European Journal of Operational Research xxx (xxxx) xxx

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 xxx

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

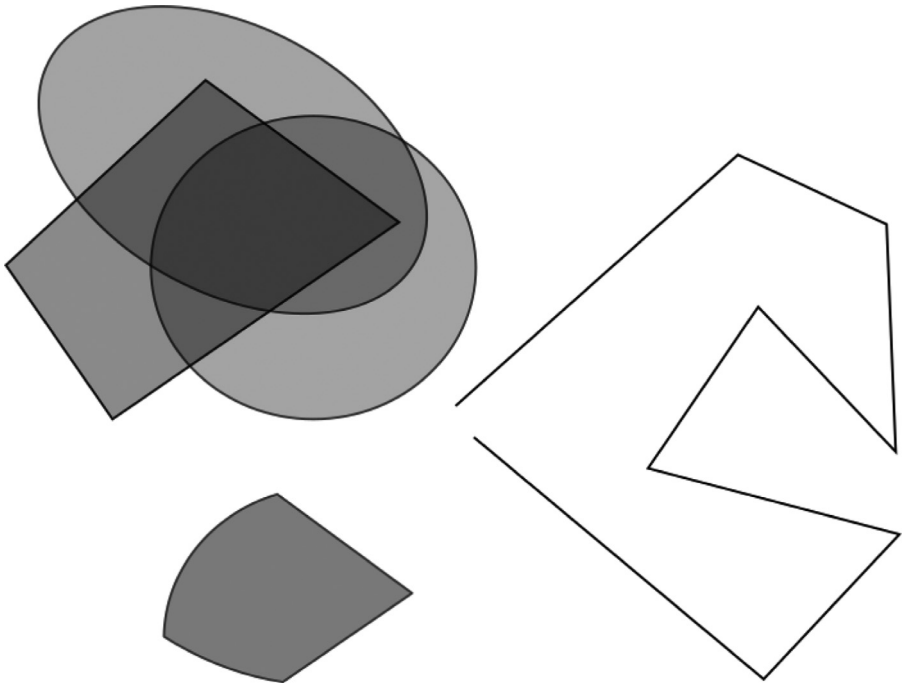


Fig. 1. An example of convex sets and polygonal chains considered in the problem.

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 $i = 1, 2$:

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

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, $j = 1, \ldots, n_{S_v}$.
- 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 $j = 1, \ldots, n_{S_v}$.
- λ_v^i: Continuous variable in $[0, n_{S_v}]$ 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 $i = 1, 2$:

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

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:

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

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).

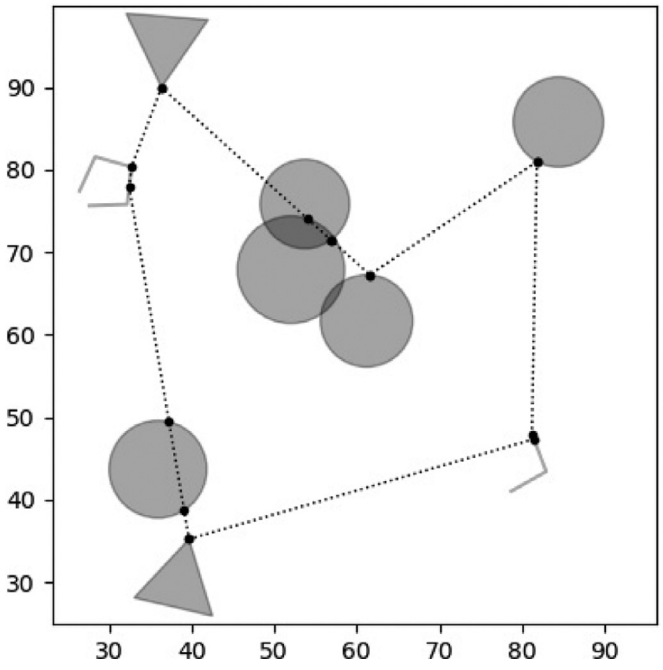


Fig. 2. An example with 9 elements: 7 convex sets and 2 polygonal chains.

