Expert Systems With Applications 195 (2022) 116604

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A multi-agent approach to the truck multi-drone routing problem

Jose Miguel Leon-Blanco [a,*], P.L. Gonzalez-R [a], Jose L. Andrade-Pineda [b], D. Canca [a], M. Calle [a]

[a] University of Seville, School of Engineering, Department of Industrial Engineering and Management Science I, C Descubrimientos s/n, 41092 Seville, Spain
[b] Robotics, Vision & Control Group, University of Seville, School of Engineering, C° Descubrimientos s/n, 41092 Seville, Spain

ARTICLE INFO

Keywords:
Unmanned aerial vehicle
Drone
Multi-agent system
Vehicle routing problem
Traveling salesman problem

ABSTRACT

In this work, we address the Truck-multi-Drone Team Logistics Problem (TmDTL), devoted to visit a set of points with a truck helped by a team of unmanned aerial vehicles (UAVs) or drones in the minimum time, starting at a certain location and ending at a different one. It is an enhanced version of the multiple Flying Sidekicks Traveling Salesman Problem (mFSTSP) presented in Murray and Raj (2020) wherein drones are allowed to visit several customers per trip.

In order to cope with large instances of the complex TmDTL, we have developed a novel agent-based method where agents represent the points that are going to be visited by vehicles. Agents evolve by means of movement inside a grid (locations vs. vehicles) according to a set of rules in the seek of better objective function values. Each agent needs to explore only a fraction of the complete problem, sharing its progress with the rest of the agents which are coordinated by one central agent which helps to maintain an asynchronous memory of solutions – e.g. on the control of the mechanism to escape from local minima.

Our agent-based approach is firstly tested using the largest instances of the single TDTL problem reported in the literature, which additionally serves as upper bounds to the TmDTL problem. Secondly, we have solved instances up to 500 locations with up to 6 drones in the fleet. Thirdly, we have tested the behavior of our approach in 500 locations problems with up to 8 drones in order to test the fleet size sensitivity.

Our experiments demonstrate the ability of the proposed agent-based system to obtain good quality solutions for complex optimization problems that arise. Further, the abstraction in solutions coding applied makes the agent-based approach scalable and flexible enough to be applied to a wide range of other optimization problems.

1. Introduction

There are a variety of practical applications where the use of drones promises an improved service, taking advantage of their travel speeds and reduced costs (Campbell et al., 2017). The drone's ability to travel directly between two points of interest is leading more and more to the conceptualization of new working models in certain logistics scenarios, mostly assuming that a moving truck acts as a base, which allows the enlargement of the action radius of the drone's operation.

In this paper, we assume a set of locations to which we need either deliver lightweight relief items or visit for an intelligence, surveillance and reconnaissance (ISR) mission. We extend the seminal Truck Drone Team Logistics (TDTL) model in (Gonzalez-R et al., 2020) to consider a fleet of multiple Unmanned Aerial Vehicles (UAVs) or drones along with a truck. Since the optimization problem that arises has to find the optimal routes for the vehicles in the system to achieve a given objective –e.g. minimizing the total mission time-, the applications under study are within the category "routing for a set of locations" (Otto et al., 2018). In contrast to other works that assume the truck to act solely as a mothership where drones change batteries and payload (Poikonen & Golden, 2020b; Salama & Srinivas, 2020; Wang et al., 2017) we consider that the truck plays an active role in the delivery process since it can serve the locations as in (Ha et al., 2018; Murray & Chu, 2015).

There are other studies on the multi-drone field (Luo et al., 2021; Murray & Raj, 2020; Poikonen & Golden, 2020a; Schermer et al., 2019), but they are typically focused on addressing small to medium size instances. Differently, our main contribution here is gaining the ability to solve large instances in the assumption that multiple customer visits are allowed for drones, which according to (Chung et al., 2020) is a more general approach. While this feature is also in Poikonen & Golden (2020a), they did not apply the drone sortie policy that we incorporate in this paper. Further flexibility is attained by allowing drones to come back to the truck at a different location than the one they were launched from.

* Corresponding author.
E-mail address: migueleon@us.es (J.M. Leon-Blanco).

https://doi.org/10.1016/j.eswa.2022.116604
Received 6 April 2021; Received in revised form 17 November 2021; Accepted 20 January 2022
Available online 31 January 2022

2. Related work

The literature on coordinating the logistics operations using a truck-drone combination focuses predominantly on extending classical routing problems – a variant of the traveling salesman problem with drones (TSP-D) (Agatz et al., 2018; Bouman et al., 2017; Ha et al., 2018; Jeong et al., 2019; Mathew et al., 2015; Murray & Chu, 2015; Roberti & Ruthmair, 2021; Vásquez et al., 2021), and a generalization of vehicle routing problems to include drones (VRP-D) (Poikonen et al., 2017; Sacramento et al., 2019; Schermer et al., 2018; Wang et al., 2017; Wang & Sheu, 2019). For a comprehensive recent review of the current research in this area, the reader is referred to the works by Rojas Viloria et al. (2020), Chung et al. (2020), Macrina et al. (2020) and Moshref-Javadi & Winkenbach (2021). As claimed in our seminal work (Gonzalez-R et al., 2020), we cover some of the gaps identified in the literature. In particular, we stated the TDTL for applications where a drone trip (every trip comprises launching and visiting a typically small number of locations and a landing in the ground vehicle at the next rendezvous point) is capable of serving several locations. This differs from the commonly accepted hypothesis in last-mile delivery literature which considers that only one single location is visited at each drone's trip – except for only a few works, see e.g. (Karak & Abdelghany, 2019; Luo et al., 2021; Murray & Raj, 2020; Poikonen & Golden, 2020a; Salama & Srinivas, 2020; Wang et al., 2019). However, notice that this limitation is not necessarily applicable; for instance, current drone technology allows for picking-up small items –e.g. vaccines, water purification tablets, or medicines– and delivering them one-at-a-time following a sequence of visits during the same drone's trip. In our approach to the problem: (i) every location stands for a customer that can be served either by drone or by truck; (ii) the truck stops occur at certain customer sites (which are not predefined) and hence, drones are only allowed to merge with the ground vehicle at these locations; (iii) Once landed, the drone always gains a fully-charged battery and would then be ready to start a new trip or stay at the truck until the truck carries it to a new serving area.

We consider all the locations are open to be visited either by drone or by truck, therefore all of them are potential points for both stop and rendezvous. (Luo et al., 2017; Wang et al., 2019) assume that both the truck and the drone can serve the locations, although both works limit the locations wherein the synchronization among trucks and drones can occur: (Wang et al., 2019) schedules it at the docking hubs –i.e. spacious areas to make a controlled landing– whereas (Luo et al., 2017) schedules the rendezvous points at truck stops among a preselected subset of candidate locations. This is a simplification that also appears in (Sacramento et al., 2019) and (Karak & Abdelghany, 2019). (Sacramento et al., 2019) consider that some points are to be served by the truck owing to the weight of the load, and limits the synchronization at truck stops among a preselected subset of candidate locations. (Schermer et al., 2019) have recently considered the possibility of launching and retrieving drones along a route – i.e. at discrete locations others than the customer locations – and has reported experimental results for their VNS/Tabu search heuristic applied on up to 50 customer instances: they claimed the potential of a reduced makespan and a higher utilization of drones. Some other authors (Caggiani et al., 2017; Carlsson & Song, 2018) adopted a continuous approach to allow the drones to meet the truck at any point of the route between two locations. Notwithstanding, we use the common assumption that synchronizations only occur while the truck stops at one of the discrete locations. Apart from this, the flexibility of our approach resides in that we do not fix a priori which of the points are going to be visited by the truck and which of them are going to be visited by drones, in contrast to completely prefixed routes for the truck (Othman et al., 2017) or partially prefixed routes (Luo et al., 2017; Mathew et al., 2015; Murray & Chu, 2015). Besides, while Sacramento et al. (2019) prohibits the truck from waiting for the drone at the same location it was launched, in our approach, we get the same result differently: owing to the active role-play of the truck and thinking of visiting all the locations as soon as possible, launching a drone and rendezvousing the same drone at the same location is of no interest. Indeed, for the addressed problem, the synchronization issue is crucial since it turns the problem into finding the chain of customers to be served by each vehicle, while specifying the sites where the battery swaps will take place.

A variety of single-truck multi-UAV works have been published in the last years. (Chang & Lee, 2018; Ferrandez et al., 2016; Moshref-Javadi & Lee, 2017) consider a system in which the truck deploys multiple drones from distributed launch sites along the truck's route. In this case, the drones return to the truck before the truck departs to its next destination. Clustering heuristics have been developed, such that the truck is routed to each cluster and nearby customers are served using UAVs. Conversely, others (Murray & Raj, 2020; Yoon, 2018) consider a single truck that may launch multiple UAVs, with the UAVs returning to the truck at a different location. (Yoon, 2018) provides a MILP formulation, which is tested on instances with up to 10 customers. (Murray & Raj, 2020) states the multiple flying sidekicks traveling salesman problem (mFSTSP) using an arbitrary number of heterogeneous UAVs that may be deployed from the depot or the delivery truck, although they adopt an overly-constrained approach for our target problem, particularly due to being limited to the one-customer per fly case. Most papers addressing the truck-drone cooperative system assume that during each trip a drone can only visit one customer, except for (Ham, 2018; Luo et al., 2017) for the flying sidekick traveling salesman problem (FSTSP) and (Cheng et al., 2018) for the parallel drone scheduling traveling salesman problem (PDSTSP) (the latter involving just drone deliveries, with no truck at all). (Ham, 2018) is a generalization of the multiple drones, multiple trucks and multiple depots case, and (Cheng et al., 2018) a pure drone-delivery scenario where no truck intervenes at all, thereby being far from our target problem scope. In (Moshref-Javadi, Hemmati, et al., 2020; Moshref-Javadi, Lee, et al., 2020) computational studies are presented where the movements of the truck and UAVs are synchronized. In these problems, the truck stops at a customer location and can launch one or more UAVs. Each UAVs delivers only one parcel to one of the clients and meets the truck at a client location different from the first one. In (Moshref-Javadi, Hemmati, et al., 2020) the authors propose a Mixed Integer Linear Programming (MILP) model, demonstrating on four sets of problems with up to 101 customers that this model can obtain considerable waiting time savings compared to the truck-only model.

Considering solution approaches, a variety of methods have been developed to address multidrone-truck combined operations (see (Chung et al., 2020)). (Phan et al., 2018) developed the TSP with multiple drones (TSP-mD) that was solved by an adaptive large neighbourhood search heuristic (ALNS), while (Murray & Raj, 2020) solved the mFSTSP by a three-phased heuristic solution approach. Furthermore, they state a very constrained model whose resolution is reduced to a medium size set of customers. In (Moshref-Javadi & Lee, 2020) the problem analysed in (Moshref-Javadi, Hemmati, et al., 2020) is addressed by a hybrid Tabu Search-Simulated Annealing algorithm. Among several parameter values, the number of drones and the drone to truck speed ratio affect the results more significantly. (Wang et al., 2017) proves several bounds to the potential savings in the total tour time that can be achieved for several scenarios with multiple trucks, multiple UAVs, as well as equal and unequal UAVs and truck speeds.

Recent works by (Poikonen & Golden, 2020a) and (Luo et al., 2021) exhibit certain similarities with our TmDTL problem. (Poikonen & Golden, 2020a) presented an interesting TSP-D with multiple drones with consideration of adjustable speeds and battery consumption rate as a function of the payload, although they assumed the truck serves solely as a mobile depot (drone primary) which does not deliver packages to customers. (Luo et al., 2021) includes also the fact that the battery endurance depends not only on the flight time, but on the self-weight of the drone and the total weight of the carried packages. They solve their model using a heuristic approach and report its performance only for a

small fleet of two drones cooperating with the truck. In contrast, we aim our research at getting an efficient resolution method for coping with a big number of targeted points which will typically require a higher fleet size.

The TmDTL problem we consider, increases the complexity of the TDTL problem as it increases the number of UAVs carried by the ground vehicle. Thus, since the TDTL problem is an NP-hard one, our problem is also NP-hard. For a study of the computational complexity of truck and drone problems, see (Boysen et al., 2018) where the truck route is prefixed and, for a more similar problem but only for one drone, the TDTL problem, see (Gonzalez-R et al., 2020). In this paper, we focus on the application of a multi-agent system (MAS) to the problem of multiple drone TDTL. According to the definition in (Barbati et al., 2012), in which there is no difference between MAS and Agent-based models (ABMs), these systems consitst of "elements (agents) characterized by some attributes, which interact with each other through the definition of appropriate rules in a given environment". MAS are sometimes preferred to the more common heuristics for addressing challenging optimization problems. The intelligence of agents will make them choose between different methods to obtain the best global improvement, as each agent has information about its neighbourhood in the solution space. The knowledge about movements to nearby positions in the sequence, the distributed nature of MASs, and the information sharing between agents make this approach scalable and suitable for different problem configurations, sizes, and complexities, see (Barbati et al., 2012; Kulkarni & Tai, 2010; Leitão et al., 2013). Notwithstanding, (Leitão et al., 2013) and (Pěchouček & Mařík, 2008) also highlight issues related to the application of MAS in the industry, such as the difficulty in evaluating the return on investment, lack of standards, skilled personnel to apply these models, knowledge about their potential, success cases, large scale industrial MAS applications and tools.

