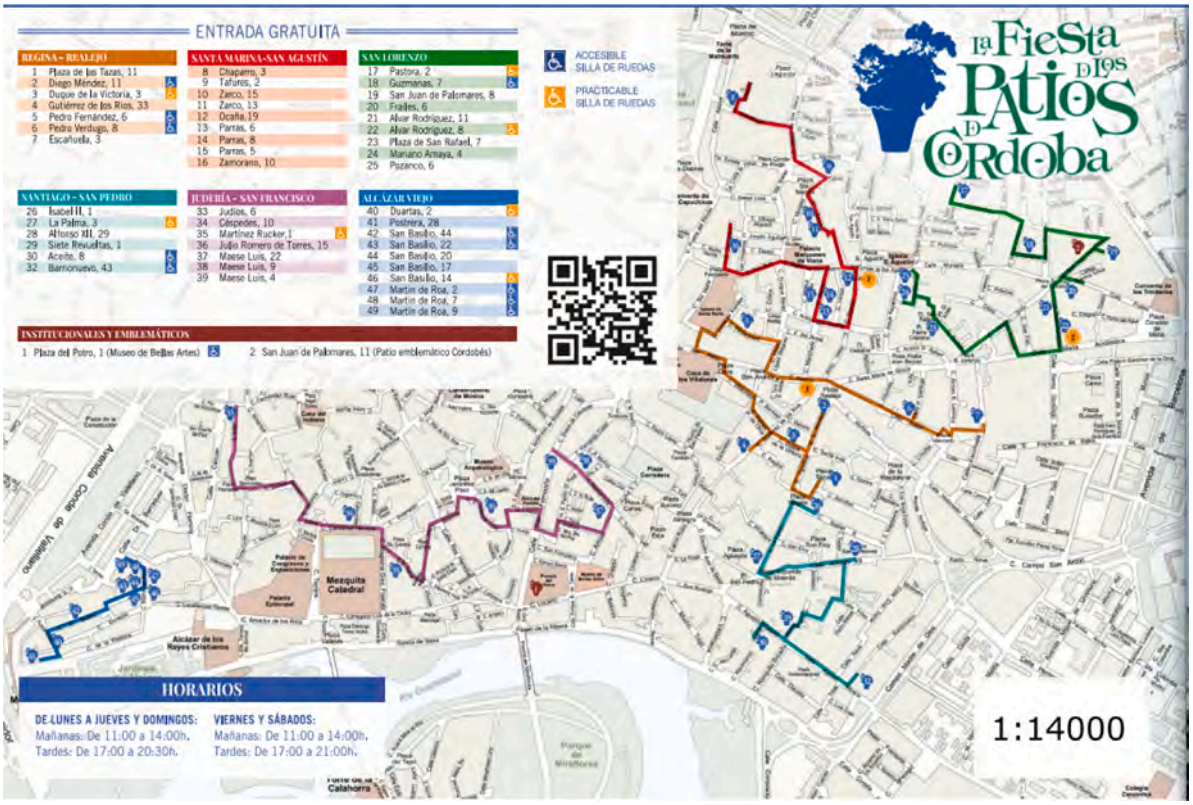


Fig. 17. Map of the Courtyards Festival in Cordoba.

second retrieval point, which coincides with the third launching point x^3_L. The helicopter then flies to x^3_R to retrieve the drone completing the third mission. From the same point, the fourth and last mission starts and ends at the point x^4_R, which is also the final destination of the helicopter tour.

Fig. 18 also shows the tour (the violet and the red dotted paths) followed by the two drones to inspect the six paths. In particular, one drone starts from the origin ($orig = x^1_L$) to visit the path of "Alcazar Viejo". It is retrieved by the helicopter at the point x^1_R and from the same point both drones are launched to visit, respectively, the paths of "Juderia-San Francisco" and "Santa Maria-San Agustin". Both drones end their mission at the point x^2_R. From this last point, they are launched to perform the visits to the paths of "San Lorenzo" and "Regina-Realejo". Then, both drones are retrieved by the helicopter at the point x^3_R where only one drone starts its last mission to visit the path of "Santiago-San Pedro". Meanwhile, the helicopter, which contains the other drone, flies to the point $x^4_R = dest$ where it retrieves the other and ends its tour. Total time taken by helicopter is approximately 21 min. We can observe that in the drone tour on the graphs "San Lorenzo" and "Regina-Realejo", there are two edges whose duplicate is represented by a dotted segment in Fig. 18. They are associated with edges of the graph that are visited once, but travelled twice by the drone, in order to perform the inspection of the entire graph. Fig. 19 shows the solution of the partial overlapping version of the problem, always obtained by setting a time limit of 4 h. In this case, we can observe that the helicopter follows a different tour and that there are more launch and retrieval points due to the possibility of launching the drone before retrieving the other. From Fig. 19 we can also see that, unlike the complete overlapping version, both drones start their first mission from the origin $orig = x^1_L = x^2_L$. One drone visits the path of "Alcazar Viejo", while the other visits the path of "Santiago-San Pedro". The first is retrieved by the helicopter at the point x^3_R and is launched again from the point x^4_L. From this latter point, this drone starts its second mission to visit the path of "Juderia-San Francisco". Meanwhile, the helicopter flies to the point x^5_R where the other drone is retrieved. From the same point $x^5_R = x^6_L$ this latter drone is then launched to inspect the path of "Santa Maria-San Agustin". Both drones are retrieved by the helicopter at the point $x^7_R = x^8_R$. From this latter point $x^8_R = x^9_L$ one drone is launched to visit the path of "Regina-Realejo". Then, the helicopter flies to the point x^{10}_L from where the other drone starts its last visit to the path of "San Lorenzo". Finally, the helicopter flies to the destination $dest$ and along its path, it retrieves the first drone at the point x^{11}_R and then the other at the point x^{12}_R. In this case, as in the solution of the complete overlapping version of the problem, we have one edge of the graph associated with the path of "Regina-Realejo" and one edge of the graph representing the path of "San Lorenzo", which are traversed twice represented with dotted segments in Fig. 19. The total travel time of the helicopter is 19 min. It is slightly lower than that associated with the solution of the complete overlapping version of the problem. Therefore, even if in this scenario we cannot observe significant changes in terms of the objective function value, we can see how the different assumptions associated with the two versions of the problem can influence the structure of the solution, by producing a different schedule of drone missions and a different location of the launch and retrieval points.

All details of this case study, including map coordinates, .lp models and solutions can be found in Puerto and Valverde (2021).

8. Concluding remarks

This paper has analysed the coordination problem that arises between a mothership vehicle and a fleet of drones that must coordinate their routes to minimise the makespan while visiting a set of targets modelled by graphs. We have presented exact mixed-integer non-linear programming formulations of the problem for its complete and partial overlapping versions. Furthermore, we strengthen the models with some valid inequalities for them.

