Contents lists available at ScienceDirect

Computers and Operations Research

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

Parallel drone scheduling vehicle routing problems with collective drones

Roberto Montemanni [*], Mauro Dell'Amico, Andrea Corsini

Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Via Amendola 2, 42122 Reggio Emilia, Italy

ARTICLE INFO

Keywords:
Parallel Drone Scheduling Vehicle Routing Problems with cooperative drones
Constraint Programming
Mixed Integer Linear Programming
Parallel Drone Scheduling Traveling Salesman Problems with cooperative drones

ABSTRACT

We study last-mile delivery problems where trucks and drones collaborate to deliver goods to final customers. In particular, we focus on problem settings where either a single truck or a fleet with several homogeneous trucks work in parallel to drones, and drones have the capability of collaborating for delivering missions. This cooperative behavior of the drones, which are able to connect to each other and work together for some delivery tasks, enhance their potential, since connected drone has increased lifting capabilities and can fly at higher speed, overcoming the main limitations of the setting where the drones can only work independently.

In this work, we contribute a Constraint Programming model and a valid inequality for the version of the problem with one truck, namely the *Parallel Drone Scheduling Traveling Salesman Problem with Collective Drones* and we introduce for the first time the variant with multiple trucks, called the *Parallel Drone Scheduling Vehicle Routing Problem with Collective Drones*. For the latter version of the problem, we propose two Constraint Programming models and a Mixed Integer Linear Programming model.

An extensive experimental campaign leads to state-of-the-art results for the problem with one truck and some understanding of the presented models' behavior on the version with multiple trucks. Some insights about future research are finally discussed.

1. Introduction

The employment of drones in last-mile delivery is considered extremely strategic for the near future by leading distribution operators. They face a continuously increasing volume of parcels to handle, mainly generated by e-commerce (Statista, 2022). Considering that drones are light-weighted and use low-emission electric motors, that they do not have to move along the road network but can fly approximately in straight lines, and that they are not affected by road traffic congestions, their adoption for deliveries could lead to advantages for the companies (operational costs reduction), for the customers (faster deliveries) and for the whole society (sustainability). Forbes (2022) refers to the heavy interest in drone technology as the "Drone Explosion". The authors of Wolleswinkel et al. (2018) forecast that autonomous vehicles will deliver about 80% of all parcels in the upcoming decade. In this work, we analyze a transition scenario where drones are used in conjunction with trucks for last-mile delivery.

Murray and Chu (2015) introduced the idea of a new routing problem in which a truck and a drone collaborate to make deliveries. The authors present two new prototypical models expanding from the traditional Traveling Salesman Problem (TSP) called the Flying Sidekick TSP (FSTSP) and the Parallel Drone Scheduling TSP (PDSTSP). In both cases, a truck and some drones collaborate to deliver parcels. In the former model, drones can be launched from the truck during its tour, while in the latter one, drones are only operated from the central depot, and the truck executes a traditional delivery tour. In the remainder of the paper, we will focus on the latter problem, addressing the interested reader, e.g., to Dell'Amico et al. (2021) and Dell'Amico et al. (2022) for details and solution strategies for the FSTSP.

More formally, in the PDSTSP there is a truck that can leave the depot, serve a set of customers, and go back to the depot. In parallel, there is also a set of drones, and each one of them can leave the depot, serve a customer, and return to the depot before serving other customers. Some of the customers cannot be served by the drones, either due to their location or the characteristic of their parcel. The objective of the optimization is to minimize the completion time of the last vehicle returning to the depot (or a cost function related to this) while serving all the customers.

A first Mixed Integer Linear Programming (MILP) model for the PDSTSP was proposed in Murray and Chu (2015) together with some simple heuristic methods. Another MILP model and the first metaheuristic method, based on a two steps strategy embedding a dynamic programming component, were discussed in Mbiadou Saleu et al. (2018). Another two-steps approach was presented in Dell'Amico et al. (2020) while a hybrid ant colony optimization metaheuristic was discussed in Dinh et al. (2021) and a variable neighbor search

[*] Corresponding author.
E-mail address: roberto.montemanni@unimore.it (R. Montemanni).

https://doi.org/10.1016/j.cor.2023.106514
Received 7 July 2023; Received in revised form 9 December 2023; Accepted 11 December 2023
Available online 26 December 2023
0305-0548/© 2023 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

one in Lei and Chen (2022). In Montemanni and Dell'Amico (2023b), an effective constraint programming approach was proposed, which optimally solved all the benchmark instances previously adopted in the literature for both exact and heuristic methods. Recently, in Nguyen et al. (2023) another exact approach based on branch-and-cut was proposed, together with new benchmark instances.