MASs have been used to tackle routing and logistics problems. As stated in (Thangiah et al., 2001), an intelligent agent architecture gains the advantage of a distributed way of addressing VRPs rather than the centralized way of most metaheuristics. Like the former authors, (Barbucha & Jedrzejovicz, 2007) also takes advantage of the intelligence and cooperation of agents to improve a population-based metaheuristic for solving instances of the vehicle routing problem with time windows (VRPTW). (Kaul, 2018) improves an Ant Colony Optimization ACO by using an agent based metaheuristic applied to the last mile delivery problem in which vehicles, orders and clients are modeled as agents. For instance, (Davidsson et al., 2005; Gath et al., 2015) have taken advantage of this feature when approaching transportation logistics problems. (Gath et al., 2015) simulate and optimize logistics problems using vehicle agents and order agents: order agents look for a transportation service provider and vehicle agents try to maximize the number of shipments while satisfying constraints, and so they negotiate with other agents. We refer to (Gath et al., 2015) for a review of MAS applications to transport and logistics problems. In most of these applications, the transportation actors (e.g. vehicles or clients) perceive individual information, and make individual decisions, while being situated in and interacting with an environment (e.g. the transportation network or the information sources) on which they have partial and incomplete information.

Routing related optimization problems have been faced by the hybridization of agent systems and metaheuristics, as in (Talukdar & Ramesh, 1992; Thangiah et al., 2001) for the VRP, in (Zeddini et al., 2008) for the Dynamic VRP or in (Kalina et al., 2015; Lopes Silva et al., 2019) for the VRPTW (Vehicle Routing Problem with Time Windows). The common approach is to think of vehicles as agents (Barbucha, 2012; Kalina et al., 2015; Lopes Silva et al., 2019; Thangiah et al., 2001; Zeddini et al., 2008). Each agent uses a meta-heuristic, like evolutionary methods in (Dazhi & Shixin, 2010) or an iterated local search (ILS) in (Lopes Silva et al., 2019). The latter concluded that an improvement in the quality of solutions was attained by increasing the number of agents and the learning capacity from their interaction. This interaction or cooperation between agents can be synchronous or asynchronous (Barbucha, 2012). The asynchronous type of communication implies the use of a central pool of solutions or a central agent coordinating communications (Talukdar et al., 1983) or (Talukdar & Ramesh, 1992). Another way of implementing this asynchronous collaboration between agents uses an auction mechanism in which an auctioneer agent gathers client data, distributes them to vehicle agents, and receives bids from those agents in order to choose the best one and assign an agent to the appropriate client (Thangiah et al., 2001; Zeddini et al., 2008). Synchronous communications are of two main types: The first one, with optimization agents running in parallel, (Barbucha, 2012), and the second one, with a reinforced learning mechanism, (Alipour et al., 2018; Lopes Silva et al., 2019).

Regarding drones/UAVs related problems, the use of MAS has been mainly focused on the real-time coordination of fleets or swarms of UAVs considering each one as an intelligent agent, and has led to a myriad of MAS approaches concerning military missions control tasks – see (Baxter et al., 2008) – and the cooperation of logistic service providers (Hasan & Niyogi, 2020) but also in the planning of UAVs' routes – see (Semsch et al., 2009) – where our research is focused. An application to the Drone TSP (DSTSP) is developed in (Houseknecht, 2019) using ant colony optimization (ACO) and trying to maximize drone's usage. More similarities arise in the approach in (Khalid & Chankov, 2020; Kim & Matson, 2017), when addressing a delivery problem with drones in collaboration with public transport buses. Their ground vehicle and the multi-UAV optimization problem is different from ours, since we consider that drones take off from the depot or the truck at one point and land at a different one.

There have been many efforts to build architectures and protocols for agents' communication such as FIPA – The Foundation of Intelligent Physical Agents (FIPA - The Foundation of Intelligent Physical Agents, n. d.), KQML – Knowledge Querying and Manipulation Language (KQML, 1993) and others mentioned in (Dickinson, 1997). Unfortunately, many of these efforts are discontinued nowadays. There are also frameworks for Agent-Based simulation and modelling (Mualla et al., 2018) such as AirSim (Shah et al., 2018), Flame (Kiran et al., 2010), Gazebo (Nathan Koenig, 2004), JADE (Bellifemine et al., 2001), JaSIM (Galland et al., 2009), NetLogo (Wilensky, 1999), Repast Simphony (North et al., 2005) or OpenABM (Janssen et al., 2008). For a more comprehensive review of Agent-Based Modelling and simulation, the works of Allan (2009) and Lopes Silva et al. (2018) are of interest. While many of those frameworks are Java-based, there are a few Python-based such as SPADE – Smart Python multi-Agent Development Environment (Palanca & Alemany, 2017), a Python version of Agent Evolution (AgE) (Faber et al., 2012; Kazirod & Knapik, 2016) and MESA (Kazil et al., 2020), an agent-based modelling framework. At the time we began this research MESA was easier and better documented than SPADE, hence that was our choice. This environment is heavily based on NetLogo (Wilensky, 1999).

In this paper, we contribute with a MAS to approximately solve the challenging TmDTL. Our approach is novel with respect to other routing problems, since we have an abstract use of multiagent systems assigning each agent to a location instead of a vehicle, an order or all of them (vehicles + orders + clients) as in (Kaul, 2018). We have followed the old cellular automata scheme in which agents move in a lattice or grid environment to generate neighbourhoods, precedences and feasibility (Macal & North, 2010) – i.e. avoiding locations to be served by the truck and later by another drone. Each agent has a set of rules based on which it takes the best decision to move inside the grid while trying to obtain the best objective function improvement. Agents are coordinated by one central agent or manager which maintains an asynchronous memory of solutions and assists the escape from local minima.

The main contributions of our paper are:

- We present an improvement over the TDTL problem described in (Gonzalez-R et al., 2020) in which a truck gets help from a drone in order to visit a set of points in the minimum time. The difference with the problem in (Gonzalez-R et al., 2020) lies in the number of drones carried

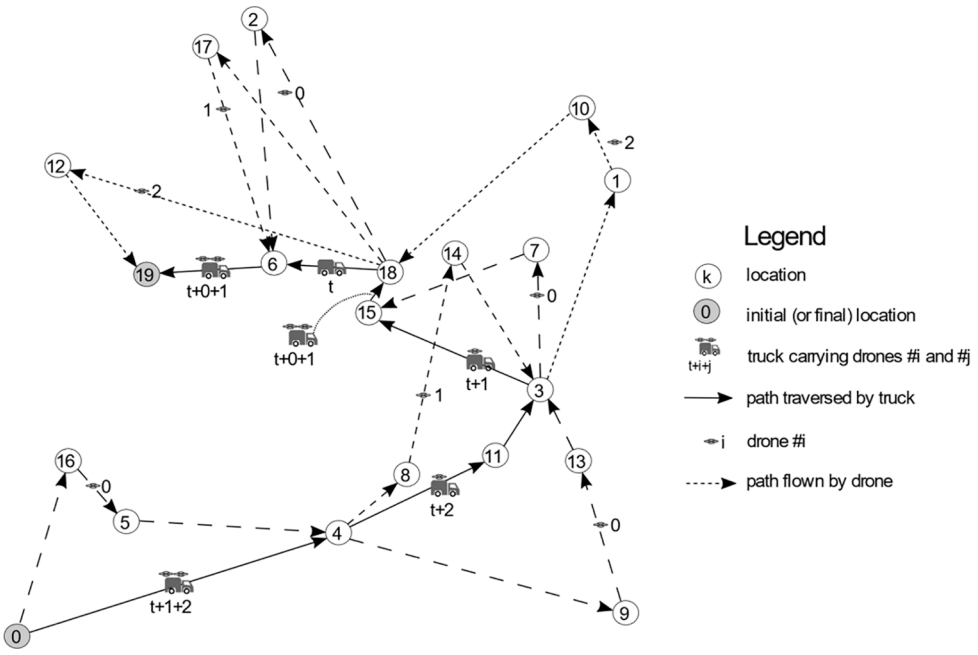

Fig. 1. Example solution for a problem with 20 locations and a truck helped by three drones.

by the truck, the solution approach and the size of the problems we are dealing with. Regarding the multiple flying sidekick TSP (mFSTSP) described in (Murray & Raj, 2020) the differences are twofold. We reduce complexity by a) supposing that all drones are identical, b) supposing the time needed for landing, recharging, take-off is negligible and c) supposing a linear endurance model for the drones. We increase complexity by letting drones visit more than one point in each flight; that is, carry more than one parcel.

- We propose a novel agent-based approach for the truck-drones routing problem in which agents represent points or clients instead of orders or vehicles. Our solution is represented in a two-dimensional grid. Agents try to move on this grid to positions that lead to an improvement in the total mission time. This abstraction leads to faster computing the fitness, since each agent only computes the improvement in quality caused by its own movement. This abstraction also leads to a more flexible solution coding as it can be applied to problems with more than one truck. This is an ongoing extension of this work. We have implemented this approach by using the MESA framework in Python.
- We compare the resolution of the instances with those presented in (Gonzalez-R et al., 2020) in a set of 86 instances (available in [dataset] (Bouman et al., 2020)) with sizes varying between 50 and 500 locations. The agent-based model obtains good results in larger instances.

The rest of this paper is organized as follows: Section 3 provides the problem description and the main hypotheses in the TmDTL. Section 4 describes the developed MAS approach and Section 5 presents a computational study and the numerical results. Finally, Section 6 concludes the paper.

3. Problem description

The Truck-multi-Drone Team Logistics Problem (TmDTL) can be formally described as follows. A complete directed graph $G = (N, A)$, which represents a set of locations distributed in a wide area are to be served by a truck and multiple identical UAVs, is given. The vertex set N is defined as $N = \{o, 1, \cdots, n-2, e\}$, where o represents the departure node (a single truck, equipped with multiple drones, is initially located at node o) and e corresponds to the destination node, where all the vehicles must finally meet. Nodes $\{1, \cdots, n-2\}$ represent the set of customers locations to be served (i.e., visiting and collecting data typically in ISR missions, as in (Hu et al., 2019) or delivering relief items in emergency situations or goods as in (Murray & Raj, 2020)). Every location needs to be served once by, at least, one of the vehicles, the truck, one of the drones or could be a rendezvous point where the truck and one or more drones meet. The arc set A is defined as $A = \{(i,j) : i,j \in N\}$. The truck moves at a speed s_{gv} while drones flight speed is supposed to be s_{av} . A UAV launched from the truck can sequentially visit and serve multiple locations in a single flight route. Due to the constraint of limited battery capacities, a UAV can only travel a quite limited distance in one route (e.g., 7 km). We suppose that all the drones have the same battery endurance, of value Q, measured in time units. The truck is used to transport, launch, and recycle the UAVs, serving as a mobile platform. Since a truck can typically travel over 500 km without refuelling, without loss of generality, we assume that the vehicle can travel unlimited distances within the region. The drones can be launched from and return back to the truck at customer locations, and they cannot take off and then land at the same location. The drones and the truck must coordinate when they arrive at the same location. If the drones arrive earlier than the truck or other drones, they wait hovering until the truck arrives. See location 3 in Fig. 1. Likewise, the truck must wait for the last drone to arrive at that location. This supposes a difficulty in the total mission time and the drone endurance calculations. The goal of the TmDTL is to find a set of routes, starting at o and ending at n, with minimum completion time, serving all customers either by truck or drone, allowing multiple visits per drone's flight, without imposing any restriction to the truck route and determining the number and location of the synchronization nodes.

In Fig. 1 we show an example solution, representing 20 locations and the routes traversed by truck, carrying drones or not, depicted with continuous lines, and the flights of drones, drawn by dashed lines. Here we use different dashed types to distinguish between paths flown by different drones. Thus, truck departs from depot 0 carrying drones 1 and 2 and visits location 4. Drone 0 takes off from the depot and after visiting locations 16 and 5 meets the truck at location 4. From that location, the truck, carrying drone #2, departs to location 11 while drones #0 and #1 take off. After visiting location 11, at location 3, the truck meets drone #0, which has visited locations 9 and 13 and drone #1, which has visited locations 8 and 14, and so on.