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:

$$\begin{aligned} \min \quad & P = \sum_{e \in E_{\text{out}}} p_e + \sum_{v \in V} f_v d_v && \text{(RL-XPPN)} \\ \text{s.t.} \quad & p_e \geq d_e - M_e(1 - z_e) \quad \forall e \in E_{\text{out}} && \text{(LIN-Mc)} \\ & p_e \geq 0,\ \forall e \in E_{\text{out}} \\ & z \in \mathcal{T}_G \\ & (\text{D1}), (\text{D2}) \\ & (\mathcal{C}-\mathrm{C}), (\mathcal{P}-\mathrm{C}), (\alpha-\mathrm{C}) \end{aligned}$$

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.

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

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:

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

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).

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

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

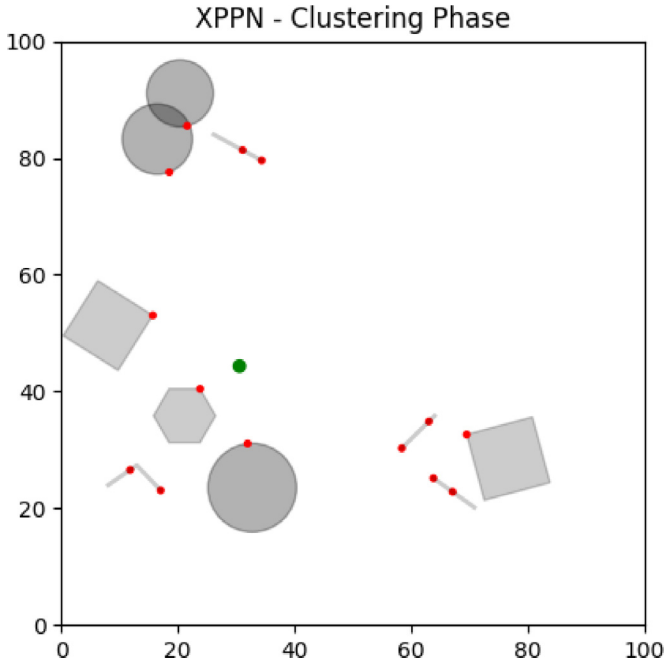


Fig. 3. Illustration of the first phase of the heuristic algorithm.

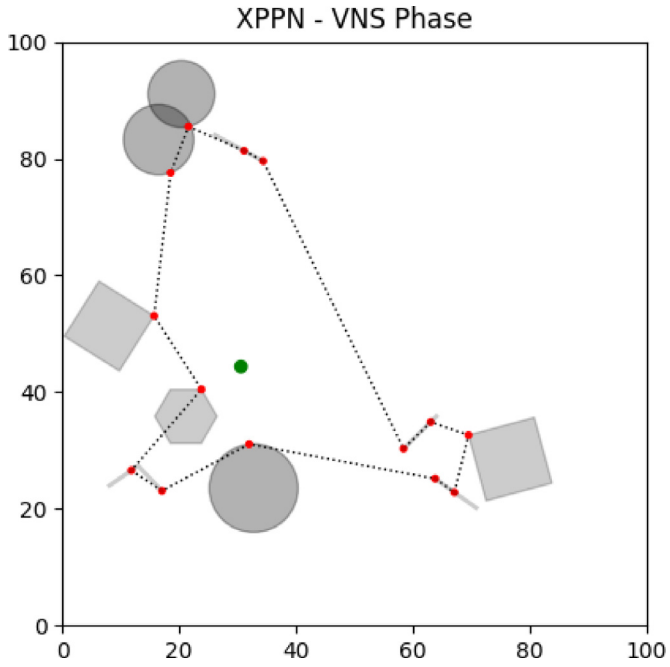


Fig. 4. Application of the VNS phase to the example of Fig. 3.

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. □

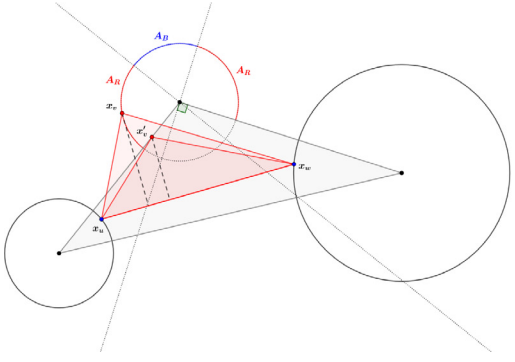
The special case in which all the neighborhoods are circles, allows us to limit even more the location of the points based on the construction given in the Proposition 1.