Several PDSTSP variants were also introduced and studied in the literature, see e.g., Otto et al. (2018) and Pasha et al. (2022) for extensive surveys. We review herein only those extensions of the original problem that we find more relevant to the present study.

The recent work (Mbiadou Saleu et al., 2022) discussed the *Parallel Drone Scheduling Multiple Traveling Salesman Problem*, which is a straightforward extension of the PDSTSP where multiple trucks are employed and the target is to minimize the time required to complete all the customer delivery. The authors proposed a hybrid metaheuristic algorithm, a mixed integer linear model, and a branch-and-cut approach. The same problem was independently introduced also in Raj et al. (2021), where the authors proposed three mixed integer linear programming models, together with a branch-and-price approach. A heuristic version of the branch-and-cut method was also introduced, aiming at solving the larger instances. A more realistic variation of the PDSTSP was introduced in Nguyen et al. (2022). In this version of the problem concepts such as capacity, load balancing, and decoupling of costs and times are taken into account. The authors proposed a mixed integer linear programming model and a *Ruin&Recreate* metaheuristic for the problem. Constraint Programming methods for these variants of the PDSTSP employing several trucks, were discussed in Montemanni and Dell'Amico (2023a), with convincing experimental results also presented.

One common assumption in the literature on combined truck-drone delivery models has been a linear battery consumption for drones, leading to fixed operation ranges and carrying capacities. Recently, power consumption models with more realistic settings have been presented, e.g., in Raj et al. (2021), Raj and Murray (2020), and Liu et al. (2017), where the impact of a drone's power consumption is analyzed as a function of both speed and payload. A review of drone energy consumption models is also available in Zhang et al. (2021). In the patent (Paczan et al., 2022) filed by Amazon Technologies Inc., a novel method using a so-called "Collective Drone" (c-drone) is introduced. Under the new settings, multiple drones can be coupled together to aerially transport items of large size and weight. By sharing resources, such as power and operating instructions, a collective drone might outperform a single drone to operate more efficiently. In the patent, not many details are provided about the process of coupling and decoupling of drones, that can be either realized by a human operator, or autonomously in a more advanced scenario. The emphasis is on the equipment of the drones, that are characterized by a physical coupling components and a purposed control software. The authors of Nguyen and Hà (2023) joined the new concept of c-drone with advanced power consumption models to create an innovative problem, called the PDSTSP-c, where c stands for *collective*. In this problem, a realistic model is used to calculate the endurance and capacity of groups of drones working together to carry out tasks. The authors were able to pre-compute the optimal speed to carry out a certain delivery with different (smaller or larger) formations of drones. Based on these calculations, they proposed a mixed integer programming model and a *Ruin&Recreate* metaheuristic for the newly introduced problem. An example of a PDSTSP-c instance is provided in Fig. 1.

The contributions of the present paper are as follows:

- A new Constraint Programming model for the PDSTSP-c is introduced, together with a valid inequality. Experimental results show the great potential of the new model;
- The PDSVRP-c problem is firstly introduced, where the settings of the PDSTSP-c are kept, but a fleet of vehicles is available instead of a single truck;

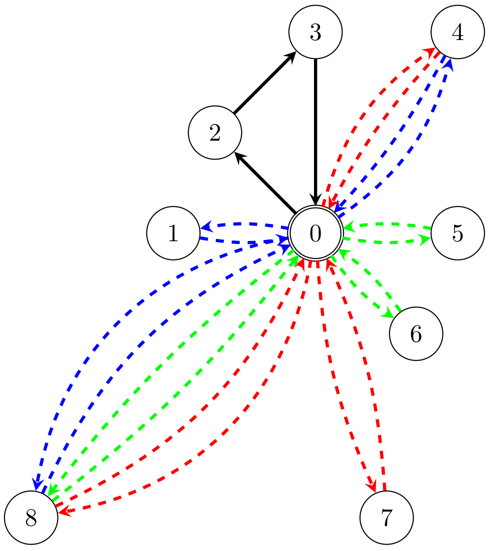

Fig. 1. Example of a PDSTSP-c instance. Node 0 is the depot, the other nodes are customers. Travel times are omitted for the sake of simplicity. The black continuous arcs represent the tour of the truck $(0, 2, 3, 0)$. The dashed arcs depict the missions of the drones, each color representing a different one. Notice that some of the missions are carried out by multiple drones.