While there are similarities between the problem in our research and

those addressed in (Murray & Raj, 2020) and in (Salama & Srinivas, 2020), it is necessary to highlight important differences between our research and those works. (Salama & Srinivas, 2020) contemplates a truck acting as a moving depot, carrying more than one drone. It moves near customer locations – a focal point – deploys drones that visit customer locations and waits until they complete their task. They state their approach maximizes drone utilisation and therefore minimises the necessary number of drones. In our case, and in (Murray & Raj, 2020), the truck continues the route once drones have taken off. But the work of (Murray & Raj, 2020) is different in that drones serve only one location in each flight. They also note a decrease in the reduction of the total mission time when increasing the number of UAVs in the fleet. This is on account of unproductive flight-time spent by the drones, which can be explained two-fold: (i) the truck roof may be in use by a different drone when a particular drone needs to land; and (ii) the drones have to wait for the truck to arrive at the recovery location. In contrast with them, as stated in our seminal work (Gonzalez-R et al., 2020), we are concerned with a more general approach where each drone is capable of visiting one or more points in each flight before having to return to the truck to change its battery. Whereas this tends to diminish the effect of (i), we still can assume that a kind of saturation appears in the enhancement one can expect from increasing the fleet size.

With the help of the vehicle, the UAVs can serve locations distributed over a large region. The vehicle carrying multiple UAVs and enough UAV replacement batteries departs from the base and proceeds along an a priori unknown route. Along the route, we select a set of locations where the vehicle launches and/or retrieves UAVs. When a UAV returns, its battery is replaced on the vehicle. After visiting all the locations, the ground vehicle arrives at the ending location. Without loss of generality, the following assumptions are made:

- The coordinates on a 2D plane of every location are known to us a priori. The road network lying in the target region is modelled as a fully connected graph, so it is possible to go directly from every location to a different one.
- Both the ground vehicle and the UAVs travel at constant speeds, which are denoted as s_{gv} (ground vehicle speed) and s_{av} (air vehicle speed) respectively. It is assumed that $s_{av} = 2s_{gv}$. In general, in the literature $s_{av} > s_{gv}$, but this is not a requirement for our solution method. We will also assume that when drones arrive earlier than the truck to a rendezvous or meeting point, they wait for the truck hovering. Thus, when drones arrive at a location earlier than the truck, the waiting time also drains the battery charge.
- The time required by the UAV to serve a customer, the time needed for replacing a UAV's battery and other maintenance tasks are negligible. It is supposed that when the last vehicle arrives to a location, drones and truck start their next trip immediately
- We consider a homogeneous fleet of drones with the same battery endurance. Batteries drain uniformly depending on the distance travelled.
- The objective function consists of minimizing the total mission time needed for visiting all the locations. The vehicles start from an initial location and arrive at a different one. Each location must be visited at least once by one vehicle, and drones must respect their battery endurance.

The main difficulties derived from this problem configuration are manifold. First, it is essential to model as in other routing problems, the order in which locations are visited, which vehicle visits each location and, related to the former, which of the drones is flying each arc or is traveling on the truck. Other common constraints are battery endurance, the avoidance of both visiting twice a location or leaving locations unvisited. Also, cycles in drone flights or in truck travels must be avoided.

4. Agent-based approach

In this section, the agent-based proposed approach to solve the TmDTL problem is presented. We first describe the abstraction of mapping locations to agents, then the construction of the initial solution, and finally the set of actions governing the behaviour of each agent.

Table 1
Coding of possible meeting types.

Type	d2	d1	d0	Comments
0	0	0	0	Only truck visits that location
1	0	0	1	Drone number 0 meets truck
2	0	1	0	Drone number 1 meets truck
3	0	1	1	Drones 0 and 1 meet truck
4	1	0	0	Drone number 2 meets truck
5	1	0	1	Drones 0 and 2 meet truck
6	1	1	0	Drones 1 and 2 meet truck
7	1	1	1	All drones meet truck

As proposed in (Barbati et al., 2012) we have chosen a mediator architecture, since in large problems, our approach will work with a high number of agents. In such situations, mediator architectures show better effectiveness than autonomous agents' architectures in terms of computational simplicity and overall solution quality. Since the TmDTL is such a complex problem, we have chosen to keep the method as simple as possible, keeping in mind the agent's behaviour but trying to avoid an additional complexity.

To address the difficulties found in this type of drone routing problem, we have modelled solutions as a grid where agents are placed, thus specifying precedencies between locations' visits. In addition to the position on the grid, agents have an attribute to model, at locations visited by truck, if it is accompanied there by zero, one or more drones. The allowed positions of agents in the grid will guarantee that locations are visited once and only once by a drone or by the truck (alone or carrying/meeting drones). Other constraints such as battery endurance, or cycles in routes will be considered when computing the mission time.

4.1. Locations as agents

We propose a cooperative asynchronous agent system as classified in (Barbucha & Jedrzejovicz, 2007). We use a manager agent acting as a mediator between all agents. The rest of the agents represent the locations that are going to be visited, including the initial and final ones. This is a novel approach in contrast whith those that suppose vehicles or even orders as agents. If the sequence contains n locations, n agents will be created apart from the manager. These agents try to improve the current solution by moving around a finite bi-dimensional *grid* or lattice being similar to the classical simulations of social groups in (Schelling, 1958). They also follow the beliefs, desires, intentions (BDI) model; first deciding what to do in order to improve the solution quality and then acting in consequence.

When the truck arrives at a location, it can meet drones and replace their batteries. These meetings may include the truck and zero or more drones. We have added an attribute to agents to model the type of meeting. In the case that the location represented by an agent is visited by the truck, the type of meeting indicates which drone or drones meet the truck. If the location is visited only by a drone, the meeting attribute is empty. The meeting attribute uses a binary coding to represent which drone or drones meet the truck: $meeting = d_{ndr-1}d_{ndr-2}d_{ndr-3}\cdots d_0$. For

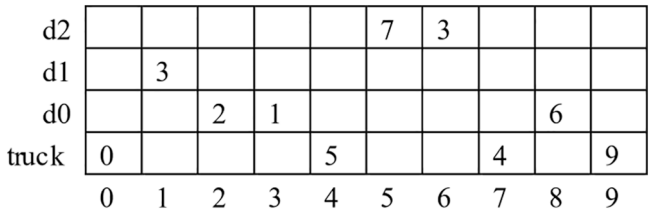

Fig. 2. A grid representing a 10 location + 3 drones problem.

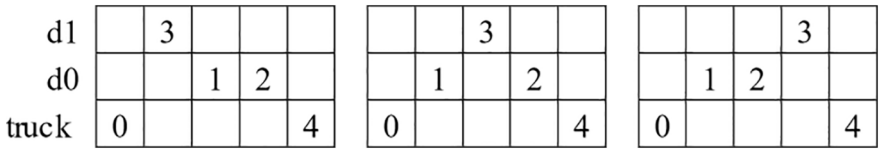

Fig. 3. Equivalent solutions.

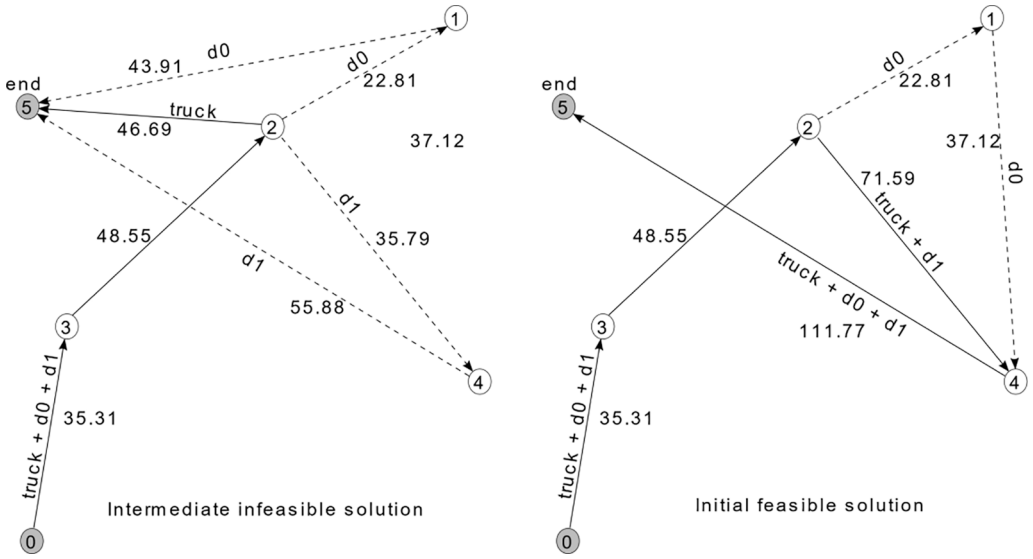

Fig. 4. Initial solution.

instance, in the case of a three drones' mission, possible meeting values are shown in Table 1. As, in our model, the time needed for battery change and other drone setup tasks is supposed negligible, we assume that when the last vehicle arrives at a meeting location, drones and truck start their next trip immediately.

Agents can move in a bi-dimensional grid (see for illustration purposes the Fig. 2). The horizontal grid dimension is the number of agents and the vertical one corresponds to the number of vehicles. Each of the columns contains one and only one agent representing one of the locations to be visited and which vehicle visits it. The bottom row represents the sequence of locations visited by the truck. The meeting attribute of each agent models which drones are with the truck in that location. It also helps in modeling drones' landing and taking off locations. The rest of the rows represent the sequence of locations visited by each drone. The only agents not moving around the grid are the first one and the last one, corresponding to the initial and final locations.

For instance, in a 10 locations problem (including the initial and final one) in which 3 drones are used with the truck to visit all locations, the grid is represented in Fig. 2.

The rows indicate the order in which each vehicle visits the locations. When the bottom row includes an agent, it is needed to know the meeting attribute of the agent to see whether a drone has landed on the truck at that location. In the first and last locations, the depots, all drones are in the truck. To know where a drone took off, it is needed to know the last location where it was in the truck, reading the meeting attribute. Thus, in Fig. 2 drone 0, after starting at location 0, visits location 2, and then location 1. Drone 2 visits location 7 before visiting location 3. This coding scheme sometimes leads to equivalent solutions with different grid distributions as shown in Fig. 3, for a problem with 5 locations.

Although these solutions are equivalent, they are not in the same neighbourhood, since the possible agents' movements avoid these solutions to be directly interchangeable.

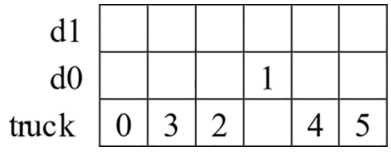

Fig. 5. Grid of complete initial feasible solution.

4.2. Initial solution. Examples

Initial agents' positions are generated by the manager agent using a constructive procedure that seeks a feasible solution in terms of drones' battery endurance. Considering a problem with n agents and ndr drones, the grid cells are numbered from $(0,0)$ to $(ndr-1,n)$. Agent 0 is assigned to the mission starting location and it is placed at cell (0, 0) whereas the agent $ndr-1$ is assigned to the ending location and placed into the cell $(ndr-1,n)$. The construction procedure assigns a maximum of ndr nearest locations to drones, holding their endurance requirements, and assigning the last one to the truck, where it meets all of the drones. This procedure is repeated with the next ndr nearest locations till all locations are covered by drones or by the truck.

To illustrate the procedure of constructing an initial solution we will use a small problem with $n = 6$ locations and $ndr = 2$ drones. We will suppose a battery endurance of 72.93 s, and a drone speed of 2 m/s, double that of the truck's speed, 1 m/s. As an intermediate step, an infeasible solution in terms of endurance is shown in the first graph in Fig. 4. This figure shows the vehicles traversing each arc and the times needed. Drones are represented by dashed lines and solid lines represent the truck's movements. As drone 1 exceeds its endurance when travelling from location 2 to location 5, in order to repair feasibility, it is transported on the truck and the truck will also visit location 4, the farthest from the two nearest locations (1 and 4) to location 2. Drone 0 visits location 1 and eventually meets truck and drone 1 at location 4.