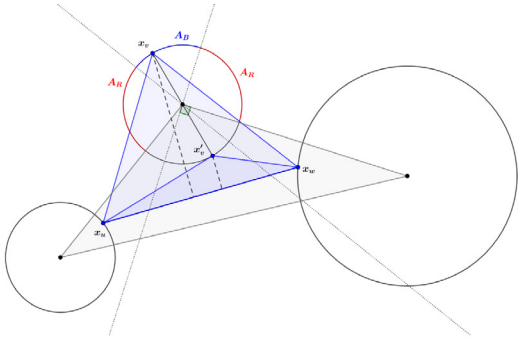
Corollary 2.1. *Any point selected in an optimal solution of the XPPN when all the neighborhoods are circles is placed in some arc of one of the circumferences inside of the convex hull generated by the center of the circles.*

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.

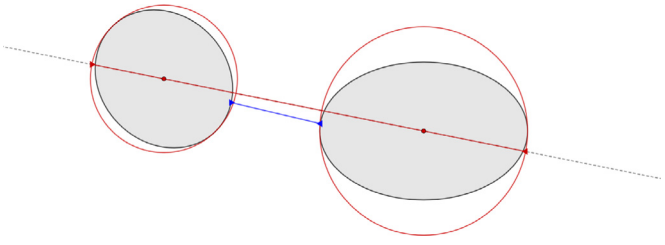


Fig. 5. Upper and lower bound when both sets are ellipsoids.

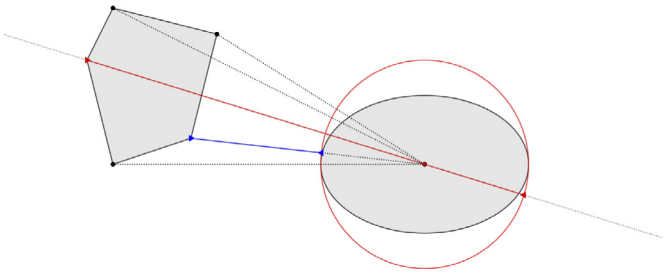


Fig. 6. Upper and lower bound when a set is a polygon and the other is an ellipsoid.

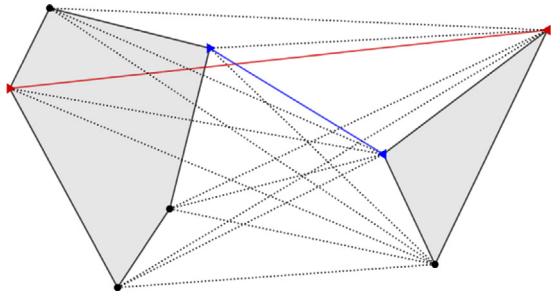


Fig. 7. Upper and lower bound when both sets are polygons.

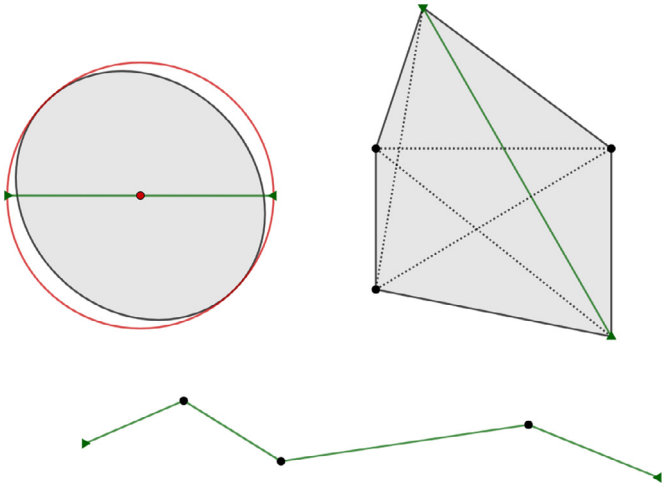


Fig. 8. Upper bound on the maximal distance within a set.