- Two new Constraint Programming models and a Mixed Integer Programming model for the PDSVRP-c are introduced and validated through some experimental tests.

The remainder of the paper is organized as follows. In Section 2, the PDSTSP-c is formally described and a new Constraint Programming model is introduced, together with a new valid inequality. Section 3 firstly introduces the PDSVRP-c as an extension of the former problem. Two Constraint Programming models and a Mixed Integer Linear Programming model are presented. Section 4 presents experimental results for the two problems, while conclusions are drawn in Section 5.

2. The parallel drone scheduling traveling salesman problem with collective drones

In this section, we formally describe the PDSTSP-c, as originally introduced in Nguyen and Hà (2023), and we present a new Constraint Programming model. We start from the single-vehicle problem because its description helps in introducing the multiple-vehicle version.

2.1. Problem description

Given a complete graph $G(V, E)$ with the set of vertices $V = \{0, 1, \dots, n\}$, with vertex 0 representing the depot, and the remaining vertices being associated with the customers (set $C = V \setminus \{0\}$). Each customer i requests delivery of a parcel of weight w_i from the depot. The fleet of vehicles available for deliveries is composed of a driver-operated delivery truck, with unlimited range and capacity, and a set D of m homogeneous drones that are based at the depot and equipped with batteries of given capacity (a fresh battery is installed before each mission). The truck performs its task within a single tour, beginning from the depot, traversing through all assigned customers, and returning to the depot. The truck travel times between pairs of vertices $i, j \in V$ is given as t_{ij}. Matrix $[t_{ij}]$ satisfies the triangular inequality $t_{ik} \leq t_{ij}+t_{jk}, i, j, k \in V$. The drones have to perform back-and-forth trips between the depot and the customers' locations to deliver the parcels. Travel times and ranges of drone missions depend on factors such as the number of drones cooperating and the traveling speed. Given a customer i and a number k of drones executing the mission, it is possible to pre-calculate the optimal speed and consequently the total travel time τ_i^k for the back-and-forth trip. When it is not possible to service a customer i for some values of k, then τ_i^k is set to $+\infty$. We group the customers that can be serviced by truck only in the set $C_{\mathcal{T}} \subsetneq C$. Instead, let $C_{\mathcal{F}} = C \setminus C_{\mathcal{T}}$ denote the (sub)set of customers that may be served with some drones' configuration, and let q_j and p_j be the minimum and maximum number of drones to serve a customer $j \in C_{\mathcal{F}}$ (observe that in the cited model the number of drones that can serve a customer always define an interval). We adopted the realistic model described in Nguyen and Hà (2023) for the calculation of their travel times, and we refer the interested reader to this paper for full details.

Notice that a main difficulty of the problem is that once k drones collaborate for a delivery mission, strict synchronization constraints must be fulfilled. The objective of the PDSTSP-c is to find a truck tour, drone-customer assignments, and drones scheduling that minimize the makespan (i.e., the maximum completion time at which all vehicles are back at the depot after completing their services) while fulfilling all the constraints and conditions listed above.

2.2. A constraint programming model

The Constraint Programming model we present is based on the Google OR-Tools CP-SAT solver (Perron and Furnon, 2023) and follows the ideas behind the Mixed Integer Linear Program described in Nguyen and Hà (2023). In particular, drone missions are modeled through a flow. Changes have however been introduced to take full advantage of the characteristic of the solver used.

The CP-SAT solver is designed to work in a multi-thread environment (compatible with all new processors) and can be seen as a portfolio-strategy with some limited data exchange among the different threads. The main process runs a Constraint Programming Solver based on a Lazy Clause Generation (LCG) (Stuckey, 2010), but other unrelated methods work in parallel to support it and exchange information such as new bounds and solutions. The concept behind LCG involves the (incremental) transformation of the problem into a SAT-formula, subsequently employing a SAT-solver to seek a solution (or prove bounds by infeasibility). The model also gets linearized to some degree, and the corresponding linear program gets (partially) solved with the (dual) simplex algorithm and other classic MILP techniques are run to enhance bounds and retrieve new solutions, aiming at supporting the satisfiability model. Finally, different instances of a Large Neighborhood Search (LNS) metaheuristic, seeking for high-quality feasible solutions, are executed.