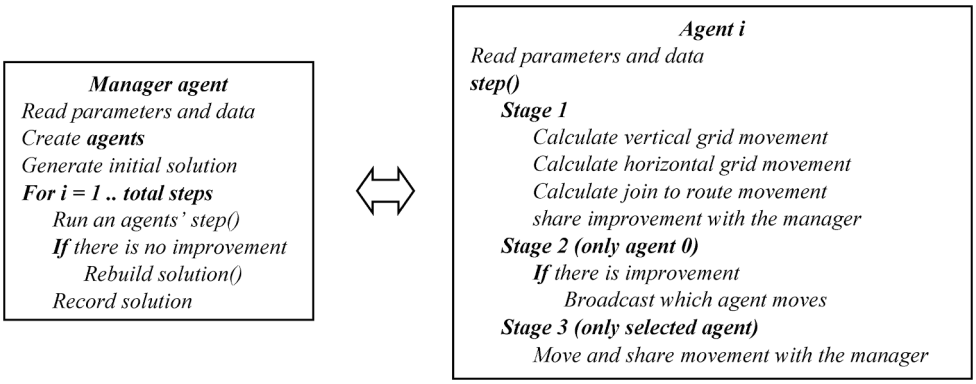

Fig. 6. Schema of multi-agent algorithm.

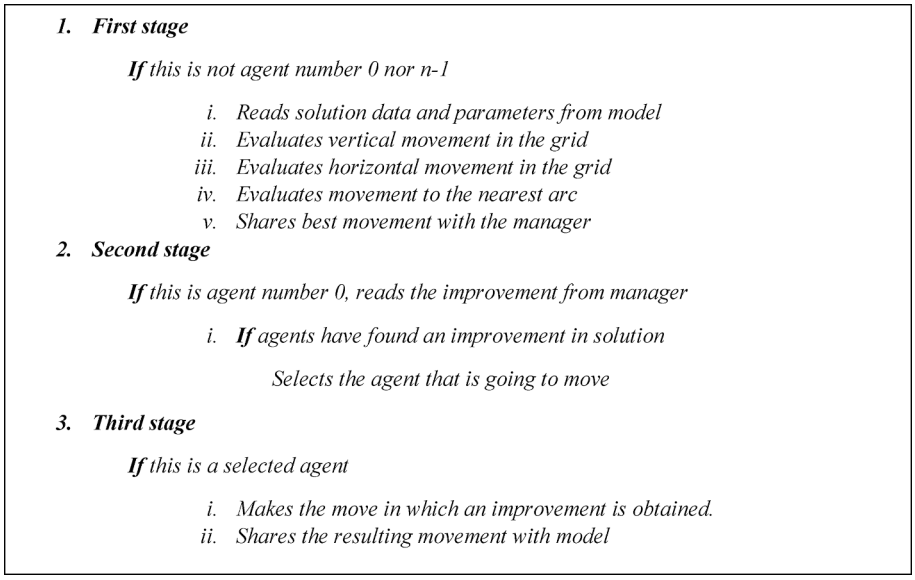

Fig. 7. Stages in agents' steps.

The resulting initial solution is shown in the second graph in Fig. 4 and the resulting grid for this new solution is illustrated in Fig. 5.

The mission time is calculated as the time needed in the truck's route, from the depot to the final location, but taking into account the synchronization processes in those locations where the truck acts as a base for the drones. In initial solutions, all locations visited by the truck are meeting points with all the drones. In this example, from Fig. 5, the arcs 0-3 and 3-2 are traversed at truck speed. Thus, $t(3) = t(0) + t_{gv03} = 35.31s$ and $t(2) = t(3) + t_{gv32} = 83.86s$ Then, drone #0 takes off from the truck at location 2, visits location 1 and then lands on the truck at location 4. The flight time is $t_{av}(4) = t_{av}(2,1) + t_{av}(1,4) = 59.93s$, the truck time is $t_{gv}(4) = t_{gv}(2,4) = 71,59s$ so drone #0 must wait hovering at location 4 till the truck arrives there and then it will meet the truck. There is no battery endurance problems since it is 72.93 s and the total flight time is the same as the ground vehicle time, 71.59 s. So, $t(4) = t(2) + t_{gv}(2,4) = 155.45s$. Finally, the total mission time is $t(5) = t(4) + 111.77 = 267.22s$. This initial solution is always feasible due to the construction procedure that has been followed.

4.3. Agents' behaviour

Once the initial solution has been generated by the manager agent, the solution is broadcasted to the rest of the agents that, to improve the solution quality, will try to move in their neighborhood positions on the grid. An schema of this behaviour is shown in Fig. 6.

The manager agent is the only one that maintains a pool of generated solutions, acting as a central asynchronous memory and setting the pace of the rest of the agents. After reading the problem data and generating a feasible initial solution, the manager performs the following set of tasks during a prefixed number of iterations:

1. Tells agents to run a step (best movement)

If there is no improvement in the agents' step (grid movement), generates a new solution

2. Stores and broadcasts the current solution

The rest of the agents evaluate the possible improvement in the solution quality by moving to neighbouring positions in the grid. In each step (grid movement), as explained before, they evaluate the improvement that would take place using three possible movements, vertical, horizontal and to the nearest arc (joining a nearby route). Next, they share the best improvement with the manager. We call this the **first stage** of the step. Then, in a **second stage**, agent 0 selects the agent that will move and eventually, in a **third stage**, the selected agent makes the movement, possibly involving other agents in that movement, and shares the result with the manager. These three stages of agents' steps are summarized in Fig. 7:

In what follows we describe in detail the possible movement of agents. We must highlight that, agents 0 and $n-1$ do not perform any kind of movements as they represent the initial and final locations. Agent 0 will be used in the second stage to select which of the other agents will move.

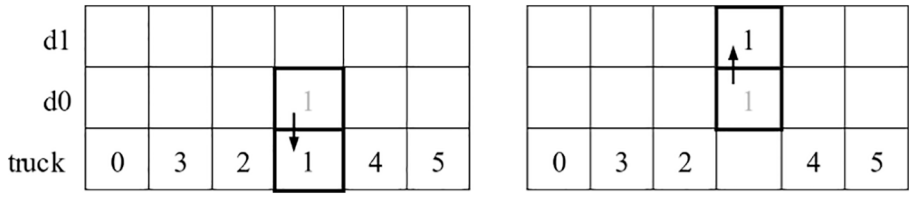

Fig. 8. yMove() example.

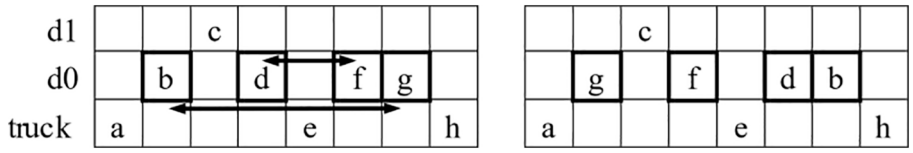

Fig. 9. Example of agent b's horizontal movement.

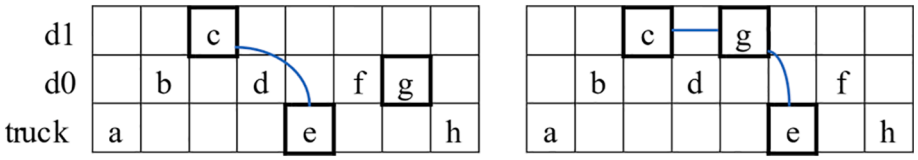

Fig. 10. Movement of agent g from its current location to a near arc in the grid.

4.3.1. *Vertical movement evaluation – **yMove**()*

The agent evaluates the improvement of the current sequence when moving vertically in the grid. This movement supposes a change in the vehicle that visits a location or in the type of meeting between truck and drones that takes place at that location. To reduce the computation time, the agent only evaluates the change that this movement implies in the mission time. These variations are a result of changes in some arcs of the current solution. To take all possible modifications into account, the sub-sequence of locations between the previous complete meeting and the next complete meeting, both included, is calculated. For example, if agent 1 is going to move, the sub-sequence that includes all the arcs affected by that movement spans from meeting at location 2 to meeting at location 4 (see Fig. 8). Only arcs 2-1, 1-4 and 2-4 are considered. This policy saves a long computation effort in larger problem instances.

4.3.2. *Horizontal movement evaluation – 2**opt**()*

In this movement, searching for a solution improvement, the agent changes its position with rear agents in the sequence. We have called this procedure 2*opt*() due to its similarities with the algorithm by (Croes, 1958). Since it is a horizontal movement, if the agent represents a truck visited location, it will try a change with the next truck visited location. If it represents a drone visited location, it will change position with agents/locations visited by the same drone in the same flight. If the movement goes further towards the immediate position in the sequence, intermediate agents will also change their positions (see example in Fig. 9).

A parameter *max_r* is used to limit this movement. To make this parameter less sensitive to changes in problem size, the maximum number of positions an agent can move in this horizontal movement, which we will call *radius*, has been modeled as a function of the problem size (*WIDTH*): $radius = WIDTH//max\_r$. This limit will reduce an excess in computation time that, according to our pilot tests, does not provide further improvement in the objective function quality.

In our research, this type of movement only obtains improvements occasionally because the swap between locations visited by the truck implies rebuilding drones' flights, which usually worsens the solution's quality. Furthermore, the swap between locations visited by drones seldom improves the solution's quality as their routes are short (due to endurance limitations) visiting only two or three locations.

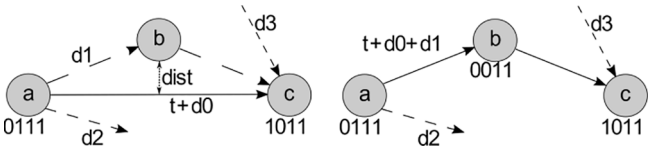

Fig. 11. Movement of agent b from its current location to a nearby arc.

4.3.3. *Movement to the nearest arc evaluation (joining a nearby route)– **node2arc**()*

In this case, agents try to improve the total mission time by moving to the intermediate position between two locations visited consecutively by the same vehicle. To this end, the agent needs to find the nearest arc in the nearest sequence or route. For instance, if agent *g* is going to move to arc $c-e$ (see Fig. 10):

If an agent movement changes a location from being visited by a drone to being visited by the truck, the meeting type must be re-calculated. A graphical example of this movement is shown in Fig. 11 (again drone movements are represented in dashed lines and truck movements in solid line), where agent b changes from being visited by a drone to being visited by the truck. As shown, the meeting type changes, and the movement over the grid is represented in Fig. 12 Note the change in the meeting code of agent b according to the contents of Table 1.

Here we show another example of an agent movement to the nearest arc. Agent 2 tests if there is a travel time improvement when location 2 changes to being visited by the truck in its route between locations 4 and 5. The two graphs in Fig. 13 represent vehicles, locations and distances between locations. The drone speed is supposed to be 2 m/s, the truck speed is 1 m/s and the battery endurance is 72.93 s (see also the corresponding movement in the grid in Fig. 14).

Initial situation: total mission time: 261.07 s (feasible). The nearest arc to location 2 is 4–5.

Modified situation: total mission time 277.78 s (non-feasible, penalty time: 57.71 s due to drone 0 flight 0–3-1–4).

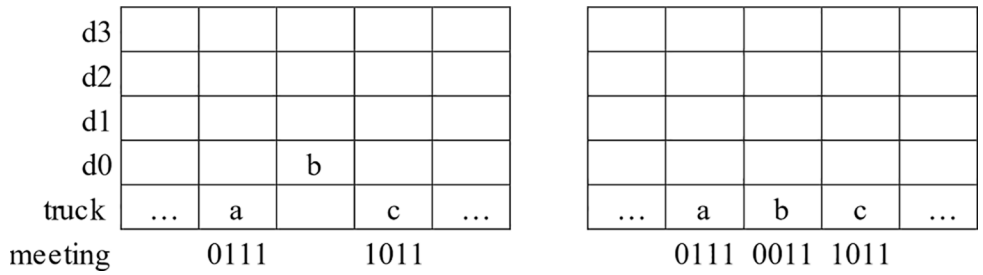

Fig. 12. Grid representation of the movement represented in Fig. 11.