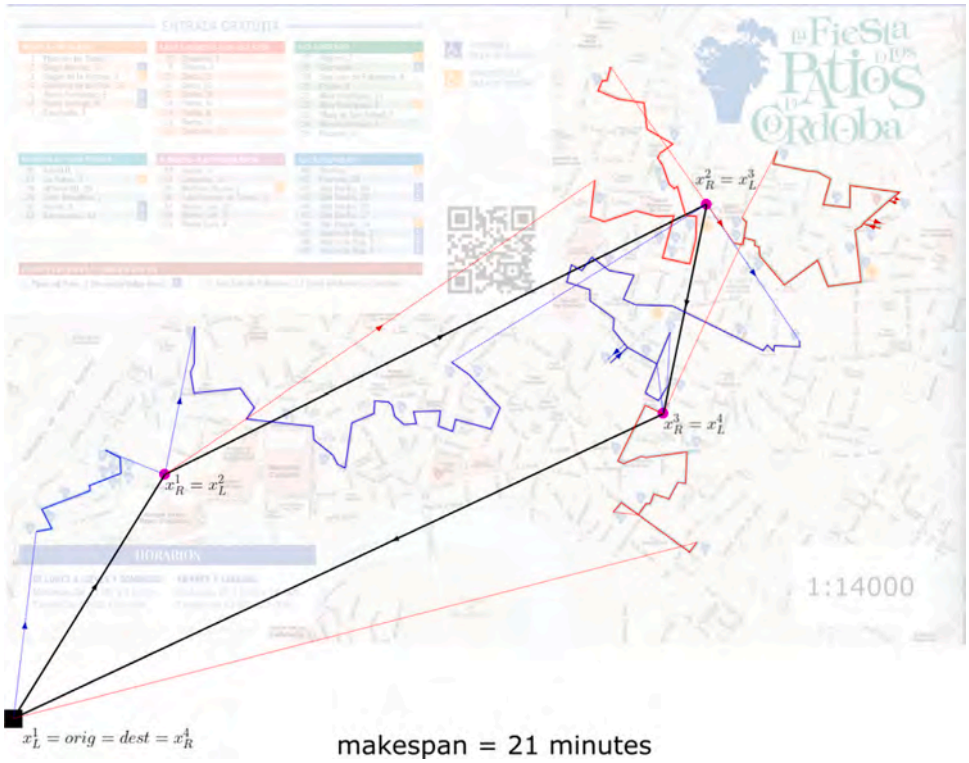


Fig. 18. The complete overlapping solution (CO).

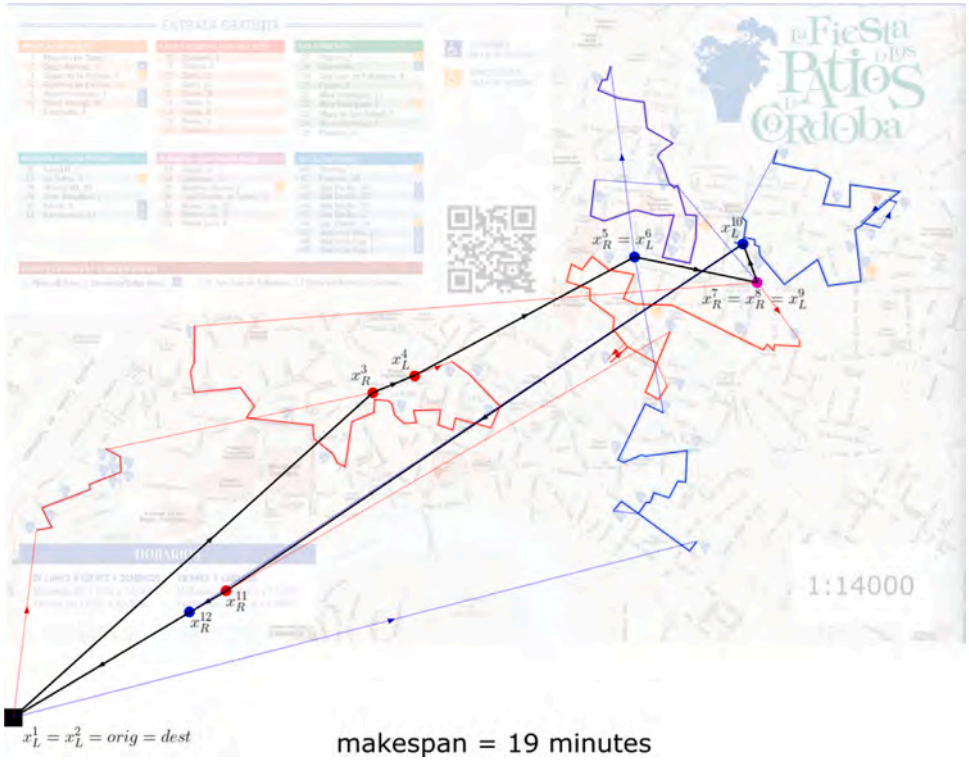


Fig. 19. The partial overlapping solution (PO).

Our computational results show that problem considered is very challenging to solve even in small and medium-sized instances. For that reason, additionally, we have proposed a matheuristic algorithm that provides good quality feasible solutions in a short computing

time; so that it is a good alternative to the exact method. We report extensive computational experiments on randomly generated instances. Furthermore, we present a case study related to inspection activities in the context of COVID-19 restrictions. We show the application of the system described in this article in the framework of the Courtyards Festival in the city of Cordoba, illustrating the solution obtained by adopting the formulation of the problem, in both versions of the model, and its solution by means of the initialisation provided by the proposed matheuristic.

The formulation and algorithms proposed in this paper can be seen as the first building block for coordination systems composed of a base vehicle and several drones. Further research on this topic must focus on finding faster and more accurate algorithms capable of solving larger instances. Other extensions that may be considered can take into account that the time the mothership takes to launch and retrieve the drones is not negligible, as well as handle the speed of the mothership and the drones as decision variables. Eventually, it is also possible to consider different shape of the graphs edges, like curve lines instead of straight lines. These problems are very interesting and beyond the scope of this paper and will be the focus of a follow-up research line.

CRediT authorship contribution statement

Lavinia Amorosi: Conceptualization, Methodology, Formal analysis, Writing – original draft, Writing – review & editing, Visualization. **Justo Puerto:** Supervision, Conceptualization, Methodology, Formal analysis, Writing – original draft, Writing – review & editing. **Carlos Valverde:** Conceptualization, Methodology, Formal analysis, Software, Writing – original draft, Writing – review & editing, Visualization.

Data availability

Data will be made available on request.

Acknowledgements

This research has been partially supported by the Spanish Ministry of Education and Science/FEDER grant number PID2020-114594GB02, projects Junta de Andalucia P18-FR-1422, FEDER-US-1256951, CEI-3-FQM331, *NetmeetData*: Ayudas Fundación BBVA a equipos de investigación científica 2019, University of Rome, Sapienza grant number RM11916B7F962975 and Ministerio de Universidades, grant number FPU2018-04591.

Appendix

In this section, we report an extension of the MINLP formulation presented in Section 3 to deal with the case of non-homogeneous fleets of drones. We introduce the parameters or input data that formally describe the problem and are summarised in Table 7.

The formulation in this appendix is quite similar to the one in Section 3.1 but, due to the assumption of non-homogeneous drones, it needs to keep track of the drones used in each action. This implies including an extra index δ in most variables. For the sake of completeness, we include the complete set of constraints of these formulations, although some of them are similar to those in Section 3.1. Table 8 summarises all the decision variables used in this formulation.

Visits to graphs

Regarding the case of a homogeneous fleet of drones presented in Section 3, to represent the movement of the drone within a graph $g \in \mathcal{G}$, we proceed to introduce some notations related to g. Let $g = (V_g, E_g)$ be a graph in $\mathcal{G}$ whose total length is denoted by $\mathcal{L}(g)$. Here, V_g denotes the set of nodes and E_g denotes the set of edges connecting pairs of nodes. Let e_g be the edge e of the graph $g \in G$ and let $\mathcal{L}(e_g)$ be its length.