While the idea above may initially appear as an inefficient approach due to potential redundancy, it proves highly effective in practice. The rationale behind this lies in the inherent challenge of predicting which algorithm is best suited to solve a given problem (No Free Lunch Theorem, Wolpert and Macready, 1997). Thus, the pragmatic strategy involves running various approaches in parallel, with the hope that one will effectively address the problem at hand. In contrast, Branch and Cut-based Mixed Integer Programming solvers like Gurobi Optimization (2023) implement a more efficient partitioning of the search space to reduce redundancy. However, they specialize in a particular strategy, which may not always be the optimal choice.

The variables used in the CP model we present for the PDSTSP-c as follows:

- x_{ij}: binary variables equal to 1 (*true*) if edge (i, j), with $i, j \in V$, is traveled by the truck, 0 (*false*) otherwise. Whereas a loop $x_{jj} = 1$ means that customer j is served by drones, while $x_{jj} = 0$ if it is served by the truck.
- z_j^k: binary variables equal to 1 if customer $j \in C_{\mathcal{F}}$ is served by k drones, 0 otherwise.
- y_{ij}: binary variables equal to 1 if vertex i is served right before vertex j within the schedule of any drone, 0 otherwise.
- $f_{ij} \in Z^+$: continuous flow variables indicating number of drones serving vertex i right before vertex j in their schedule.
- $T_j \in R^+$: continuous variables representing the time at which the mission to customer $j \in C_{\mathcal{F}}$ is completed by the drones. T_0 is the start time of the operations (typically 0), with all the vehicles at the depot.
- $\alpha \in R^+$: continuous variable denoting the completion time, by which all the carriers are back to the depot.

$$(CP1): \quad \min \alpha \tag{1}$$

$$s.t. \quad \alpha \geq \sum_{i \in V} \sum_{j \in V, i \neq j} t_{ij} x_{ij} \tag{2}$$

$$\alpha \geq T_j \qquad j \in C_{\mathcal{F}} \tag{3}$$

$$x_{jj} = \sum_{q_j \leq k \leq p_j} z_j^k \qquad j \in C_{\mathcal{F}} \tag{4}$$

$$\text{Circuit}(x_{ij}, \text{ with } i, j \in V, j \neq i \text{ if } j \in C_{\mathcal{T}}) \tag{5}$$

$$\sum_{j \in C_{\mathcal{F}}} f_{0j} \leq m \tag{6}$$

$$\sum_{i \in C_{\mathcal{F}} \cup \{0\}, i \neq j} f_{ij} = \sum_{q_j \leq k \leq p_j} k z_j^k \qquad j \in C_{\mathcal{F}} \tag{7}$$

$$\sum_{i \in C_{\mathcal{F}} \cup \{0\}, i \neq j} f_{ij} = \sum_{l \in C_{\mathcal{F}} \cup \{0\}, l \neq j} f_{jl} \qquad j \in C_{\mathcal{F}} \cup \{0\} \tag{8}$$

$$f_{ij} \leq m y_{ij} \qquad i, j \in C_{\mathcal{F}} \cup \{0\}, i \neq j \tag{9}$$

$$y_{ij} \implies T_j \geq T_i + \sum_{q_j \leq k \leq p_j} \tau_j^k z_j^k \qquad i \in C_{\mathcal{F}} \cup \{0\}, j \in C_{\mathcal{F}}, i \neq j \tag{10}$$

$$0 \leq f_{ij} \leq m \qquad i, j \in C_{\mathcal{F}} \cup \{0\}, i \neq j \tag{11}$$

$$x_{ij} \in \{0; 1\} \qquad i, j \in V \tag{12}$$

$$z_j^k \in \{0; 1\} \qquad j \in C_{\mathcal{F}}, q_j \leq k \leq p_j \tag{13}$$

$$y_{ij} \in \{0; 1\} \qquad i, j \in C_{\mathcal{F}} \cup \{0\}, i \neq j \tag{14}$$