- 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-

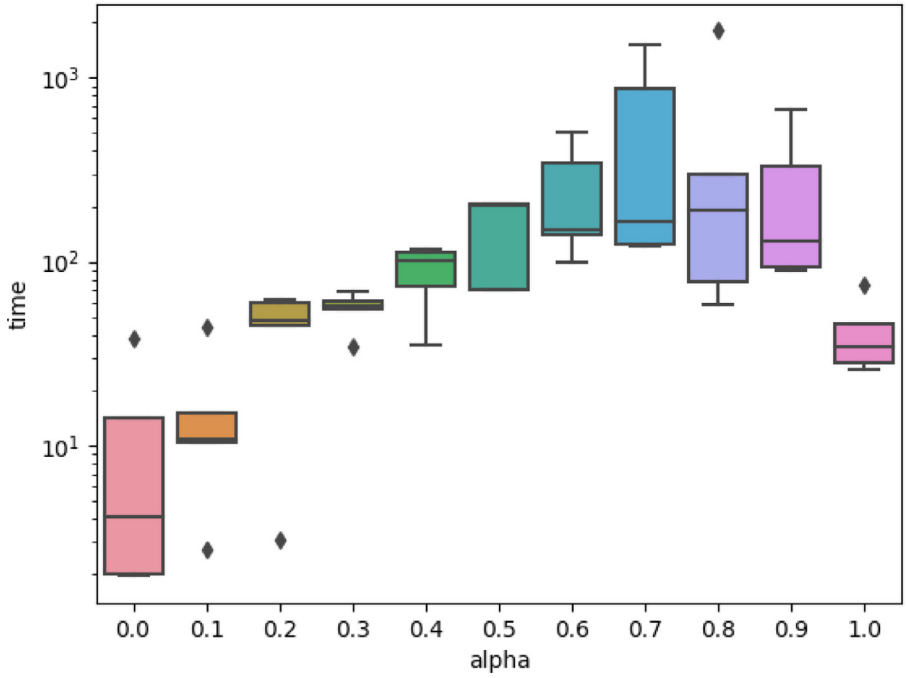


Fig. 9. Comparison of the times of the MTZ formulation varying the α parameter.

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).

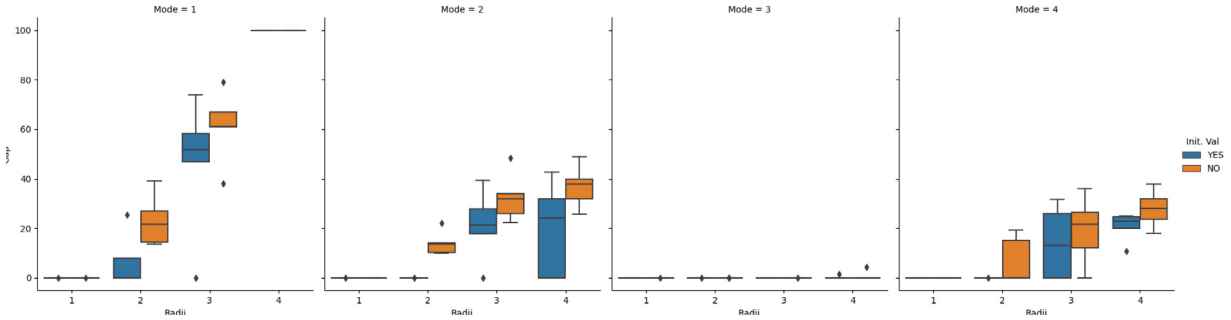


Fig. 10. Comparison of the final gap between MTZ formulation with and without initial solution after 7200 seconds for instances with 15 neighborhoods.