Table 7
Nomenclature for AMMDRPG with a non-homogeneous fleet of drones.

Problem parameters
$orig$: coordinates of the point defining the origin of the mothership path (or tour).
$dest$: coordinates of the point defining the destination of the mothership path (or tour).
$\mathcal{G}$: set of the target graphs.
$g = (V_g, E_g)$: set of nodes and edges of each target graph $g \in \mathcal{G}$.
$\mathcal{L}(e_g)$: length of edge e of graph $g \in \mathcal{G}$.
$\mathcal{L}(g) = \sum_{e_g \in E_g} \mathcal{L}(e_g)$: total length of the graph $g \in \mathcal{G}$.
B^{e_g}, C^{e_g}: coordinates of the endpoints of edge e of graph $g \in \mathcal{G}$.
α^{e_g}: fraction of edge e of graph $g \in \mathcal{G}$ that must be visited.
α^g: fraction of graph $g \in \mathcal{G}$ that must be visited.
v_M: mothership speed.
$\mathcal{D}$: set of drones.
v_δ: drone δ speed.
N_δ: drone δ endurance.
$\mathcal{O}$: set of drone operations to perform visits to the target graphs
M: big-M constant.

Each edge e_g is parameterised by its endpoints $B^{e_g} = (B^{e_g}(x_1), B^{e_g}(x_2))$ and $C^{e_g} = (C^{e_g}(x_1), C^{e_g}(x_2))$ and we can compute its length $\mathcal{L}(e_g) = \|C^{e_g} - B^{e_g}\|$.

For each edge e_g an indicator binary variable μ^{e_g} is associated that is, one if the drone visits the segment e_g. Furthermore, we define the entry and exit points $R^{e_g} = (B^{e_g}, C^{e_g}, \rho^{e_g})$ and $L^{e_g} = (B^{e_g}, C^{e_g}, \lambda^{e_g})$ that determine the fraction of the edge visited by the drone. The coordinates of the points R^{e_g} and L^{e_g} are given, respectively by

$$R^{e_g} = \rho^{e_g} B^{e_g} + (1 - \rho^{e_g}) C^{e_g} \quad \text{and} \quad L^{e_g} = \lambda^{e_g} B^{e_g} + (1 - \lambda^{e_g}) C^{e_g},$$

where $\rho^{e_g} \in [0, 1]$ and $\lambda^{e_g} \in [0, 1]$ are variables to determine the position of the points in the segment.

As discussed in Section 2, we consider two modes of visit to the target graphs $g \in \mathcal{G}$:

(i) Visiting a fraction α^{e_g} of each edge e_g which can be modelled by using the following constraints:

$$|\lambda^{e_g} - \rho^{e_g}| \geq \alpha^{e_g}, \quad \forall e_g \in E_g. \tag{α-E}$$

These inequalities state that the difference between the parameterisations of the entry and exit points associated with with each edge e_g must be greater than the fraction of the length of e_g required for traverse.

(ii) Visit a fraction α^g of the total length of the graph:

$$\sum_{e_g \in E_g} \mu^{e_g} |\lambda^{e_g} - \rho^{e_g}| \mathcal{L}(e_g) \geq \alpha^g \mathcal{L}(g). \tag{α-G}$$

This constraint ensures that the sum of the fractions of the length of the edges chosen to be crossed must be greater than the fraction of the length of g that must be traversed.

In both cases, the corresponding constraints are non-linear. To linearise them, we need to introduce a binary variable entry^{e_g} that determines the direction of travel at the edge e_g, as well as the definition of the auxiliary variables $v^{e_g}_{\min}$ and $v^{e_g}_{\max}$ of the access and exit points in that segment. Then, for each edge e_g, the absolute value constraint (α-E) can be represented by:

$$|\rho^{e_g} - \lambda^{e_g}| \geq \alpha^{e_g} \Longleftrightarrow \begin{cases} \rho^{e_g} - \lambda^{e_g} & = & v^{e_g}_{\max} - v^{e_g}_{\min}, \\ v^{e_g}_{\max} & \leq & 1 - \text{entry}^{e_g}, \\ v^{e_g}_{\min} & \leq & \text{entry}^{e_g}, \\ v^{e_g}_{\min},\ v^{e_g}_{\max} & \geq & 0, \\ v^{e_g}_{\max} + v^{e_g}_{\min} & \geq & \alpha^{e_g}. \end{cases} \tag{α-E}$$

The first four inequalities model the standard trick of linearisation of the absolute value. The last constraint ensures that the value of

Table 8
Decision variables for AMMDRPG with a non-homogeneous fleet of drones.

Binary and integer decision variables
$\mu^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G})$: equal to 1 if edge e of graph g (or a fraction of it) is visited by the drone, 0 otherwise.
$\text{entry}^{e_g} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G})$: auxiliary binary variable used for linearising expressions.
$u^{e_g o\delta} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall o \in \mathcal{O},\ \forall \delta \in D$: equal to 1 if the drone δ enters in graph g by the edge e_g at operation o, 0 otherwise.
$z^{e_g e'_g} \in \{0,1\},\ \forall e_g, e'_g \in E_g\ (g \in \mathcal{G})$: equal to 1 if the drone goes from e_g to e'_g, 0 otherwise.
$v^{e_g o\delta} \in \{0,1\},\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall o \in \mathcal{O},\ \forall \delta \in D$: equal to 1 if the drone δ exits from graph g by e_g at operation o, 0 otherwise.
Continuous decision variables
$s^{e_g} \in [0, \lvert E_g\rvert - 1],\ \forall e_g \in E_g\ (g \in \mathcal{G})$: continuous non-negative variable representing the order of visit to the edge e of graph g.
$\rho^{e_g} \in [0,1]$ and $\lambda^{e_g} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathcal{G})$: defining the entry and exit points on e_g.
$\nu^{e_g}_{\min}$ and $\nu^{e_g}_{\max} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathcal{G})$: auxiliary variables used for linearising expressions.
$p^{e_g} \in [0,1],\ \forall e_g \in E_g\ (g \in \mathcal{G})$: auxiliary variable used for modelling the product of μ^{e_g} and $\lvert\lambda^{e_g} - \rho^{e_g}\rvert$.
$x^o_L \in \mathbb{R}^2,\ \forall o \in \mathcal{O}$: coordinates representing the point where the mothership launches the drones at operation o.
$x^o_R \in \mathbb{R}^2,\ \forall o \in \mathcal{O}$: coordinates representing the point where the mothership retrieves the drones at operation o.
$R^{e_g} \in \mathbb{R}^2,\ \forall e_g \in E_g\ (g \in \mathcal{G})$: coordinates representing the entry point on edge e_g of graph g.
$L^{e_g} \in \mathbb{R}^2,\ \forall e_g \in E_g\ (g \in \mathcal{G})$: coordinates representing the exit point on edge e_g of graph g.
$d_L^{e_g o\delta} \geq 0,\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall o \in \mathcal{O},\ \forall \delta \in D$: representing the distance travelled by the drone δ from the launching point x^o_L on the mothership at operation o to the first visiting point R^{e_g} on e_g.
$p_L^{e_g o\delta} \geq 0,\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall o \in \mathcal{O},\ \forall \delta \in D$: auxiliary variable used for modelling the product of $d_L^{e_g o\delta}$ and $u^{e_g o\delta}$.
$d^{e_g} \geq 0,\ \forall e_g \in E_g\ (g \in \mathcal{G})$: representing the distance travelled by the drone from the retrieval point R^{e_g} to the launching point L^{e_g} on e_g.
$d^{e_g e'_g} \geq 0,\ \forall e_g, e'_g \in E_g\ (g \in \mathcal{G})$: representing the distance travelled by the drone from the launching point L^{e_g} on e_g to the retrieval point $R^{e'_g}$ on e'_g.
$p^{e_g e'_g} \geq 0,\ \forall e_g, e'_g \in E_g\ (g \in \mathcal{G})$: auxiliary variable used for modelling the product of $d^{e_g e'_g}$ and $z^{e_g e'_g}$.
$d_R^{e_g o\delta} \geq 0,\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall o \in \mathcal{O},\ \forall \delta \in D$: representing the distance travelled by the drone δ from the last visiting point L^{e_g} on e_g to the retrieval point x^o_R on the mothership at operation o.
$p_R^{e_g o\delta} \geq 0,\ \forall e_g \in E_g\ (g \in \mathcal{G}),\ \forall o \in \mathcal{O},\ \forall \delta \in D$: auxiliary variable used for modelling the product of $d_R^{e_g o\delta}$ and $v^{e_g o\delta}$.
$d_{orig} \geq 0$: distance from the origin $orig$ to the first launching point x^1_L.
$d^o_{LR} \geq 0,\ \forall o \in \mathcal{O}$: representing the distance travelled by the mothership from the launching point x^o_L to the retrieval point x^o_R at operation o.
$d^o_{RL} \geq 0,\ \forall o \in \mathcal{O} \setminus \lvert\mathcal{O}\rvert$: representing the distance travelled by the mothership from the retrieval point x^o_R at operation o to the launching point $x^{(o+1)}_L$ at operation $o+1$.
$d_{dest} \geq 0$: distance from the last retrieval point $x^{\lvert\mathcal{O}\rvert}_R$ to the destination $dest$.
$time^o_D \geq 0,\ \forall o \in \mathcal{O}$: maximum time spent by a drone during operation o.
$time^o_M \geq 0,\ \forall o \in \mathcal{O}$: time spent by the mothership to go from the launching point x^o_L to the retrieval point x^o_R of operation o.
$time_M \geq 0$: total time spent by the mothership to go from the origin to the destination (makespan).