$$T_j \geq 0 \qquad j \in C_{\mathcal{F}} \cup \{0\} \tag{15}$$

Following the trivial objective function (1), the constraints have the following meaning. Constraint (2) says that the total time α has to be greater than or equal to the time required by the truck tour. Analogously, constraints (3) impose that α has to be greater than or equal to the completion time of the eventual drone mission to serve customer j. Given the logic of the variables, constraints (4) state that each drone-eligible customer has to be visited either by the truck or by a group of drones; Constraint (5) uses the CP-SAT method *Circuit* (Perron and Furnon, 2023) to have a feasible truck tour that skips each customer j for which $x_{jj} = 1$ (these customers will be visited by drones). Notice that the input parameter for the *Circuit* command is a set of arcs that the tour can visit, including self-loops. In our case we exclude only those x_{jj} for which $j \in C_{\mathcal{T}}$ (corresponding to customers that are not drone-eligible). The Constraints (6)–(8) model the operations and synchronization of the drones as a flow problem (see Nguyen and Hà, 2023 for more detailed explanations): Constraint (6) states the flow going out from node 0 has to be less than or equal to m (remind that each drone is represented as a unit of flow); Constraints (7) impose that if a customer j is serviced by k drones, than the flow entering node j has to equal k; Constraints (8) are classic conservation equalities, imposing that the flows entering and exiting a node must be equal. Constraints (9) activate the variables y corresponding to arcs used by flows (variables f) which are necessary to calculate the completion time of drones. Constraints (10) are active only if the variable $y_{ij} = 1$ and state that the synchronization constraint on arc (i, j) must be respected. This is achieved through the CP-SAT command *OnlyEnforceIf* (Perron and Furnon, 2023), which is indicated with $\implies$ in the model. The remaining constraints (11)–(15) define the domain of the variables.

2.3. Valid inequality

The following valid inequality can be introduced to improve the linear relaxation of model *CP1*.

Theorem 1. *The following inequality is valid for the model CP1:*

$$m\alpha \geq \sum_{j\in C_{\mathcal{F}}} \sum_{q_j\leq k\leq p_j} k\tau_j^k z_j^k \tag{16}$$

Proof. The value of α has to be greater than the maximum completion time of drone missions among all drone-eligible customers (from constraints (3)), which in turn is (by definition) greater than or equal to the average time spent into missions by the drones. This last value is obtained by dividing by the number of drones (m) the cumulative time spent into drone-missions, expressed for each accomplished mission as the time of the mission itself (τ_j^k) multiplied by the number of drones involved (k). Formally:

$$\alpha \geq \max_{j\in C_{\mathcal{F}}} T_j \geq \frac{\sum_{j\in C_{\mathcal{F}}} \sum_{q_j\leq k\leq p_j} k\tau_j^k z_j^k}{m} \Rightarrow (16) \quad \square$$

The valid inequality (16) will intuitively be tight when the following two conditions hold: (i) the time waited by the drones to synchronize prior to multi-drone missions is small, since this time is not captured by the inequality; (ii) the vehicles (and the drones in particular) have similar total mission times, therefore making the maximum completion time comparable to the average mission time. Notice that both these conditions tend to be fulfilled by optimized solutions.

Finally, we anticipate that the inequality (16) will be valid also for all the models discussed in Section 3 for the PDSVRP-c.

3. The parallel drone vehicle routing problem with collective drones

In this section, we build upon Section 2 and introduce the PDSVRP-c, a natural extension of the PDSTSP-c where multiple vehicles operate in parallel to the drones. The problem is introduced in Section 3.1 while two models based on Constraint Programming are discussed in Sections 3.2 and 3.3. The first one is a 2-indices formulation based on the $CP1$ model of the previous section while the second one is a 3-indices formulation. Section 3.4 outlines a MILP model to serve as a baseline.

3.1. Problem description

A formal definition of the PDSVRP-c can be proposed as a straightforward extension of the PDSTSP-c provided in Section 2.1. The difference is that now we have a fleet S of s trucks, with the same characteristics of the single truck employed for the PDSTSP-c: unlimited capacity, unlimited range, and same traveling speed. No concept of collaboration exists for the trucks and each customer has to be served either by one of the trucks or by drones.

Having a fleet of trucks does not change substantially the problem, but has an impact on the optimization since we now have to plan multiple tours and account for the mission time of each truck while calculating the completion time α. We will see in the next sections two alternative Constraint Programming models and a Mixed Integer Linear Programming formulation.