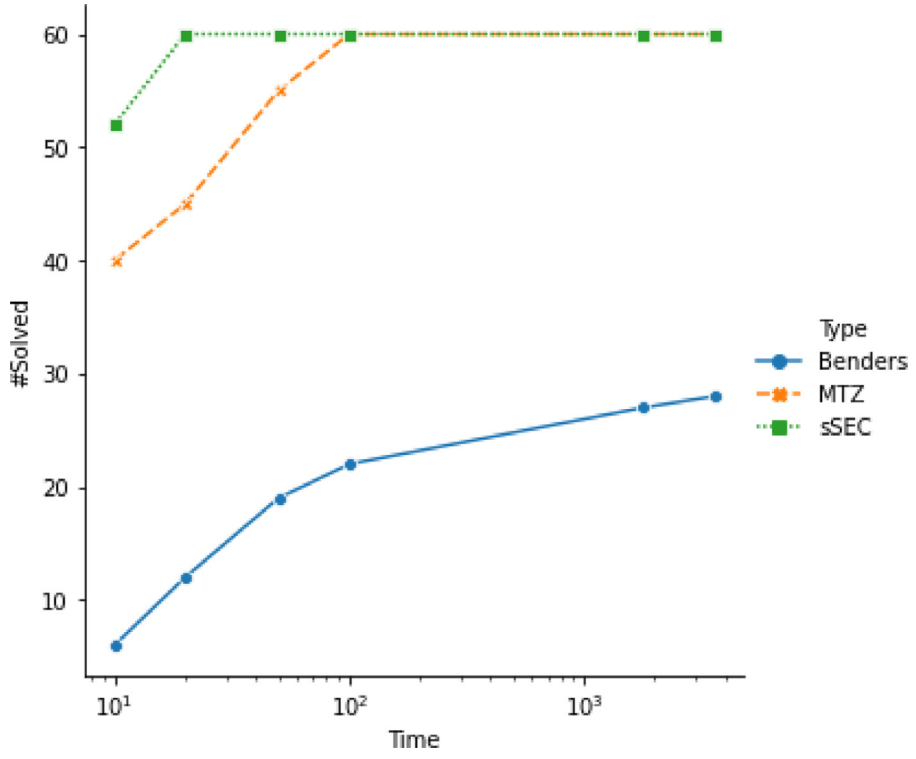


Fig. 11. Performance profile: Time vs #Solved.