the linear expression of the absolute value is higher than the required fraction α^{e_g}.

Similarly, (α-G) can be linearised as follows:

$$\sum_{e_g \in E_g} \mu^{e_g} |\rho^{e_g} - \lambda^{e_g}| \mathcal{L}(e_g) \geq \alpha^g \mathcal{L}(g).$$

$$\iff \begin{cases} \rho^{e_g} - \lambda^{e_g} & = & \nu^{e_g}_{\max} - \nu^{e_g}_{\min}, \\ \nu^{e_g}_{\max} & \leq & 1 - \text{entry}^{e_g}, \\ \nu^{e_g}_{\min} & \leq & \text{entry}^{e_g}, \\ \nu^{e_g}_{\min},\ \nu^{e_g}_{\max} & \geq & 0, \\ p^{e_g} & \leq & \nu^{e_g}_{\max} + \nu^{e_g}_{\min}, \\ p^{e_g} & \leq & \mu^{e_g}, \\ p^{e_g} & \geq & \nu^{e_g}_{\max} + \nu^{e_g}_{\min} + \mu^{e_g} - 1, \\ \sum_{e_g \in E_g} p^{e_g} \mathcal{L}(e_g) & \geq & \alpha^g \mathcal{L}(g), \end{cases} \qquad (\alpha\text{-G})$$

where p^{e_g} is the auxiliary variable that represents the product of the binary variable μ^{e_g} and the difference in absolute values $|\rho^{e_g} - \lambda^{e_g}|$. The first four inequalities again linearise the absolute value expression. The following three constraints model the product of the expression of the absolute value and the binary variable μ^{e_g}. The last inequality ensures that the fraction of the length of those edges chosen to be crossed must be greater than the fraction of the length of g required to be traversed.

Elimination of subtours

As already presented in Section 3, to prevent the existence of subtours within each graph $g \in \mathcal{G}$ that the drone must visit, one can include, among others, the compact formulation that uses Miller-Tucker-Zemlin constraints (MTZ) or subtour elimination constraints (SEC).

For the MTZ formulation, we use continuous variables s^{e_g}, defined in Table 8, which state the order to visit the edge e_g and set the following constraints for each $g \in \mathcal{G}$:

$$s^{e_g} - s^{e'_g} + |E_g| z^{e_g e'_g} \leq |E_g| - 1, \qquad \forall e_g \neq e'_g \in E_g, \qquad (\text{MTZ}_1)$$

$$0 \leq s^{e_g} \leq |E_g| - 1, \qquad \forall e_g \in E_g. \qquad (\text{MTZ}_2)$$

Alternatively, we can also use the family of subtour elimination constraints for each $g \in \mathcal{G}$:

$$\sum_{e_g, e'_g \in S} z_g^{e_g e'_g} \leq |S| - 1, \quad \forall S \subset E_g. \qquad (\text{SEC})$$

To find the SEC inequalities, as usual, we search for disconnected components in the current solution. Among them, we choose the shortest subtour found in the solution to be added as a lazy constraint to the model.

Drone constraints

To model this problem, as described in Section 3.1, we adopt the concept of operation. Let us denote by $\mathcal{O}$ the set of operations that the

mothership and drone fleet have to perform. These operations are visits to the different graphs in $\mathcal{G}$ with the required constraints. An operation $o \in \mathcal{O}$ refers to the event in which the mothership launches some drones from a take-off location, denoted by x_L^o and then takes them back to a retrieval location x_R^o.

For each operation $o \in \mathcal{O}$, each of the drones launched from the mothership must follow a path that starts from and returns to the mothership, while visiting the required edges of g.

To include the definition of these paths in our mathematical programming formulation, we need to make decisions to choose:

(i) The optimal assignment of drones to visit graphs in a given operation o.
(ii) The order to visit the edges of each graph in its corresponding operation.

We model the route that the drone follows using the binary variables $u^{e_g o\delta}$, $z^{e_g e'_g}$ and $v^{e_g o\delta}$ defined in Table 8.

$$\sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} u^{e_g o\delta} \le 1, \qquad \forall o \in \mathcal{O}, \forall \delta \in \mathcal{D}. \qquad \text{(Drone ROUTE}_1^{\Delta}\text{-CO)}$$

$$\sum_{g\in\mathcal{G}}\sum_{e_g\in E_g} v^{e_g o\delta} \le 1, \qquad \forall o \in \mathcal{O}, \forall \delta \in \mathcal{D}. \qquad \text{(Drone ROUTE}_2^{\Delta}\text{-CO)}$$

$$\sum_{e_g\in E_g}\sum_{o\in\mathcal{O}}\sum_{\delta\in D} u^{e_g o\delta} = 1, \qquad \forall g \in \mathcal{G}, \qquad \text{(Drone ROUTE}_3^{\Delta}\text{-CO)}$$

$$\sum_{e_g\in E_g}\sum_{o\in\mathcal{O}}\sum_{\delta\in D} v^{e_g o\delta} = 1, \qquad \forall g \in \mathcal{G}, \qquad \text{(Drone ROUTE}_4^{\Delta}\text{-CO)}$$

$$\sum_{e_g\in E_g} u^{e_g o\delta} = \sum_{e_g\in E_g} v^{e_g o\delta}, \qquad \forall g \in \mathcal{G}, \forall o \in \mathcal{O}, \forall \delta \in \mathcal{D}, \qquad \text{(Drone ROUTE}_5^{\Delta}\text{-CO)}$$

$$\sum_{o\in\mathcal{O}}\sum_{\delta\in D} u^{e_g o\delta} + \sum_{e'_g\in E_g} z_g^{e'_g e_g} = \mu^{e_g}, \qquad \forall e_g \in E_g : g \in \mathcal{G}, \qquad \text{(Drone ROUTE}_6^{\Delta}\text{-CO)}$$

$$\sum_{o\in\mathcal{O}}\sum_{\delta\in D} v^{e_g o\delta} + \sum_{e'_g\in E_g} z_g^{e_g e'_g} = \mu^{e_g}, \qquad \forall e_g \in E_g : g \in \mathcal{G}. \qquad \text{(Drone ROUTE}_7^{\Delta}\text{-CO)}$$