3.2. A 2-indices constraint programming model

This model is the direct extension of that discussed in Section 2.2 for the $CP1$ and delegates the Constraint Programming solver to handle the multiple truck tours. The variables remain the same, although now the x can take the shape of multiple tours instead of a single one. Another important difference is the definition of the variables T_j. In the $CP1$ model of Section 2.2, they are only related to the drones and represent the time in which the mission to a customer is completed. Here, they are extended to the customers served by the trucks and represent the *starting* time of the service of the truck to the customer. Formally we use the new variables $\overline{T}$ with the following meaning

- $\overline{T}_j \in R^+$: continuous variables with a different meaning depending of the type of vehicle involved. It represents the time at which the mission to customer $j \in C_{\mathcal{F}}$ is completed when the visit is carried out by drones (it is the time they are back to the depot). If the customer is serviced by a truck, it is the time the truck reaches the customer and the service is started. $\overline{T}_0$ denotes the start time of the operations (typically 0), with all the vehicles at the depot.

$$(CP2) \quad \min \alpha \tag{17}$$

$$s.t. \quad \alpha \geq \overline{T}_j + t_{j0}x_{j0} \qquad j \in C \tag{18}$$

$$x_{jj} = \sum_{q_j\leq k\leq p_j} z_j^k \qquad j \in C_{\mathcal{F}} \tag{19}$$

$$\text{MultipleCircuit}(x_{ij}, \text{ with } i,j \in V, i \neq 0 \vee j \neq 0, j \neq i \text{ if } i \in C_{\mathcal{T}}) \tag{20}$$

$$\sum_{j\in C} x_{0j} \leq s \tag{21}$$

$$x_{ij} \implies \overline{T}_j \geq \overline{T}_i + t_{ij} \qquad i \in V, j \in C, i \neq j \tag{22}$$

$$\sum_{j\in C_{\mathcal{F}}} f_{0j} \leq m \tag{23}$$

$$\sum_{i\in C_{\mathcal{F}}\cup\{0\}, i\neq j} f_{ij} = \sum_{q_j\leq k\leq p_j} kz_j^k \qquad j \in C_{\mathcal{F}} \tag{24}$$

$$\sum_{i\in C_{\mathcal{F}}\cup\{0\}, i\neq j} f_{ij} = \sum_{l\in C_{\mathcal{F}}\cup\{0\}, l\neq j} f_{jl} \qquad j \in C_{\mathcal{F}} \cup \{0\} \tag{25}$$

$$f_{ij} \leq my_{ij} \qquad i,j \in C_{\mathcal{F}} \cup \{0\}, i \neq j \tag{26}$$

$$y_{ij} \implies \overline{T}_j \geq \overline{T}_i + \sum_{q_j\leq k\leq p_j} \tau_j^k z_j^k \qquad i \in C_{\mathcal{F}} \cup \{0\}, j \in C_{\mathcal{F}}, i \neq j \tag{27}$$

$$0 \leq f_{ij} \leq m \qquad i,j \in C_{\mathcal{F}} \cup \{0\}, i \neq j \tag{28}$$

$$x_{ij} \in \{0;1\} \qquad i,j \in V \tag{29}$$

$$z_j^k \in \{0;1\} \qquad j \in C_{\mathcal{F}}, q_j \leq k \leq p_j \tag{30}$$

$$y_{ij} \in \{0;1\} \qquad i,j \in C_{\mathcal{F}} \cup \{0\}, i \neq j \tag{31}$$

$$\overline{T}_j \geq 0 \qquad j \in V \tag{32}$$

The constraints strictly follow the meaning already described for the $CP1$ model in Section 2.2. The only changes are as follows. Constraints (18) are now extended to cover also the case of truck visits. In this case α is defined based on the time required by each truck to go back to the depot after visiting each of its assigned customers. This constraint is valid since the travel times satisfy the triangular property, although it could be made valid also for the general case with the use of a *OnlyEnforceIf* statement (see below). Constraint (20) describes a set of circuits through the *MultipleCircuit* command of CP-SAT (Perron and Furnon, 2023) to reflect we are now dealing with several tours instead of one. The command takes in input the set of arcs feasible to be traversed, with the possibility of self-loops in case a customer is visited by the drones. Notice that the arc $(0,0)$ is excluded in order to avoid subtours not involving the depot, as well as self-loops involving customers that cannot be visited by the drones. The number of tours is not a parameter of the command, and therefore the new constraint (21) is necessary to force the number of tours to be s at most. Whereas the new constraints (22) calculate the service start time for each customer visited by a truck (remember that $\implies$ indicates the *OnlyEnforceIf*, which activates the constraint if, and only if, $x_{ij} = 1$).