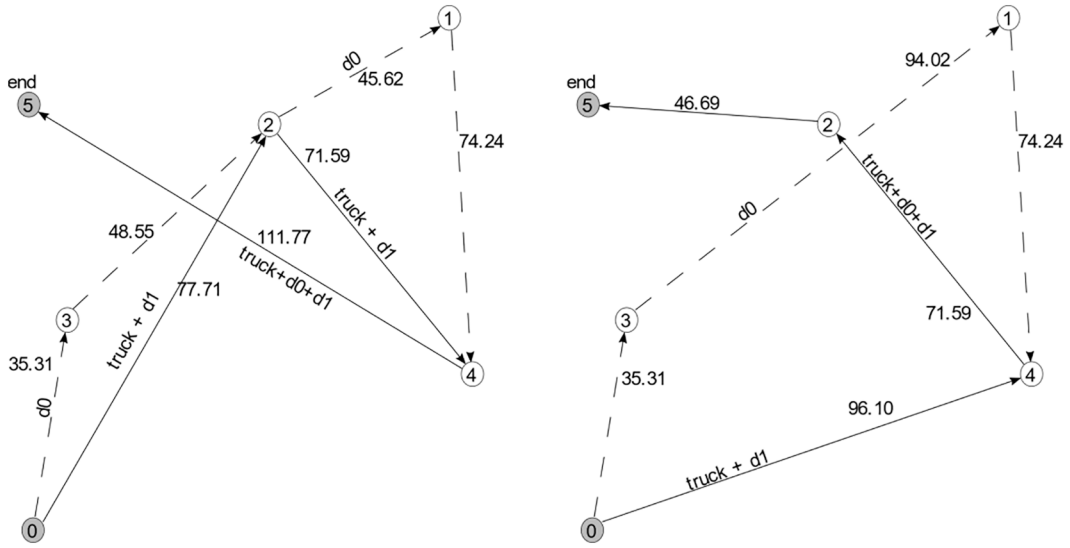

Fig. 13. Agent no. 2 location2arc() movement.

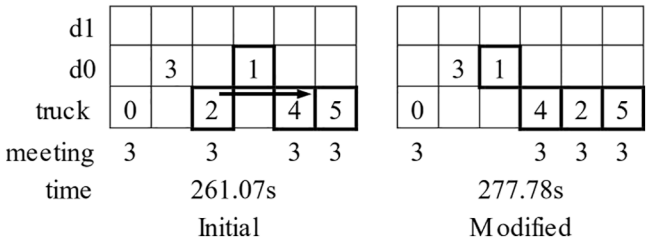

Fig. 14. Agent no. 2 location2arc() grid movement.

4.3.4. *Selection of the agent to move – select*()

In the **second stage** of each agent's step, the static agent 0 performs the task of choosing between other agents which one will move. If there is at least one agent that can improve the quality of the solution, agent 0 reads from the manager the list of agents that allow an improvement in the objective function solution together with the values of these improvements. Then, it assigns to each agent a probability proportional to the improvement that each agent allows. Eventually, it throws the dice to choose one of the agents using these probabilities. After having selected the index of the agent, agent 0 shares this information with the manager; in this way, in the third stage, all the agents will know which of them will move.

4.3.5. *Making the best movement*

In the **third stage** of each agent's step, the selected agent will make its best movement and will share it with the manager agent, which in turn will let all agents know the result of that movement.

4.3.6. *Sharing the resulting movement with the manager*

To share the movement with the manager agent, the agent communicates the improvement to the manager as well as the final position of the rest of the agents (since some of the movements imply not only the

1. Create a list rep_agents with identifiers of those agents that can be repositioned.
2. $iteration \coloneqq 0$
3. **While** $total\_time > model\_time$ *AND* $iterations \leq max\_its$
   a. $iteration \mathrel{+}= 1$
   b. Chose randomly n_{out} agents from rep_agents
   c. Extract those n_{out} agents from solution
   d. $solution \coloneqq rebuild(partial\_sol, extracted)$
   e. $total.time \coloneqq time\_calc(solution)$
4. Record solution data in manager

Fig. 15. ***rand_sol***() procedure.

1. **For** all indexes in *extracted*
   a. *min_time* := 10000.0 # a big initial time
   b. Try a position # including meetings
   c. *time* := *time_calc*(*solution*)
   d. **If** *time* < *min_time*
      i. *min_time* := *time*
      ii. *best_partial_sol* := *partial_sol*
2. Return *best_partial_sol*

Fig. 16. Solutions' rebuilding procedure.

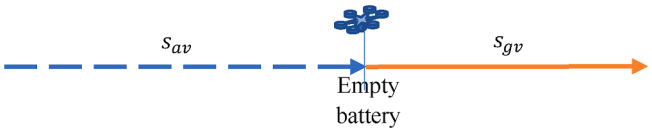

Fig. 17. Schema of over-endurance flying time penalty.

movement of the affected agent but also the movements of other ones), and the penalty value implied by these positions.

4.4. Rules to escape from local minima

At the end of each step, if no agent has improved the quality of the solution, the search is stuck at a local minimum. Then, the manager agent generates a solution (see procedure *rand_sol()* in Fig. 15), in trying to get out of this minimum. For this task, the manager uses two parameters:

- n_{out} : is the number of agents extracted from the solution and returned to it in different positions. To facilitate its adjustment, it is calculated as an integer fraction of the sequence length (*WIDTH* of the grid, $n_{out} = WIDTH//R$), where R is a parameter needed to initiate the procedure.
- max_*its* : Maximum allowed number of iterations in the procedure *rand_sol*() . This parameter allows tuning the computation time vs the solution quality obtained in this procedure.

The manager agent uses three variables in this procedure:

- *model_time* → Best time (mission completion time) found by agents and stored in the manager.
- *total_time* → Time found after applying the current procedure (in this case, *rand_sol*()).
- *iteration* → Current iteration in *rand_sol*() procedure.

Once the agents have been extracted from the solution, the solution must be rebuilt by reinserting these agents in the grid. The procedure is shown in Fig. 16, being:

- *extracted* → Set of n_{out} agents removed from the solution

To calculate the value of the objective function (in our problem, the total mission time needed to complete a mission), as other authors do (Murray & Chu, 2015), we assume the drones continue flying when waiting for the truck or another drone, until they arrive at the meeting point. This point is relevant due to the limited endurance of drones.

Intending to improve the search procedure in the solution space, our algorithm allows solutions in which the maximum endurance constraint is neglected. For these solutions, a time value penalty is added to the flight time. Each time a drone arrives at a meeting point, its flight time is compared with its endurance. If the time is greater than the endurance, we suppose that the drone continues flying but at a slower speed (see Fig. 17). At each meeting point, the maximum excess time of all drones arriving at that point is calculated and added to an excess time variable.

The penalty time is obtained by considering that the over-endurance distance is travelled by the aerial vehicle at the speed of the ground vehicle (truck) $s_{gv} < s_{av}$.

The penalty time expression, t_{penal}, is calculated as follows (1, 2):

$$t_{penal} = \frac{s_{av}}{s_{gv}} \times p_{factor} \times \sum_i exc_i \tag{1}$$

$$exc_i = \max_j \left(t_{flightij} - Q\right) \tag{2}$$

Where:

- s_{gv} – Represents the ground vehicle speed
- s_{av} – Corresponds to the drone speed
- Q – Is the drone time endurance
- $p_{factor} \geq 1$ – Is an algorithm tuning parameter
- exc_i – Represents the excess in flying time over endurance for all drones landing on location i
- $t_{flightij}$ – Is the flying time for drone j landing on location i

Thus, the maximum excess in flying time at each location is added up to obtain the total penalty time. We decided to use a correction p_{factor} since, in some instances, the gain in the mission time allowing the excess in the drones' endurance was greater than the total penalty obtained without this correction.

5. Results and discussions

We have tested the behaviour of our agent-based algorithm when applied to a set of different size instances from the literature. The experiments have been carried out on a desktop personal computer running Windows, with an Intel Core i7 2.8 GHz processor and 16 GB of RAM. The algorithm has been coded in Python 3 using the Mesa agent library (Kazil et al., 2020).

First, we compared the results obtained using the agent-based approach algorithm with the results obtained using the iterated greedy approach from (Gonzalez-R et al., 2020) in medium-size instances (those from 50 to 250 locations) and using only one drone. Then, we used our agent-based algorithm for solving larger size instances using more than one drone. For these instances we compared the results with those obtained by using only one drone.

As expected, the agent-based approach helps to find solutions for large-size instances in a short computation time, whereas it is less competitive in small size problems. In these small instances, concerning the computation time, there is no advantage when compared with the iterated greedy approach for the same order in the quality of solutions. On the contrary, we report the results also for 375 and 500 locations problems using a fleet up to 6 drones, allowing them to visit more than one location per flight and allowing the truck to serve locations, which have not been reported in the literature before, as seen in the exhaustive review in (Moshref-Javadi & Winkenbach, 2021). In medium and small size instances (<20 locations), agent-based methods are less competitive

Table 2
Tested parameter values.

Parameter	Type	Values
Penalty parameter (p_{factor})	Categorical	{1.5, 2.0, 2.5}
Parameter to build maximum swapping radius (*max_r*)	Integer	[4, 6]
Parameter to build maximum extracted locations (R)	Integer	[8, 10]
Maximum iterations to escape from local minima (*max_its*)	Integer	[1, 2]

Table 3
Best parameter configurations found by Irace package.

config	p_{factor}	*max_r*	R	*max_its*
1	**2.5**	**4**	**10**	**1**
8	2.5	5	10	1
15	2.5	4	9	2
16	2.0	6	10	2

than simpler heuristics, obviously for small instances (<10 locations) they are also less competitive than exact methods.

5.1. Global results

As in our previous work (Gonzalez-R et al., 2020), the set of instances used in this research is based on the one available in [dataset](Bouman et al., 2020). We have only considered the uniform set of instances; where locations are situated in a two-dimensional plane, distances are the Euclidean distances, and the locations are distributed uniformly over a (1, 100) square.

Parameter tuning has been made by using the Iterated Racing (Irace) algorithm by (López-Ibáñez et al., 2016). The considered values for these parameters are summarized in Table 2.

The parameter p_{factor} is used in equation (1) to impose the penalty time in which a non-feasible solution is augmented. *max_r* is used to limit horizontal movement of agents (see Fig. 9). R is used to calculate n_{out}, the number of agents extracted in the *rand_sol*() procedure (see Fig. 15) to escape from local minima. Finally, *max_its* is the maximum number of iterations in the *rand_sol*() procedure, to balance computation time and quality of solutions.

The maximum computation time was set to 600 s in the first set of experiments. We took a sample of 100 problems between 50 and 500 locations. 14 of these problems, 2 for each problem size, are reserved for the statistical searching of the best parameter configurations. The best-attained configurations are shown in Table 3.

A Wilcoxon signed pair rank test between the best parameter configurations provides significant differences between configurations 8 and 16 but not between 1 and 8, so finally we chose configuration number 1.

We made 10 runs for each of the instances available in [dataset] (Bouman et al., 2020) with an alpha parameter value equal to 2. Each problem size was run changing the number of drones from 1 to 6. (Since there are 10 instances for each problem size from 50 to 175 locations, 20 instances of size 250, 15 instances of 375 and 11 of 500 locations and each problem size was run changing the number of drones from 1 to 6, the total number of experiments was of 5160). Remember that our algorithm admits non-feasible solutions, in terms of drones' battery endurance, allowing the exploration of different regions of the search space.

The average objective function value (total mission time in seconds) and its coefficient of variation for problem sizes, varying from 50 to 500 locations and fleet sizes from 1 to 6 drones, are shown in Table 4:

The averages in the mission time show a decrease as the number of

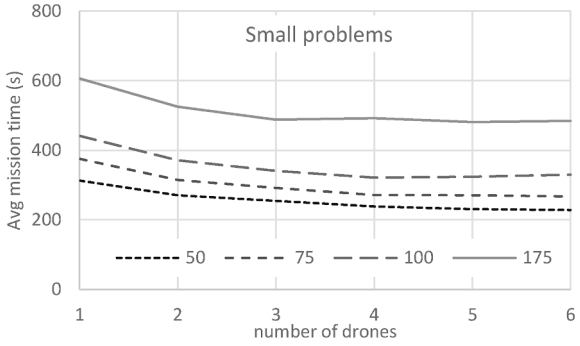

Fig. 18. Mission time averages for small problems.

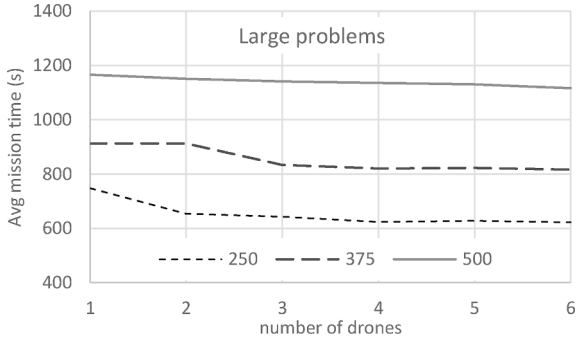

Fig. 19. Mission time averages for large problems.

Table 4
Mission time averages (m.t.avg.) and their coefficients of variation (c.v.) (%) for the agent-based TmDTL problem using 1 to 6 drones.