The inequalities (Drone ROUTE$_1^{\Delta}$-CO) and (Drone ROUTE$_2^{\Delta}$-CO) state that a drone δ visits at most one graph g at operation o. Constraints (Drone ROUTE$_3^{\Delta}$-CO) and (Drone ROUTE$_4^{\Delta}$-CO) ensure that each graph is visited at some operation o by some drone δ. Eqs. (Drone ROUTE$_5^{\Delta}$-CO) ensure that the operation of entering and exiting the graph g occurs in the same operation o and is performed by the same drone δ. Constraints (Drone ROUTE$_6^{\Delta}$-CO) state that if an edge e of graph g is visited by the drone δ, one of two alternative situations must occur: either e is the first edge of graph g visited by the drone δ at operation o, or edge e is visited by the drone δ after visiting another edge e' of graph g. Similarly, constraints (Drone ROUTE$_7^{\Delta}$-CO) state that if an edge e of graph g is visited by the drone δ, either e is the last edge of graph g visited by the drone at operation o, or the drone δ must move to another edge e' of graph g after visiting edge e.

Distance and time constraints

The goal of the AMMDRPG is to find a feasible solution that minimises the total time taken by the mothership (makespan). To account for the different distances between the decision variables of the model, we need to set the continuous variables $d_L^{e_g o\delta}$, d^{e_g}, $d^{e_g e'_g}$, $d_R^{e_g o\delta}$, d_{orig}, d_{RL}^o, d_{LR}^o and d_{dest} defined in Table 8. This can be done by means of the following constraints:

$$\|x_L^o - R^{e_g}\| \le d_L^{e_g o\delta}, \qquad \forall e_g \in E_g : g \in \mathcal{G}, \forall o \in \mathcal{O}, \forall \delta \in \mathcal{D}, \qquad \text{(Drone DIST}_1^{\Delta}\text{-CO)}$$

$$\|R^{e_g} - L^{e_g}\| \le d^{e_g}, \qquad \forall e_g \in E_g : g \in \mathcal{G}, \qquad \text{(Drone DIST}_2^{\Delta}\text{-CO)}$$

$$\|R^{e_g} - L^{e'_g}\| \le d^{e_g e'_g}, \qquad \forall e_g \ne e'_g \in E_g : g \in \mathcal{G}, \qquad \text{(Drone DIST}_3^{\Delta}\text{-CO)}$$

$$\|L^{e_g} - x_R^o\| \le d_R^{e_g o\delta}, \qquad \forall e_g : g \in \mathcal{G}, \forall o \in \mathcal{O}, \forall \delta \in \mathcal{D}, \qquad \text{(Drone DIST}_4^{\Delta}\text{-CO)}$$

$$\|orig - x_L^1\| \le d_{orig}, \qquad \text{(Mothership DIST}_1^{\Delta}\text{-CO)}$$

$$\|x_L^o - x_R^o\| \le d_{LR}^o, \qquad \forall o \in \mathcal{O}. \qquad \text{(Mothership DIST}_2^{\Delta}\text{-CO)}$$

$$\|x_R^o - x_L^{o+1}\| \le d_{RL}^o, \qquad \forall o \in \mathcal{O} : o < |\mathcal{O}|, \qquad \text{(Mothership DIST}_3^{\Delta}\text{-CO)}$$

$$\|x_R^{|\mathcal{O}|} - dest\| \le d_{dest}, \qquad \text{(Mothership DIST}_4^{\Delta}\text{-CO)}$$

Thus, we can express the time that a drone $\delta \in D$ takes to visit a graph $g \in \mathcal{G}$ during operation $o \in \mathcal{O}$ as follows:

$$time_\delta^o \ge \frac{1}{v_\delta}\left(\sum_{e_g\in E_g} u^{e_g o\delta} d_L^{e_g o\delta} + \sum_{e_g, e'_g\in E_g} z^{e_g e'_g} d^{e_g e'_g} + \sum_{e_g\in E_g} \mu^{e_g} d^{e_g} + \sum_{e_g\in E_g} v^{e_g o\delta} d_R^{e_g o\delta}\right) - N_\delta\left(1 - \sum_{e_g\in E_g} u^{e_g o\delta}\right) \qquad \text{(Drone TIME}_o^{\Delta}\text{-CO)}$$

The first addend within the brackets in the RHS of the constraint (Drone TIME$_o^{\Delta}$-CO), accounts for the time spent by the drone δ to depart from the launch point x_L^o to the first retrieval point on the graph R^{e_g}. The second addend considers the time consumed by the drone to go from the edge e_g to e'_g on the graph g. The third computes the time required to traverse the required edges in g. The fourth measures the time taken to travel from the last launching point $L^{e''_g}$ to the retrieval point x_R^o. The bigM term ensures that the constraint becomes active only when a graph g is visited during the operation o by the drone δ. The reader may observe that the endurance constraint (Endurance$^{\Delta}$-CO) restricts the time the drone spends performing the operation o to be less than its endurance N_δ. Hence, it is possible to take N_δ as the bigM constant in (Drone TIME$_o^{\Delta}$-CO).

In order to compute the maximum time a drone can spend visiting a graph $g \in \mathcal{G}$ associated with the operation o $\forall o \in \mathcal{O}$, we introduce the following constraints:

$$time_D^o \ge time_\delta^o \ \forall \delta \in D \qquad \text{(Drone MAX TIME}_o^{\Delta}\text{-CO)}$$

Constraints (Mothership TIME$_o^{\Delta}$-CO) define the time that the mothership takes to go from the launch point x_L^o to the retrieval point x_R^o associated with the operation o.

$$time_M^o = \frac{d_{LR}^o}{v_M} \ \forall o \in \mathcal{O} \qquad \text{(Mothership TIME}_o^{\Delta}\text{-CO)}$$

Thus, the overall time spent by the mothership to move from the origin to the destination (makespan) can be expressed as follows:

$$time_M = \frac{1}{v_M}(d_{orig} + \sum_{o\in\mathcal{O}}(d_{LR}^o + d_{RL}^o) + d_{dest}) \qquad \text{(Mothership TIME}^{\Delta}\text{-CO)}$$