3.3. A 3-indices constraint programming model

This model is another extension of the $CP1$ model in Section 2.2 that uses s separate sets of variables to describe the tours of the s trucks. All the variables remain the same, apart from the x variables which are substituted by a set of variables w such that $w_{ij}^k = 1$ if edge (i,j) is traveled by truck $k \in S$, 0 otherwise. Notice that $w_{jj}^k = 1$ means that customer j is not served by truck k, hence it is not part of its tour. In addition, $w_{00}^k = 1$ means that truck k is not operated in the solution. Notice that differently from model $CP2$, here all the loop variables for the truck are inserted when invoking the method *Circuit* used to find a circuit for each truck (see (38) below) since now only one of the trucks will have to visit a node contained in $C_{\mathcal{T}}$ (as imposed by the constraints (37) below). Finally notice that the timing variables T are the same used in model $CP1$ of Section 2.2.

$$(CP3) \quad \min \alpha \tag{33}$$

$$\text{s.t.} \quad \alpha \geq \sum_{i \in V} \sum_{j \in V, i \neq j} t_{ij} w_{ij}^k \qquad k \in S \tag{34}$$

$$\alpha \geq T_j \qquad j \in C_{\mathcal{F}} \tag{35}$$

$$\sum_{k=1}^{s} (1 - w_{jj}^k) + \sum_{q_j \leq k \leq p_j} z_j^k = 1 \qquad j \in C_{\mathcal{F}} \tag{36}$$

$$\sum_{k=1}^{s} w_{jj}^k = s - 1 \qquad j \in C_{\mathcal{T}} \tag{37}$$

$$\text{Circuit}(w_{ij}^k, \text{ with } i, j \in V) \qquad k \in S \tag{38}$$

$$w_{ij}^k \leq 1 - w_{00}^k \qquad k \in S, i, j \in C \tag{39}$$

$$\sum_{j \in C_{\mathcal{F}}} f_{0j} \leq m \tag{40}$$

$$\sum_{i \in C_{\mathcal{F}} \cup \{0\}, i \neq j} f_{ij} = \sum_{q_j \leq k \leq p_j} k z_j^k \qquad j \in C_{\mathcal{F}} \tag{41}$$

$$\sum_{i \in C_{\mathcal{F}} \cup \{0\}, i \neq j} f_{ij} = \sum_{l \in C_{\mathcal{F}} \cup \{0\}, l \neq j} f_{jl} \qquad j \in C_{\mathcal{F}} \cup \{0\} \tag{42}$$

$$f_{ij} \leq m y_{ij} \qquad i, j \in C_{\mathcal{F}} \cup \{0\}, i \neq j \tag{43}$$

$$y_{ij} \implies T_j \geq T_i + \sum_{q_j \leq k \leq p_j} \tau_j^k z_j^k \qquad i \in C_{\mathcal{F}} \cup \{0\}, j \in C_{\mathcal{F}}, i \neq j \tag{44}$$

$$0 \leq f_{ij} \leq m \qquad i, j \in C_{\mathcal{F}} \cup \{0\}, i \neq j \tag{45}$$

$$w_{ij}^k \in \{0; 1\} \qquad k \in S, i, j \in V \tag{46}$$

$$z_j^k \in \{0; 1\} \qquad j \in C_{\mathcal{F}}, q_j \leq k \leq p_j \tag{47}$$

$$y_{ij} \in \{0; 1\} \qquad i, j \in C_{\mathcal{F}} \cup \{0\}, i \neq j \tag{48}$$

$$T_j \geq 0 \qquad j \in C_{\mathcal{F}} \cup \{0\} \tag{49}$$

The constraints strictly follow the meaning already described for the $CP1$ model in Section 2.2. The changes reflect the presence of multiple trucks and are as follows. Inequalities (34) now constrain α to be equal to or larger than the length of the tour of each truck k. Equalities (36) now express that each drone-eligible customer has to be visited either by one of the trucks or the drones. The new constraints (37) state that customers in $C_{\mathcal{T}}$ cannot be visited by drones, and have to be serviced by exactly one truck. Constraints (38) are now independently defined for each truck k, dropping the concept of giant-tour introduced for the $CP2$ model. The new technical constraint (39) forces the circuit of a truck k to be empty once the relative variable w_{00}^k takes the value 1.