Problem sizes		Number of drones					
		1	**2**	**3**	**4**	**5**	**6**
50	m.t.avg.	312.3	270.2	254.7	238.0	231.0	228.2
	c.v.	(7.04%)	(10.86%)	(10.39%)	(10.76%)	(12.66%)	(11.29%)
75	m.t.avg.	375.7	314.8	291.6	271.4	270.7	266.7
	c.v.	(7.26%)	(11.55%)	(11.56%)	(12.32%)	(12.89%)	(13.69%)
100	m.t.avg.	441.5	370.9	340.8	321.6	323.4	329.7
	c.v.	(5.99%)	(9.75%)	(11.53%)	(13.15%)	(14.20%)	(13.91%)
175	m.t.avg.	606.0	525.3	488.2	491.9	481.1	484.5
	c.v.	(8.00%)	(10.16%)	(11.44%)	(15.89%)	(15.84%)	(21.31%)
250	m.t.avg.	747.8	653.9	643.3	623.5	627.9	622.6
	c.v.	(6.55%)	(7.81%)	(7.59%)	(7.91%)	(10.67%)	(8.90%)
375	m.t.avg.	912.3	912.8	834.1	820.9	822.3	816.5
	c.v.	(8.21%)	(8.15%)	(14.18%)	(18.25%)	(21.50%)	(24.57%)
500	m.t.avg.	1166.1	1150.6	1140.6	1136.0	1130.3	1115.8
	c.v.	(17.01%)	(15.83%)	(15.20%)	(16.45%)	(17.20%)	(17.04%)

Table 5
Objective function obtained by the iterated greedy (IG) and by the agent-based system (ABS) both using only one drone.

n	IG	ABS
50	311.4	312.3
75	359.3	375.7
100	406.4	441.5
175	511.5	606.0
250	605.9	747.8

Table 6
Mission time improvement with the number of drones with respect to one drone.

	Number of drones				
$\Delta t_{1,ndr}$	2	3	4	5	6
50	13.48%	18.46%	23.81%	26.04%	26.95%
75	16.22%	22.38%	27.77%	27.96%	29.01%
100	15.99%	22.81%	27.16%	26.75%	25.33%
175	13.32%	19.43%	18.83%	20.62%	20.05%
250	12.55%	13.97%	16.62%	16.03%	16.74%
375	−0.05%	8.57%	10.02%	9.86%	10.51%
500	1.33%	2.19%	2.58%	3.07%	4.31%

Table 7
Mission time incremental improvement compared with one drone less.

	Number of drones				
$\Delta t_{ndr-1,ndr}$	2	3	4	5	6
50	13.48%	5.76%	6.56%	2.92%	1.23%
75	16.22%	7.35%	6.94%	0.26%	1.46%
100	15.99%	8.12%	5.63%	−0.56%	−1.94%
175	13.32%	7.05%	−0.75%	2.20%	−0.72%
250	12.55%	1.62%	3.08%	−0.71%	0.85%
375	−0.05%	8.62%	1.58%	−0.18%	0.71%
500	1.33%	0.87%	0.40%	0.50%	1.28%

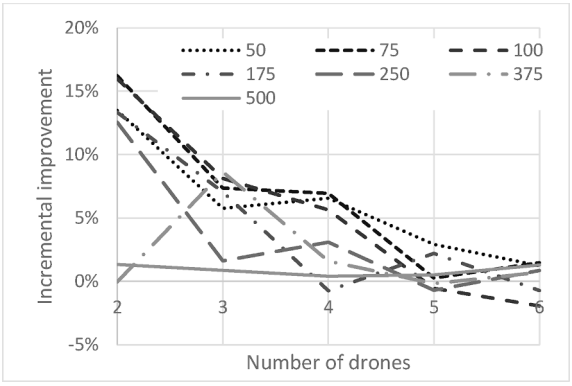

Fig. 20. Mission time improvement when adding a drone to the fleet.

drones increases and an increase as the problem size enlarges. The decrease in the mission time when the fleet size expands is less important for small-size problems, indicating the limit in the optimal fleet size. This effect is better shown in Fig. 18 and Fig. 19.

To validate these results, they are compared with those obtained in our previous research (Gonzalez-R et al., 2020) for the TDTL problem, where the truck only carried one drone and the problem was addressed with an iterated greedy meta-heuristic. As shown in Table 5, for small size problems and similar computing times, the average quality of solutions, though being slightly poorer is of the same order of magnitude. This is a good result in medium size, less complex problems (up to 250 locations and only one drone fleet) since the behavior and intelligence of agent-based systems involve greater computational complexity. This added complexity is balanced with the problem partitioning made by agents.

Table 8
One-way ANOVA test results for mission time grouped by number of drones from 1 to 6, number of locations from 50 to 500 and 600 s computing time per run.

Number of locations	P-value
50	$6.2793*10^{-101}$
75	$1.8632*10^{-104}$
100	$3.98202*10^{-95}$
175	$8.91217*10^{-40}$
250	$1.0853*10^{-108}$
375	$5.61603*10^{-22}$
500	0.54168935

Table 9
Experiment design for 2400 s computing time and problems containing 250, 375 & 500 locations.

Variable parameters	Type	Values
Penalty parameter (p_{factor})	Categorical	{1.5, 2.0, 2.5}
Parameter to build maximum swapping radius (*max_r*)	Integer	[4, 6]
Parameter to build maximum extracted locations (R)	Integer	[10, 15, 20]
Maximum iterations to escape from local minima (*max_its*)	Integer	[1, 2]

Table 6 shows the improvement obtained in the mission time t when the fleet size *ndr* increases in comparison with the time obtained when only one drone helps the truck (3):

$$\Delta t_{1,ndr} = \frac{t(1) - t(ndr)}{t(1)} \tag{3}$$

These improvements can be better interpreted when the comparisons are made considering the improvement obtained when using one drone less (4), as reported in Table 7 and Fig. 20.Table 8.

$$\Delta t_{ndr-1,ndr} = \frac{t(ndr-1) - t(ndr)}{t(ndr-1)} \tag{4}$$

Table 7 reports the incremental improvement in the total mission time when adding a new drone to the fleet with respect to the total mission time using one drone less – i.e. the average improvement in the mission time for 250 locations instances when using 4 drones with respect to using 3 drones is of 3%.

We have carried out ANOVA tests in order to analyze the differences between mission times averages when the variation in the number of drones is statistically meaningful. Here, for the sake of clarity, we only show the P-values results. A more complete set of results is included in Appendix 1.

These results show how the improvement decreases as the number of drones used increases. This improvement is smaller for larger size instances. The P-values are smaller than 0.05 in all cases except in the last one, corresponding to the 500 locations instances, in line with the expected behaviour of our algorithm. In our opinion, since the computation time is limited to 600 s, for these kinds of complex problems, the agents' ability to evolve in the search for better quality solutions is constrained. Thus, in order to assess whether this behaviour is due to limitations in computing time, we have re-run the experiments, this time allowing the algorithm to run for 2400 s for the larger size instances.

5.2. Analysis of large-size instances

As previously mentioned, a set of additional experiments extending the computation time to 2400 s have been carried out for problems with

Table 10
Mission time averages (m.t.avg.) and their coefficients of variation (c.v.) (%) for the agent-based TmDTL problem using 1 to 6 drones for 250 to 500 locations problems.

Problem sizes		Number of drones					
		1	2	3	4	5	6
250	m.t.avg.	683.05	600.16	582.32	574.29	552.70	542.32
	c.v.	(5.63%)	(7.92%)	(7.74%)	(8.09%)	(10.37%)	(9.91%)
375	m.t.avg.	862.11	865.62	852.82	756.73	746.14	730.53
	c.v.	(5.83%)	(6.52%)	(6.68%)	(9.48%)	(8.06%)	(10.46%)
500	m.t.avg.	1005.11	998.84	1016.37	1007.85	939.87	889.74
	c.v.	(9.90%)	(8.54%)	(11.97%)	(10.58%)	(16.54%)	(15.49%)

Table 11
Average mission time improvement when compared with the mission time using only one drone.

	Number of drones				
Δt_{1n}	2	3	4	5	6
250	12.14%	14.75%	15.92%	19.08%	20.60%
375	−0.41%	1.08%	12.22%	13.45%	15.26%
500	0.62%	−1.12%	−0.27%	6.49%	11.48%

Table 12
Average mission time improvements when compared with the mission time using one drone less.

	Number of drones				
$\Delta t_{(n-1)n}$	2	3	4	5	6
250	12.14%	2.97%	1.38%	3.76%	1.88%
375	−0.41%	1.48%	11.27%	1.40%	2.09%
500	0.62%	−1.75%	0.84%	6.75%	5.33%

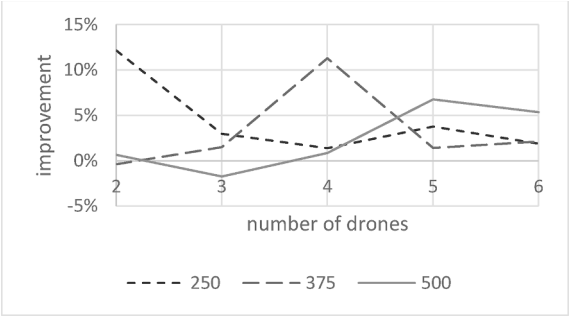

Fig. 21. Mission time improvement when adding one drone to the fleet in large problems.

250, 375, and 500 locations. For these new experiments, the maximum number of drones' steps has been removed as a stopping condition. The parameters were adjusted again as summarized in Table 9:

The best parameter values obtained by the Irace package for the new set of experiments are:

- p_{factor} : 1.5
- max_r : 6
- R : 20
- max_its : 1

For each one of the instances, 20 runs have been carried out by varying the fleet size from 1 to 6 drones. This supposes a total of 920 × 6 = 5520 experiments and approximately 3680 h of computation time.

The results show, as expected, an improvement in the solutions quality and a reduction in their variability (Table 10.

In this case, Table 11 and Table 12 show the improvement in the mission time when the fleet size increases. Table 11 represents these increments when comparing with the use of only one drone whereas Table 12 depicts the case when comparing the use of one drone less (see also Figure 21).

To validate the differences observed in average mission times when varying the number of drones, we carried out a new set of ANOVA tests (see Table 13), separated by problem size.

Now, differences are meaningful for P values, corroborating that, for our method, 600 s is a too short time.

Table 13
ANOVA test for mission time grouped by number of drones from 1 to 6 and 250 to 500 locations problems.

Locations	P-value
250	0
375	$1.7932*10^{-254}$
500	$1.62729*10^{-39}$

Table 14
Mission time averages (s) and coefficient of variation (%) for 500 locations problems, 5 to 8 drones and 2400 s comp. time.

Number of drones			
5	6	7	8
939.87 (16.54%)	889.74 (15.49%)	905.28 (19.53%)	912.18 (22.86%)

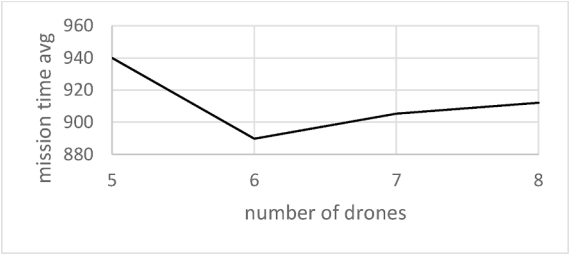

Fig. 22. Total mission time averages in 500 locations and 5–8 drones problems.

5.3. Fleet size analysis for complex problems

To confirm the results obtained for the larger size instances, we made a new test in which the algorithm was run for 2400 s, varying the fleet size between 6 and 8 drones, and testing only the 11 instances with 500 locations, running again 20 times in each instance. The objective function reduced its values as expected, but the variability continues to be relatively high (Table 14) and no improvement is observed with more than 6 drones, which implies a limit in fleet size for this configuration (see also Fig. 22).

Although we can now see a great improvement between 5 and 6 drones, again the increasing complexity of these large problems makes it difficult to achieve reasonable improvements when considering a higher

Table 15
ANOVA test for mission time grouped by number of drones from 5 to 6 and from 6 to 8, 500 locations problems.

	P-value
5 to 6 drones	0.00062667
6 to 8 drones	0.45150698

number of drones, between 6 and 8. The ANOVA tests corroborate the differences obtained between 5 and 6 drones and the similarities between mission times when using 6 to 8 drones (Table 15).

One more time, we have tested these results to assess the progress in the mission time when adding drones to the fleet:

And again, notwithstanding the misssion times are statistically different for each number of drones, the P-values confirm that, the variation is less marked than the ones in less complex problems so, we can think again in a shortage in testing time.

This variability in the quality of solutions can be better seen in the following boxplot graphics (Figs. 23 and 24). They represent the total mission time for each run of each instance and highlight that instances 6, 12 and 21, showing an important dispersion in their results no matter the size of the fleet.