Coordination and endurance constraints

The coordination between the drones and the mothership must ensure that the maximum time $time_D^o$ spent by a drone to visit a graph g at operation o is less than or equal to the time that the mothership needs to move from the launching point to the retrieval point during operation o. To this end, we need to define the following coordination constraint for each operation $o \in \mathcal{O}$:

$$time_D^o \le time_M^o. \qquad \text{(DCW}^{\Delta}\text{-CO)}$$

We can model the time endurance constraint for a particular operation $o \in \mathcal{O}$ and the drone $\delta \in \mathcal{D}$ by limiting the time travelled by the drone δ for this operation o:

$$time^o_\delta \leq N_\delta. \qquad \text{(Endurance}^\Delta\text{-CO)}$$

AMMDRPG-complete overlapping formulation (with non-homogeneous fleet of drones)

Putting together all the constraints introduced before, the following formulation minimises the total time travelled by the mothership (makespan), ensuring coordination with the drone fleet while guaranteeing the required coverage of the target graphs.

min $time_M$

(AMMDRPG-CO with a non-homogeneous fleet of drones)

s.t. (α-E) or (α-G),
(MTZ_1)–(MTZ_2) or (SEC),
(Drone ROUTE$^\Delta_1$-CO)–(Drone ROUTE$^\Delta_7$-CO),
(Drone DIST$^\Delta_1$-CO)–(Drone DIST$^\Delta_4$-CO),
(Mothership DIST$^\Delta_1$-CO)–(Mothership DIST$^\Delta_4$-CO),
(Drone TIME$^\Delta_o$-CO), (Drone MAX TIME$^\Delta_o$-CO),
(Mothership TIME$^\Delta_o$-CO), (Mothership TIME$^\Delta$-CO),
(DCW$^\Delta$-CO), (Endurance$^\Delta$-CO)

The objective function accounts for the time travelled by the mothership (makespan). Constraints (Drone ROUTE$^\Delta_1$-CO)–(Drone ROUTE$^\Delta_7$-CO) model the route followed by the drone $\delta \in \mathcal{D}$, (MTZ_1)–(MTZ_2) or (SEC) ensure that the displacement of the drone $\delta \in \mathcal{D}$ assigned to the target graph $g \in \mathcal{G}$ is a route, (α-E) or (α-G) define what is required in each visit to a target graph. Finally, constraints (Drone DIST$^\Delta_1$-CO)–(Mothership DIST$^\Delta_4$-CO) set the variables $d_L^{e_g o\delta}$, d^{e_g}, $d^{e_g e'_g}$, $d_R^{e_g o\delta}$, d_{orig}, d^o_{RL}, d^o_{LR} and d_{dest}, defined in Table 8, which represent the Euclidean distances needed in the model.

Strengthening the formulations

In this section, we present some results that adjust the bigM constants for each of the models. These constants appear when we linearise the bilinear terms of (Drone TIME$^\Delta_o$-CO). We use the McCormick's envelopes by adding variables $p \geq 0$ representing the products. To strengthen the formulations, we provide tight upper and lower bounds for these constants. The reader may note that the same bounds can be used for both models. Therefore, wlog, we focus on the bigM constants that appear in (AMMDRPG-CO with a non-homogeneous fleet of drones).

Big M constants bounding the distance from the launch/retrieval point on the path followed by the mothership to the retrieval/launch point on the target graph $g \in \mathcal{G}$

To linearise the first addend in (DCW-CO), we define the auxiliary non-negative continuous variables $p_L^{e_g o\delta}$ (resp. $p_R^{e_g o\delta}$) and we model the product by including the following constraints:

$$p_L^{e_g o\delta} \leq M_L^{e_g o\delta} u^{e_g o\delta},$$
$$p_L^{e_g o\delta} \leq d_L^{e_g o\delta},$$
$$p_L^{e_g o\delta} \geq m_L^{e_g o\delta} u^{e_g o\delta},$$
$$p_L^{e_g o\delta} \geq d_L^{e_g o\delta} - M_L^{e_g o\delta}(1 - u^{e_g o\delta}).$$

Note that, among all graph nodes and the origin and destination points, it is possible to identify the pair of points at the maximum distance. From this pair of points, we can build a circle whose diameter is the segment that joins them. Hence, because we are minimising the distance travelled by the mothership, every launch or retrieval point is inside this circle, and the best upper bound $M_L^{e_g o\delta}$ or $M_R^{e_g o\delta}$ can be described as:

$$M_R^{e_g o\delta} = \max_{\{v \in V_g \cup \{\text{orig,dest}\}, v' \in V_{g'} \cup \{\text{orig,dest}\} : g, g' \in \mathcal{G}\}} \|v - v'\| = M_L^{e_g o\delta}.$$

On the other hand, the minimum distance in this case can be zero. This bound is attainable whenever the launch or retrieval points of the mothership are the same as the retrieval or launch point on the target graph $g \in \mathcal{G}$.

Bounds on the bigM constants for the distance from the launch to the retrieval points on the target graph $g \in \mathcal{G}$

When the drone visits a graph g, it has to go from one edge e_g to another edge e'_g depending on the order given by $z^{e_g e'_g}$. This fact produces a product of variables linearised by the following constraints:

$$p^{e_g e'_g} \leq M^{e_g e'_g} z^{e_g e'_g},$$
$$p^{e_g e'_g} \leq d^{e_g e'_g},$$
$$p^{e_g e'_g} \geq m^{e_g e'_g} d^{e_g e'_g},$$
$$p^{e_g e'_g} \geq d^{e_g e'_g} - M^{e_g e'_g}(1 - z^{e_g e'_g}).$$

Since we take into account the distance between two edges $e_g = (B^{e_g}, C^{e_g})$, $e_g{}' = (B^{e'_g}, C^{e'_g}) \in E_g$, the maximum distance between their vertices gives us the upper bound:

$$M^{e_g e'_g} = \max\{\|B^{e_g} - C^{e'_g}\|, \|B^{e_g} - B^{e'_g}\|, \|C^{e_g} - B^{e'_g}\|, \|C^{e_g} - C^{e'_g}\|\}.$$

We observe that the minimum distance between edges $m^{e_g e'_g}$ can easily be obtained by computing the minimum distance between two edges, which results in a simple second-order cone program.

Experimental results

In this section, we discuss the results obtained by testing the formulation of AMMDRPG with a non-homogeneous fleet of drones, presented in Appendix, on the same set of instances described in Section 6.

Table 5 reports the results obtained by adopting the Gurobi commercial solver. We consider the exact solution that provides and does not provide an initial solution computed by the matheuristic described in Section 5. More precisely, the first column of Table 9 indicates the number of target graphs to be visited by the drone fleet, the second column reports the endurance of the drones, and the third column distinguishes between the visit of a fraction of each edge (e) and a fraction of each target graph (g). The fourth column reports the size of the drone fleet. This last column contains three subcolumns reporting, for each cardinality of the set $\mathcal{D}$, respectively, the average gap without initialisation (wi), the average gap with initialisation of the solution provided by the matheuristic (i) and the solution time, in seconds, of the matheuristic (TimeH) for each combination of the listed parameters. The time limit for these experiments is set equal to 2 h.

We can observe that the value of the average gap ranges between a minimum of 0.67 and a maximum of 0.97. This shows that the model is hard to solve even with small-sized instances. Furthermore, we can see that, in most cases, the average gap associated with the variant of the model consisting of visiting a given fraction of each edge is greater than the one associated with the variant that imposes visiting a given fraction of each target graph. Another thing we can observe is that the average gap increases with the number of drones and decreases with the drone endurance.