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

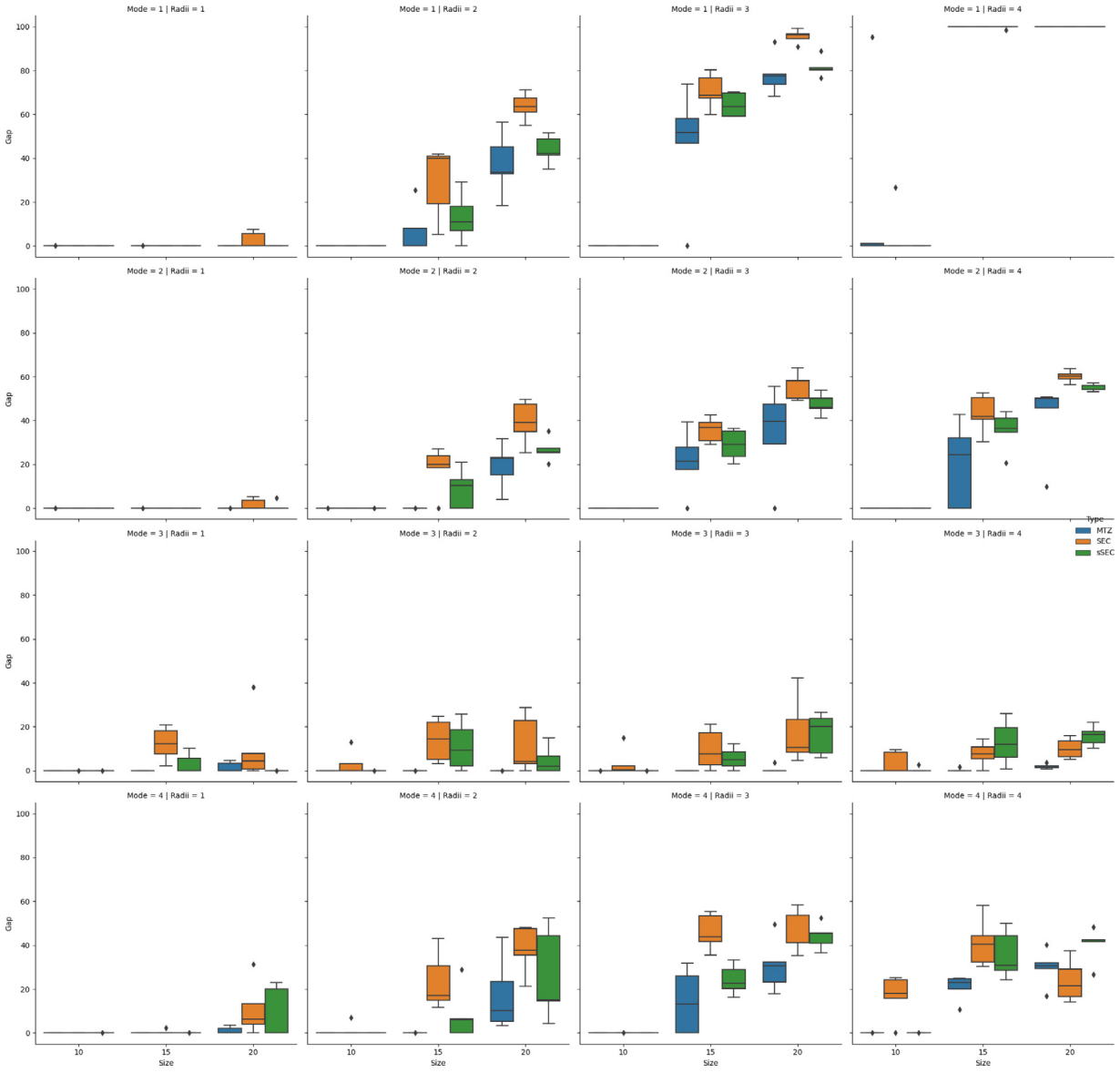


Fig. 12. Final gap after 7200 seconds.

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.

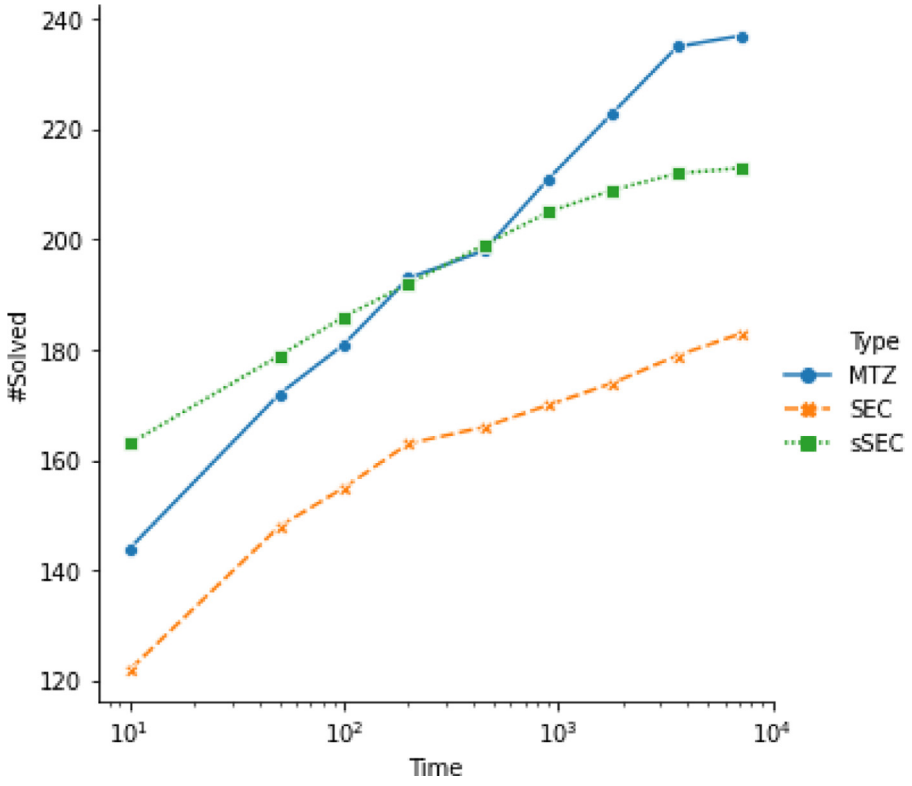


Fig. 13. Performance profile: Time vs #Solved.