6. Conclusions

We have designed a novel agent-based system capable of addressing complex routing problems, such as the Truck multi-Drone Team Logistic problem, in which a set of locations or nodes must be visited by a team composed of a truck and several UAVs for ISR (Intelligence, Surveillance, and Reconnaissance) or logistics missions. The locations must be visited once, either by truck or by one of the drones. The drones have a limited endurance due to their limited battery capacity, whereas the truck is supposed to have unlimited endurance. Unlike other works, each drone may visit an a priori unknown number of locations during each flight. Moreover, no a priori route is defined for the truck.

We have followed an approach in which locations are modelled as software agents seeking to find good quality solutions by moving themselves in an abstract grid to represent the problem' solutions. They follow a communication scheme managed by one agent and choose their allowed movements with certain probabilities, avoiding early convergence to local minima. In the case that an agent does not find improvements in the objective function value, the manager agent corrects other agents positions, shaking them and allowing them to move to a different region of the search space. This process is repeated a predetermined number of steps or during a pre-determined amount of time.

The algorithm has been tested by using a well-known benchmark set by varying the number of drones and the problem sizes. To validate the proposed approach, the results obtained with only one drone have been compared to those obtained with our previous metaheuristic. The results are promising since our agent-based algorithm has found solutions of a similar quality to the ones found in our former study for 50 locations problems, and in similar computation times. Moreover, for larger-size problems, the algorithm has found solutions in reasonable computation times.

We have also detected an increase on the variability of results, as the complexity of the problem increases, both with the number of locations and the size of the fleet. In our opinion, this variability could be reduced by simply enlarging the computation time.

As a future research line, we consider a direct enhancement of our agent-based system by deeping into development of a parallel version. If the communication between agents is fluent, the advantage of this

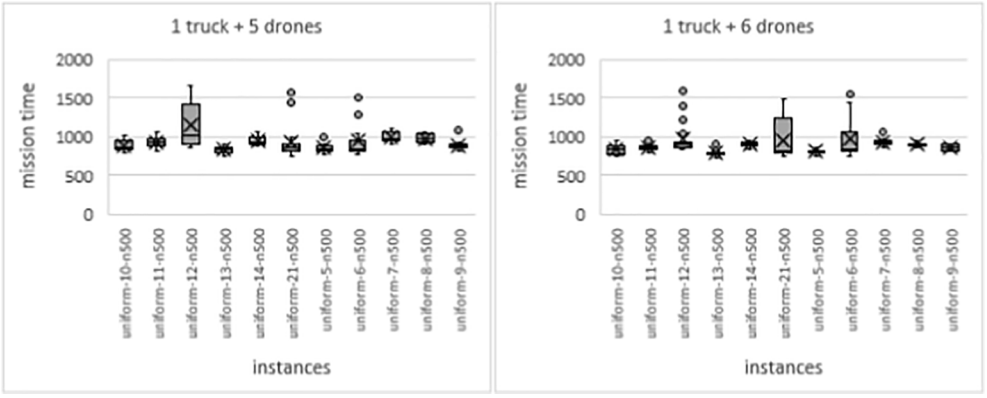

Fig. 23. Boxplot graphic for 500 locations problems experiments, 1 truck, 5 and 6 drones.

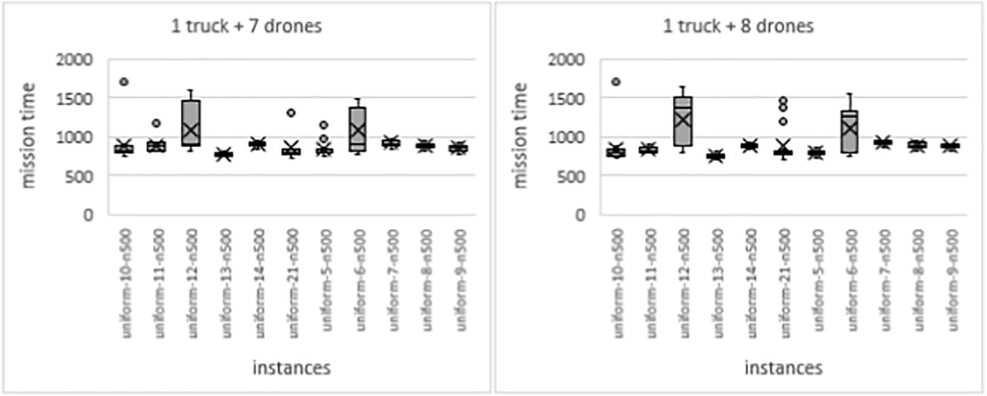

Fig. 24. Boxplot graphic for 500 locations problems experiments, 1 truck and 7 drones.

inherently parallel architecture would benefit the speedup in computation time, which could be superlinear with the number of agents. These improvements of the algorithm will allow it to address not only the studied problem, finding the optimum fleet size in the largest problem instances but also the inclusion of more than one ground vehicle. The parallel programming of the algorithm will further allow to consider constraints that would approximate the problem to other real-life situations such as differences between vehicles, payload or battery depletion laws, as they have been considered in the literature.

CRediT authorship contribution statement

Jose Miguel Leon-Blanco: Methodology, Software, Writing – original draft. **P.L. Gonzalez-R:** Conceptualization, Supervision. **Jose L. Andrade-Pineda:** Investigation, Resources. **D. Canca:** Data curation, Writing – review & editing. **M. Calle:** Formal analysis, Visualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was supported by Universidad de Sevilla and Junta de Andalucia [grant number US-1381656]. Thanks are done to the reviewers for their helpful comments.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.eswa.2022.116604.

References

Agatz, N., Bouman, P., & Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science, 52*(4), 965–981. https://doi.org/10.2139/ssrn.2639672

Alipour, M. M., Razavi, S. N., Feizi Derakhshi, M. R., & Balafar, M. A. (2018). A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem. *Neural Computing and Applications, 30*(9), 2935–2951. https://doi.org/10.1007/s00521-017-2880-4

Allan, R. J. (2009). Survey of agent based modelling and simulation tools. *Engineering, 501*(October), 57–72.

Barbati, M., Bruno, G., & Genovese, A. (2012). Applications of agent-based models for optimization problems: A literature review. *Expert Systems with Applications, 39*(5), 6020–6028. https://doi.org/10.1016/j.eswa.2011.12.015

Barbucha, D. (2012). Search modes for the cooperative multi-agent system solving the vehicle routing problem. *Neurocomputing, 88*, 13–23. https://doi.org/10.1016/j.neucom.2011.07.032

Barbucha, D., & Jedrzejovicz, P. (2007). An agent-based approach to vehicle routing problem. *International Journal of Computer and Information Engineering, 1*(2), 36–41.

Baxter, J. W., Horn, G. S., & Leivers, D. P. (2008). Fly-by-agent: Controlling a pool of UAVs via a multi-agent system. *Knowledge-Based Systems, 21*(3), 232–237. https://doi.org/10.1016/j.knosys.2007.11.005

Bellifemine, F. L., Poggi, A., & Rimassa, G. (2001). Developing multi-agent systems with JADE. In C. Castelfranchi, & Y. Lespérance (Eds.), *Intelligent Agents VII, LNAI 1986* (pp. 89–103). Berlin Heidelberg: Springer-Verlag.

Bouman, P., Agatz, N., & Schmidt, M. (2017). Dynamic programming approaches for the traveling salesman problem with drone. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3035323

Bouman, P., Agatz, N., & Schmidt, M. (2020). *TSP-D Dataset (Instances and some solutions)* (1.3). 4 Mar 2020. https://github.com/pcbouman-eur/TSP-D-Instances/releases.

Boysen, N., Briskorn, D., Fedtke, S., & Schwerdfeger, S. (2018). Drone delivery from trucks: Drone scheduling for given truck routes. *Networks, 72*(4), 506–527. https://doi.org/10.1002/net.v72.410.1002/net.21847

Caggiani, L., Marinelli, M., Orco, M. D., & Ottomanelli, M. (2017). En-route truck-drone parcel delivery for optimal vehicle routing strategies. *XXII SIDT National Scientific Seminar*.

Campbell, J. F., Sweeney, D., Zhang, J., & Pan, D. (2017). Strategic Design of Drone Delivery Systems VIII International Workshop on Locational Analysis and Related Problems UMSL and St . Louis. *VIII International Workshop on Locational Analysis and Related Problems, September*, 1–66. http://redloca.ulpgc.es/images/doc-ws/Talk_Campbell_Workshop2017.pdf.

Carlsson, J. G., & Song, S. (2018). Coordinated logistics with a truck and a drone. *Management Science, 64*(9), 4052–4069. https://doi.org/10.1287/mnsc.2017.2824

Chang, Y. S., & Lee, H. J. (2018). Optimal delivery routing with wider drone-delivery areas along a shorter truck-route. *Expert Systems with Applications, 104*, 307–317. https://doi.org/10.1016/j.eswa.2018.03.032

Cheng, C., Adulyasak, Y., & Rousseau, L. (2018). *Formulations and Exact Algorithms for Drone Routing Problem* (Issue July). https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2018-31.pdf.

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. https://doi.org/10.1016/j.cor.2020.105004

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research, 6*(6), 791–812. https://doi.org/10.1287/opre.6.6.791

Davidsson, P., Henesey, L., Ramstedt, L., Törnquist, J., & Wernstedt, F. (2005). An analysis of agent-based approaches to transport logistics. *Transportation Research Part C: Emerging Technologies, 13*(4), 255–271. https://doi.org/10.1016/j.trc.2005.07.002

Dazhi, W., & Shixin, L. (2010). An Agent-based Evolutionary Search for Dynamic Travelling Salesman Problem. *2010 WASE International Conference on Information Engineering, 1*, 111–114. 10.1109/ICIE.2010.34.

Dickinson, I. J. (1997). *Agent standards*. http://shiftleft.com/mirrors/www.hpl.hp.com/techreports/97/HPL-97-156.pdf.

Faber, Ł., Piketak, K., Byrski, A., & Kisiel-Dorohinicki, M. (2012). Agent-based simulation in AgE framework. In A. Byrski, Z. Oplatková, M. Carvalho, & M. Kisiel-Dorohinicki (Eds.), *Advances in intelligent modelling and simulation: simulation tools and applications* (pp. 55–83). Berlin Heidelberg: Springer, 10.1007/978-3-642-28888-3_3.

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. *Journal of Industrial Engineering and Management, 9*(2), 374–388. https://doi.org/10.3926/jiem.1929

FIPA – The Foundation of Intelligent Physical Agents. (n.d.). Retrieved May 19, 2020, from http://www.fipa.org/.

Galland, S., Gaud, N., Demange, J., & Koukam, A. (2009). Environment Model for Multiagent-Based Simulation of 3D Urban Systems. *7th European Workshop on Multiagent Systems (EUMAS09)*.

Gath, M., Herzog, O., & Vaske, M. (2015). Concurrent and distributed shortest-path searches in multiagent-based transport systems. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9420*, 140–157. https://doi.org/10.1007/978-3-319-27543-7_7

Gonzalez-R, P. L., Canca, D., Andrade-Pineda, J. L., Calle, M., & Leon-Blanco, J. M. (2020). Truck-drone team logistics: A heuristic approach to multi-drop route planning. *Transportation Research Part C: Emerging Technologies, 114*, 657–680. https://doi.org/10.1016/j.trc.2020.02.030

Ha, Q. M., Deville, Y., Pham, Q. D., & Hà, M. H. (2018). On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies, 86*(December 2017), 597–621. https://doi.org/10.1016/j.trc.2017.11.015.

Ham, A. M. (2018). Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming. *Transportation Research Part C: Emerging Technologies, 91*(March), 1–14. https://doi.org/10.1016/j.trc.2018.03.025

Hasan, M., & Niyogi, R. (2020). A meta-heuristic based multi-agent approach for last mile delivery problem. *ICEIS 2020 - Proceedings of the 22nd International Conference on Enterprise Information Systems, 1*(Iceis), 498–505. 10.5220/0009349004980505.

Houseknecht, J. (2019). *An ACO-inspired, probabilistic, greedy approach to the Drone Traveling Salesman Problem* [Liberty]. https://digitalcommons.liberty.edu/honors/849.

Hu, M., Liu, W., Lu, J., Fu, R., Peng, K., Ma, X., & Liu, J. (2019). On the joint design of routing and scheduling for Vehicle-Assisted Multi-UAV inspection. *Future Generation Computer Systems, 94*, 214–223. https://doi.org/10.1016/j.future.2018.11.024

Janssen, M. A., Alessa, L. N., Barton, M., Bergin, S., & Lee, A. (2008). Towards a community framework for agent-based modelling. *Journal of Artificial Societies and Social Simulation, 11*(2), 6. http://jasss.soc.surrey.ac.uk/11/2/6.html.