Regarding the number of target graphs, we can see that, increasing it from 5 to 10, the exact method without initialisation of the solution obtained with the matheuristic is even harder. Indeed, the red entries of the table mean that some instances could not find a feasible solution within the time limit (note that in the brackets we indicate the number

Table 9
Comparison between an exact solution with and without initialisation by the matheuristic solution.

$\|\mathcal{G}\|$	N_D	v.t.	$\|\mathcal{D}\|$								
			1			2			3		
			Gap (wi)	Gap (i)	TimeH	Gap (wi)	Gap (i)	TimeH	Gap (wi)	Gap (i)	TimeH
5	20	e	0.82	0.83	61.56	0.9	0.92	63.80	0.91	0.93	60.87
		g	0.80	0.79	44.97	0.92	0.89	37.32	0.96	0.94	39.05
	30	e	0.80	0.83	65.21	0.82	0.85	64.41	0.90	0.92	63.34
		g	0.71	0.76	55.77	0.88	0.84	44.36	0.91	0.91	44.59
	40	e	0.78	0.81	68.81	0.82	0.83	64.80	0.86	0.91	63.19
		g	0.73	0.74	43.92	0.84	0.81	38.27	0.90	0.85	37.51
	50	e	0.74	0.77	66.67	0.80	0.81	63.86	0.86	0.85	63.51
		g	0.67	0.71	43.42	0.89	0.81	43.98	0.83	0.80	44.35
	60	e	0.72	0.76	67.68	0.80	0.82	66.08	0.82	0.84	64.40
		g	0.73	0.78	44.69	0.86	0.79	40.63	0.85	0.82	50.01
10	20	e	0.85	0.83	137.93	–	0.92	128.53	–	0.95	124.44
		g	0.85 (2)	0.81	119.20	0.97 (2)	0.90	83.50	0.97 (3)	0.97	70.00
	30	e	0.81	0.81	159.00	0.88 (3)	0.87	132.15	0.93 (2)	0.95	127.35
		g	0.83 (1)	0.80	132.67	0.86 (3)	0.86	80.29	0.9 (1)	0.91	76.72
	40	e	0.78	0.79	191.37	0.84	0.85	131.26	0.89 (1)	0.92	132.10
		g	0.80	0.80	115.00	0.85 (3)	0.87	68.39	0.92 (1)	0.96	69.40
	50	e	0.78	0.81	188.32	0.85 (1)	0.88	134.01	0.91 (3)	0.93	132.82
		g	0.80	0.80	87.23	0.84 (3)	0.83	66.14	0.92 (2)	0.92	64.94
	60	e	0.82	0.84	155.27	0.83 (2)	0.86	131.94	0.87 (3)	0.92	130.11
		g	0.78	0.77	97.89	0.88 (2)	0.87	76.53	0.92 (3)	0.94	69.53

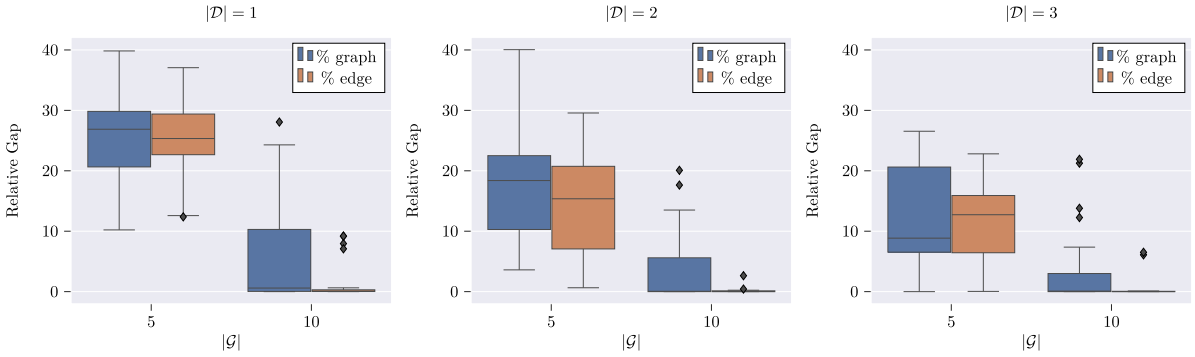


Fig. 20. Relative gap boxplots.

of these instances). The number of instances not solved increases with the number of drones. Furthermore, for the minimum level of endurance, the exact solution of the model without initialisation provided by the matheuristic does not provide any solution within the time limit for instances with 10 graphs and 2 or 3 drones.

Considering the comparison with the exact method starting from the solution provided by the matheuristic, we can note that the values of the average gap are very close to those related to the exact solution method without initialisation. Thus, initialisation does not speed up the convergence of the solver. However, we can see that the matheuristic is always able to find a feasible solution to the problem, even for the cases in which the solver is not (instances with 10 graphs and 2 or 3 drones and minimum value of endurance).

Moreover, the average solution times of the matheuristic range between a minimum of 37 s to a maximum of 3 min. They increase with the drone endurance for the variant of the model in which a given fraction of each edge must be visited, while they decrease by increasing the number of drones for the variant of the model in which a given fraction of each target graph must be visited. By increasing the number of target graphs from 5 to 10, the average solution times of the matheuristic become more than double for both model variants.

Summing up, the results obtained show that the exact solution method given by solving the formulation is very challenging even for small-sized instances. However, by exploiting this, the matheuristic is able to provide solutions for all instances quite quickly.

The boxplots in Fig. 20 represent the relative gap of the solution provided by the matheuristic concerning that provided by the exact solution of the mathematical programming model within the time limit, with the initialisation of the solution found by the matheuristic. We can observe that the maximum value of the relative gap is equal to 40, but it decreases when the number of target graphs increases from 5 to 10 and it tends to be smaller when a given fraction of each edge must be visited concerning the other case, that is, when a given fraction of each graph to be visited is imposed. In particular, for instances with 10 target graphs, the relative gap is not greater than 25, except an outlier in the case with 1 drone and a given fraction of each graph to be visited. When the number of drones is equal to 2, this maximum value decreases further, and is reduced to 10 when the fleet consists of 3 drones. Furthermore, when a given fraction of each edge to be visited is imposed, the relative gap is even smaller, around zero, in most of cases. Thus, we can conclude that the matheuristic, even though it does not speed up the convergence to the optimum, it provides solutions of good quality, especially for instances of large size, both in terms of target

graphs and drones, and for the most challenging variant of the problem in which a given fraction of each edge must be visited.

References

AltiGator, 2015. A drone to rescue immigrants. URL: https://altigator.com/en/rescue-immigrants-with-a-drone/.

Amorosi, L., Caprari, R., Crainic, T., Dell'Olmo, P., Ricciardi, N., 2020. CIRRELT-2020-17 An Integrated Routing-Scheduling Model for a Hybrid UAV-Based Delivery System. CIRRELT.

Amorosi, L., Chiaraviglio, L., D'Andreagiovanni, F., Blefari-Melazzi, N., 2018. Energy-efficient mission planning of UAVs for 5G coverage in rural zones. In: 2018 IEEE International Conference on Environmental Engineering. EE, pp. 1–9. http://dx.doi.org/10.1109/EE1.2018.8385250.

Amorosi, L., Chiaraviglio, L., Galan-Jimenez, J., 2019. Optimal energy management of uav-based cellular networks powered by solar panels and batteries: Formulation and solutions. IEEE Access 7, 53698–53717. http://dx.doi.org/10.1109/ACCESS.2019.2913448.