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

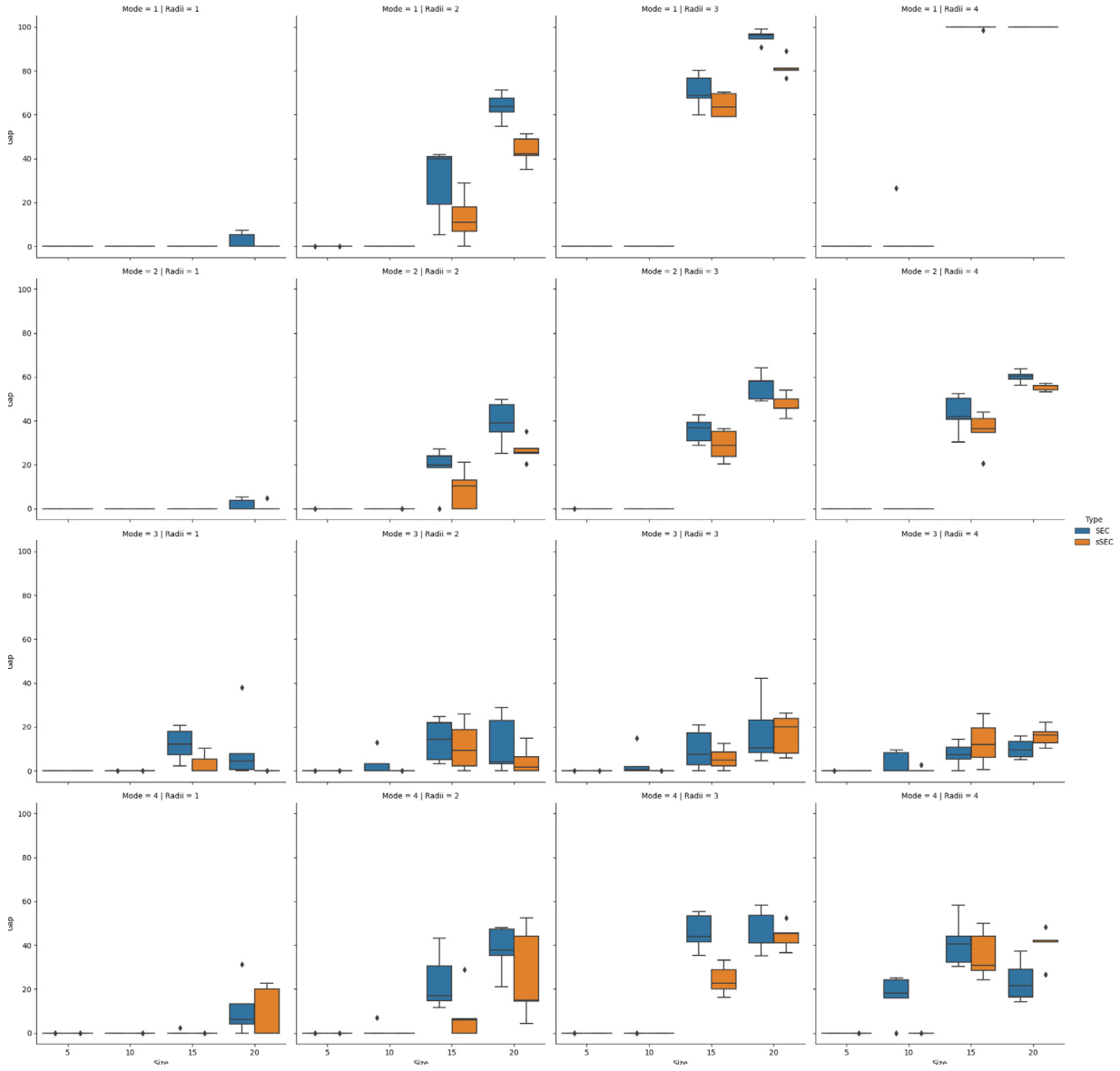


Fig. 14. Number of SEC added in the execution time.

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.