Jeong, H. Y., Song, B. D., & Lee, S. (2019). Truck-drone hybrid delivery routing: Payload-energy dependency and No-Fly zones. *International Journal of Production Economics, 214*, 220–233. https://doi.org/10.1016/j.ijpe.2019.01.010

Kalina, P., Vokřínek, J., & Mařík, V. (2015). Agents toward vehicle routing problem with time windows. *Journal of Intelligent Transportation Systems, 19*(1), 3–17. https://doi.org/10.1080/15472450.2014.889953

Karak, A., & Abdelghany, K. (2019). The hybrid vehicle-drone routing problem for pick-up and delivery services. *Transportation Research Part C: Emerging Technologies, 102*(September 2018), 427–449. 10.1016/j.trc.2019.03.021.

Kaul, C. (2018). *An agent based and ant colony metaheuristic approach to the last mile logistics problem* [The Pennsylvania State University]. https://etda.libraries.psu.edu/catalog/15135csk19.

Kazil, J., Masad, D., & Crooks, A. (2020). Utilizing Python for Agent-Based Modeling: The Mesa Framework. In R. Thomson, H. Bisgin, C. Dancy, A. Hyder, & M. Hussain (Eds.), *Social, Cultural, and Behavioral Modeling (Issue April)* (pp. 308–317). Springer International Publishing. https://doi.org/10.1007/978-3-030-61255-9_30.

Kazirod, M., & Knapik, M. (2016). *Pyage*. https://github.com/maciek123/pyage.

Khalid, R., & Chankov, S. M. (2020). Drone delivery using public transport: An agent-based modelling and simulation approach. In M. Freitag, H.-.-D. Haasis, H. Kotzab, & J. Pannek (Eds.), *Dynamics in logistics* (pp. 374–383). Springer International Publishing.

Kim, M., & Matson, E. T. (2017). Highlights of Practical Applications of Cyber-Physical Multi-Agent Systems. In J. Bajo, Z. Vale, P. Pawlewski, E. Del Val, P. Novais, F. Lopes, N. D. Duque Méndez, V. Julián, & J. Holmgren (Eds.), *Highlights of Practical*

Applications of Cyber-Physical Multi-Agent Systems (Vol. 722). Springer International Publishing, 10.1007/978-3-319-60285-1.

Kiran, M., Richmond, P., Holcombe, M., Chin, L. S., Worth, D., & Greenough, C. (2010). FLAME: Simulating large populations of agents on parallel hardware architectures. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems* (pp. 1633–1636).

KQML. (1993). https://www.csee.umbc.edu/csee/research/kqml/.

Kulkarni, A. J., & Tai, K. (2010). Probability Collectives: A multi-agent approach for solving combinatorial optimization problems. *Applied Soft Computing Journal, 10*(3), 759–771. https://doi.org/10.1016/j.asoc.2009.09.006

Leitão, P., Mařík, V., & Vrba, P. (2013). Past, present, and future of industrial agent applications. *IEEE Transactions on Industrial Informatics, 9*(4), 2360–2372. https://doi.org/10.1109/TII.2012.2222034

Lopes Silva, M. A., de Souza, S. R., Freitas Souza, M. J., & Bazzan, A. L. C. (2019). A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems. *Expert Systems with Applications, 131*, 148–171. https://doi.org/10.1016/j.eswa.2019.04.056

Lopes Silva, M. A., de Souza, S. R., Freitas Souza, M. J., & de França Filho, M. F. (2018). Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis. *Applied Soft Computing Journal, 71*, 433–459. https://doi.org/10.1016/j.asoc.2018.06.050

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives, 3*, 43–58. https://doi.org/10.1016/j.orp.2016.09.002

Luo, Z., Liu, Z., & Shi, J. (2017). A two-echelon cooperated routing problem for a ground vehicle and its carried unmanned aerial vehicle. *Sensors, 17*(5), 1144. https://doi.org/10.3390/s17051144

Luo, Z., Poon, M., Zhang, Z., Liu, Z., & Lim, A. (2021). The multi-visit traveling salesman problem with multi-drones. *Transportation Research Part C: Emerging Technologies, 128*(April), Article 103172. https://doi.org/10.1016/j.trc.2021.103172

Macal, C. M., & North, M. J. (2010). Tutorial on agent-based modelling and simulation. *Journal of Simulation, 4*(3), 151–162. https://doi.org/10.1057/jos.2010.3

Macrina, G., Di Puglia Pugliese, L., Guerriero, F., & Laporte, G. (2020). Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies, 120*(August), Article 102762. https://doi.org/10.1016/j.trc.2020.102762

Mathew, N., Smith, S. L., & Waslander, S. L. (2015). Planning paths for package delivery in heterogeneous multirobot teams. *IEEE Transactions on Automation Science and Engineering, 12*(4), 1298–1308. https://doi.org/10.1109/TASE.2015.2461213

Moshref-Javadi, M., Hemmati, A., & Winkenbach, M. (2020). A truck and drones model for last-mile delivery: A mathematical model and heuristic approach. *Applied Mathematical Modelling, 80*, 290–318. https://doi.org/10.1016/j.apm.2019.11.020

Moshref-Javadi, M., & Lee, S. (2017). Using drones to minimize latency in distribution systems. *67th Annual Conference and Expo of the Institute of Industrial Engineers 2017*, 235–241.

Moshref-Javadi, M., Lee, S., & Winkenbach, M. (2020). Design and evaluation of a multi-trip delivery model with truck and drones. *Transportation Research Part E: Logistics and Transportation Review*, *136*(December 2019), 101887. 10.1016/j.tre.2020.101887.

Moshref-Javadi, M., & Winkenbach, M. (2021). Applications and Research avenues for drone-based models in logistics: A classification and review. *Expert Systems with Applications, 177*(February), Article 114854. https://doi.org/10.1016/j.eswa.2021.114854

Mualla, Y., Bai, W., Galland, S., & Nicolle, C. (2018). Comparison of agent-based simulation frameworks for unmanned aerial transportation applications. *Procedia Computer Science, 130*, 791–796. https://doi.org/10.1016/j.procs.2018.04.137

Murray, C. C., & 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. https://doi.org/10.1016/j.trc.2015.03.005

Murray, C. C., & Raj, R. (2020). The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies, 110*, 368–398. https://doi.org/10.1016/j.trc.2019.11.003

Nathan Koenig, A. H. (2004). Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator. *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, *4*, 2149–2154.

North, M. J., Howe, T. R., Collier, N. T., & Vos, J. R. (2005). The repast simphony runtime system. *Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms*, *10*, 13–15.

Othman, M. S. B., Shurbevski, A., Karuno, Y., & Nagamochi, H. (2017). Routing of carrier-vehicle systems with dedicated last-stretch delivery vehicle and fixed carrier route. *Journal of Information Processing, 25*(0), 655–666. https://doi.org/10.2197/ipsjjip.25.655

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.v72.410.1002/net.21818

Palanca, J., & Alemany, S. (2017). *SPADE*. https://spade-mas.readthedocs.io/.

Pěchouček, M., & Mařík, V. (2008). Industrial deployment of multi-agent technologies: Review and selected case studies. *Autonomous Agents and Multi-Agent Systems, 17*(3), 397–431. https://doi.org/10.1007/s10458-008-9050-0

Phan, A. T., Nguyen, T. D., & Pham, Q. D. (2018). Traveling salesman problem with multiple drones. *ACM International Conference Proceeding Series, 46–53*. https://doi.org/10.1145/3287921.3287932

Poikonen, S., & Golden, B. (2020a). Multi-visit drone routing problem. *Computers and Operations Research, 113*, 104802. https://doi.org/10.1016/j.cor.2019.104802

Poikonen, S., & Golden, B. (2020b). The mothership and drone routing problem. *INFORMS Journal on Computing, 32*(2), 249–262. https://doi.org/10.1287/ijoc.2018.0879

Poikonen, S., Wang, X., & Golden, B. (2017). The vehicle routing problem with drones: Extended models and connections. *Networks, 70*(1), 34–43. https://doi.org/10.1002/net.v70.110.1002/net.21746

Roberti, R., & Ruthmair, M. (2021). Exact methods for the traveling salesman problem with drone. *Transportation Science, 55*(2), 315–335. https://doi.org/10.1287/TRSC.2020.1017

Rojas Viloria, D., Solano-Charris, E. L., Muñoz-Villamizar, A., & Montoya-Torres, J. R. (2020). Unmanned aerial vehicles/drones in vehicle routing problems: A literature review. *International Transactions in Operational Research*, 1–32. https://doi.org/10.1111/itor.12783

Sacramento, D., Pisinger, D., & Ropke, S. (2019). An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies, 102*, 289–315. https://doi.org/10.1016/j.trc.2019.02.018

Salama, M., & Srinivas, S. (2020). Joint optimization of customer location clustering and drone-based routing for last-mile deliveries. *Transportation Research Part C: Emerging Technologies, 114*(March), 620–642. https://doi.org/10.1016/j.trc.2020.01.019

Schelling, T. C. (1958). The strategy of conflict: Prospectus for a reorientation of game theory. *Journal of Conflict Resolution, 2*(3), 203–264. https://doi.org/10.1177/002200275800200301

Schermer, D., Moeini, M., & Wendt, O. (2019). A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations. *Computers and Operations Research, 109*, 134–158. https://doi.org/10.1016/j.cor.2019.04.021

Schermer, D., Moeini, M., Wendt, O., B, M. M., & Wendt, O. (2018). Algorithms for Solving the Vehicle Routing Problem with Drones. In N. T. Nguyen, D. H. Hoang, T.-P. Hong, H. Pham, & B. Trawiński (Eds.), *Intelligent Information and Database Systems, 10th Asian Conference (ACIIDS 2018)* (pp. 352–361). Springer International Publishing AG. 10.1007/978-3-319-75417-8_33.

Semsch, E., Jakob, M., Pavlicek, D., & Pechoucek, M. (2009). Autonomous UAV Surveillance in Complex Urban Environments. *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, *2*, 82–85. 10.1109/WI-IAT.2009.132.

Shah, S., Dey, D., Lovett, C., & Kapoor, A. (2018). AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In M. Hutter, & R. Siegwart (Eds.), *Field and service robotics (Springer P* (pp. 621–635). Cham: Springer, 10.1007/978-3-319-67361-5_40.

Talukdar, S., Pyo, S., & Giras, T. (1983). Asynchronous procedures for parallel processing. *IEEE Transactions on Power Apparatus and Systems, PAS-102*(11), 3652–3659.

Talukdar, S. N., & Ramesh, V. C. (1992). A-teams for real-time operations. *International Journal of Electrical Power and Energy Systems*, *14*(2–3), 138–143. http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-0042265325&partnerID=40&rel=R8.2.0.

Thangiah, S. R., Shmygelska, O., & Mennell, W. (2001). An agent architecture for vehicle routing problems. In *Proceedings of the ACM Symposium on Applied Computing* (pp. 517–521), 10.1145/372202.372445.

Vásquez, S. A., Angulo, G., & Klapp, M. A. (2021). An exact solution method for the TSP with Drone based on decomposition. *Computers and Operations Research, 127*, 105127. https://doi.org/10.1016/j.cor.2020.105127

Wang, D., Hu, P., Du, J., Zhou, P., Deng, T., & Hu, M. (2019). Routing and scheduling for hybrid truck-drone collaborative parcel delivery with independent and truck-carried drones. *IEEE Internet of Things Journal, 6*(6), 10483–10495. https://doi.org/10.1109/JIOT.2019.2939397

Wang, X., Poikonen, S., & Golden, B. (2017). The vehicle routing problem with drones: Several worst-case results. *Optimization Letters, 11*(4), 679–697. https://doi.org/10.1007/s11590-016-1035-3

Wang, Z., & Sheu, J. B. (2019). Vehicle routing problem with drones. *Transportation Research Part B: Methodological, 122*, 350–364. https://doi.org/10.1016/j.trb.2019.03.005

Wilensky, U. (1999). *Center for connected learning and computer-based modeling*. NetLogo: Northwestern University. http://ccl.northwestern.edu/netlogo/.

Yoon, J. J. (2018). The Traveling Salesman Problem with Multiple Drones: An Optimization Model for Last-Mile Delivery [Massachusetts Institute of Technology]. In *Massachusetts Institute of Technology. Supply Chain Management Program*. https://dspace.mit.edu/handle/1721.1/117930#files-area.

Zeddini, B., Temani, M., Yassine, A., & Ghedira, K. (2008). An agent-oriented approach for the dynamic vehicle routing problem. *International Workshop on Advanced Information Systems for Enterprises, 2008*, 70–76. https://doi.org/10.1109/IWAISE.2008.16