Amorosi, L., Puerto, J., Valverde, C., 2021. Coordinating drones with mothership vehicles: The mothership and drone routing problem with graphs. Comput. Oper. Res. 136, 105445. http://dx.doi.org/10.1016/j.cor.2021.105445.

Amorosi, L., Puerto, J., Valverde, C., 2022. An extended model of coordination of an all-terrain vehicle and a multivisit drone. Int. Trans. Oper. Res. http://dx.doi.org/10.1111/itor.13179.

Blanco, V., Fernández, E., Puerto, J., 2017. Minimum spanning trees with neighborhoods: Mathematical programming formulations and solution methods. European J. Oper. Res. 262 (3), 863–878. http://dx.doi.org/10.1016/j.ejor.2017.04.023.

Blanco, V., Puerto, J., Ben-Ali, S.E.-H., 2013. Revisiting several problems and algorithms in continuous location with ℓ_τ norms. Comput. Optim. Appl. 58, 563–595.

Campbell, J.F., Corberán, Á., Plana, I., Sanchis, J.M., 2018. Drone arc routing problems. Networks 72 (4), 543–559. http://dx.doi.org/10.1002/net.21858.

Campbell, J.F., Corberán, Á., Plana, I., Sanchis, J.M., Segura, P., 2021. Solving the length constrained K-drones rural postman problem. European J. Oper. Res. 292 (1), 60–72. http://dx.doi.org/10.1016/j.ejor.2020.10.035.

Campbell, J.F., Sweeney, D., Zhang, J., 2017. Strategic design for delivery with trucks and drones. Comput. Sci..

Carlsson, J.G., Song, S., 2018. Coordinated logistics with a truck and a drone. Manage. Sci. 64 (9), 4052–4069. http://dx.doi.org/10.1287/mnsc.2017.2824.

Chiaraviglio, L., Amorosi, L., Blefari-Melazzi, N., Dell'olmo, P., Lo Mastro, A., Natalino, C., Monti, P., 2019a. Minimum cost design of cellular networks in rural areas with UAVs, optical rings, solar panels, and batteries. IEEE Trans. Green Commun. Netw. 3 (4), 901–918. http://dx.doi.org/10.1109/TGCN.2019.2936012.

Chiaraviglio, L., Amorosi, L., Blefari-Melazzi, N., Dell'Olmo, P., Natalino, C., Monti, P., 2018. Optimal design of 5G networks in rural zones with UAVs, optical rings, solar panels and batteries. In: 2018 20th International Conference on Transparent Optical Networks. ICTON, pp. 1–4. http://dx.doi.org/10.1109/ICTON.2018.8473712.

Chiaraviglio, L., Amorosi, L., Malandrino, F., Chiasserini, C.F., Dell'Olmo, P., Casetti, C., 2019b. Optimal throughput management in UAV-based networks during disasters. In: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). pp. 307–312. http://dx.doi.org/10.1109/INFCOMW.2019.8845190.

Chung, S.H., Sah, B., Lee, J., 2020. Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. Comput. Oper. Res. 123, 105004. http://dx.doi.org/10.1016/j.cor.2020.105004.

Dayarian, I., Savelsbergh, M., Clarke, J.-P., 2020. Same-day delivery with drone resupply. Transp. Sci. 54 (1), 229–249. http://dx.doi.org/10.1287/trsc.2019.0944.

Dell'Amico, M., Montemanni, R., Novellani, S., 2021. Algorithms based on branch and bound for the flying sidekick traveling salesman problem. Omega 104, 102493. http://dx.doi.org/10.1016/j.omega.2021.102493.

Dönmez, Z., Kara, B.Y., Karsu, Ö., Saldanha-da Gama, F., 2021. Humanitarian facility location under uncertainty: Critical review and future prospects. Omega 102, 102393. http://dx.doi.org/10.1016/j.omega.2021.102393.

Ferrandez, S.M., Harbison, T., Weber, T., Sturges, R., Rich, R., 2016. Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. J. Ind. Eng. Manage. 9 (2), 374–388. http://dx.doi.org/10.3926/jiem.1929.

Garone, E., Naldi, R., Casavola, A., Frazzoli, E., 2010. Cooperative mission planning for a class of carrier-vehicle systems. In: 49th IEEE Conference on Decision and Control. CDC, pp. 1354–1359. http://dx.doi.org/10.1109/CDC.2010.5717171.

Jiménez, J.G., Chiaraviglio, L., Amorosi, L., Blefari-Melazzi, N., 2018. Multi-period mission planning of UAVs for 5G coverage in rural areas: a heuristic approach. In: 2018 9th International Conference on the Network of the Future. NOF, pp. 52–59. http://dx.doi.org/10.1109/NOF.2018.8598123.

Mathew, N., Smith, S.L., Waslander, S.L., 2015. Planning paths for package delivery in heterogeneous multirobot teams. IEEE Trans. Autom. Sci. Eng. 12 (4), 1298–1308. http://dx.doi.org/10.1109/TASE.2015.2461213.

Murray, C.C., Chu, A.G., 2015. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transp. Res. C 54, 86–109. http://dx.doi.org/10.1016/j.trc.2015.03.005.

Otto, A., Agatz, N., Campbell, J., Golden, B., Pesch, E., 2018. Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. Networks 72, 1–48. http://dx.doi.org/10.1002/net.21818.

Pei, Z., Dai, X., Yuan, Y., Du, R., Liu, C., 2021. Managing price and fleet size for courier service with shared drones. Omega 104, 102482. http://dx.doi.org/10.1016/j.omega.2021.102482.

Poikonen, S., Golden, B., 2020a. The mothership and drone routing problem. INFORMS J. Comput. 32 (2), 249–262. http://dx.doi.org/10.1287/ijoc.2018.0879.

Poikonen, S., Golden, B., 2020b. Multi-visit drone routing problem. Comput. Oper. Res. 113, 104802. http://dx.doi.org/10.1016/j.cor.2019.104802.

Puerto, J., Valverde, C., 2021. Project: Instances for the case study of the all terrain mothership multiple drone routing problem with graphs (AMMDRPG). URL: https://github.com/z72vamac/case_study_AMMDRPGST.

Reim, G., 2020. US army catches 'air-launched effect' drones in mid-air using another UAV. URL: https://www.flightglobal.com/military-uavs/us-army-catches-air-launched-effect-drones-in-mid-air-using-another-uav/140498.article.

Tokekar, P., Hook, J.V., Mulla, D., Isler, V., 2016. Sensor planning for a symbiotic UAV and UGV system for precision agriculture. IEEE Trans. Robot. 32 (6), 1498–1511. http://dx.doi.org/10.1109/TRO.2016.2603528.

Trotta, A., Andreagiovanni, F.D., Di Felice, M., Natalizio, E., Chowdhury, K.R., 2018. When UAVs ride a bus: Towards energy-efficient city-scale video surveillance. In: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. pp. 1043–1051. http://dx.doi.org/10.1109/INFOCOM.2018.8485863.

Ulmer, M.W., Thomas, B.W., 2018. Same-day delivery with heterogeneous fleets of drones and vehicles. Networks 72 (4), 475–505. http://dx.doi.org/10.1002/net.21855.

Wen, T., Zhang, Z., Wong, K.K.L., 2016. Multi-objective algorithm for blood supply via unmanned aerial vehicles to the wounded in an emergency situation. PLoS One 11 (5), http://dx.doi.org/10.1371/journal.pone.0155176.