3.4. A 3-indices mixed integer linear programming model

We finally present the Mixed Integer Linear Programming (MILP) formulation of the PDSVRP-c, which is based on the 3-indices $CP3$ model of Section 3.3. For the sake of simplicity in the presentation of the model, we adopt a new variable u_j^k that for $j \in C$ takes value 1 if $k \in S$ serves customer j, 0 otherwise. In case $j = 0$, u_0^k takes instead value 1 if $k \in S$ is deployed (used), 0 otherwise. Notice that this variable can be defined as $u_j^k = 1 - w_{jj}^k$ in the logic of the $CP3$ model, but in the MILP model the loop variables w_{jj}^k are not used.

$$(MILP) \quad \min \alpha \tag{50}$$

$$\text{s.t.} \quad \alpha \geq \sum_{i \in V} \sum_{j \in V, i \neq j} t_{ij} w_{ij}^k \qquad k \in S \tag{51}$$

$$\alpha \geq T_j \qquad j \in C_{\mathcal{F}} \tag{52}$$

$$\sum_{k \in S} u_j^k + \sum_{q_j \leq k \leq p_j} z_j^k = 1 \qquad j \in C_{\mathcal{F}} \tag{53}$$

$$\sum_{k \in S} u_j^k = 1 \qquad j \in C_{\mathcal{T}} \tag{54}$$

$$u_j^k \leq u_0^k \qquad j \in C, k \in S \tag{55}$$

$$\sum_{i \in V, i \neq j} w_{ij}^k + \sum_{l \in V, l \neq j} w_{jl}^k = 2u_j^h \qquad j \in V, k \in S \tag{56}$$

$$\sum_{i,j \in H, i \neq j} w_{ij}^k \leq |H| - 1 \qquad H \subseteq C, k \in S \tag{57}$$

$$\frac{f_{ij}}{m} \leq y_{ij} \leq f_{ij} \qquad i, j \in C_{\mathcal{F}} \cup \{0\}, i \neq j \tag{58}$$

$$T_j + M(1 - y_{ij}) \geq T_i + \sum_{q_j \leq k \leq p_j} \tau_j^k z_j^k \quad i \in C_{\mathcal{F}} \cup \{0\}, j \in C_{\mathcal{F}}, i \neq j \tag{59}$$

$$0 \leq f_{ij} \leq m \qquad i, j \in C_{\mathcal{F}}, i \neq j \tag{60}$$

$$w_{ij}^k \in \{0; 1\} \qquad k \in S, i, j \in V, i \neq j \tag{61}$$

$$z_j^k \in \{0; 1\} \qquad j \in C_{\mathcal{F}}, q_j \leq k \leq p_j \tag{62}$$

$$u_j^k \in \{0; 1\} \qquad j \in C, k \in S \tag{63}$$

$$y_{ij} \in \{0; 1\} \qquad i, j \in C_{\mathcal{F}} \cup \{0\}, i \neq j \tag{64}$$

$$T_j \geq 0 \qquad j \in C_{\mathcal{F}} \cup \{0\} \tag{65}$$

The model minimizes the time to serve all the customers (50). Constraints (51) force α to be larger than any tour of the trucks, and inequalities (52) guarantee that α is larger than the completion time of any drone's mission time. Eqs. (53) assign customers from $C_{\mathcal{F}}$ to either a drone or a truck, while constraints (54) force the customers that can be visited only by a truck ($C_{\mathcal{T}}$) to receive such a visit. Inequalities (55) impose that a customer can be visited by a truck only if it is in use. Equalities (56) are flow conservation constraints for the truck tours. Inequalities (57) are subtour elimination constraints (Dantzig et al., 1954) and guarantee that truck tours are circuits including the depot. Notice that these constraints are exponential in number, depending on any possible subset H of C. In our implementation, they will be generated dynamically as described in Section 3.4.1 below. Constraints (58) refers to the flow of drones and guarantee that the number of

Table 1
Experimental results on the PDSTSP-c. Small instances.

Instance	RnR fast (Nguyen and Hà, 2023)[a]		RnR (Nguyen and Hà, 2023)[a]		MILP (Nguyen and Hà, 2023)[b]		*MILP*+(16)[c]			*CP*1[d]			*CP*1+ (16)[d]			Best
	UB	Sec_{bst}	UB	Sec_{bst}	[LB, UB]	Sec_{tot}	[LB, UB]	Sec_{tot}	Sec_{bst}	[LB, UB]	Sec_{tot}	Sec_{bst}	[LB, UB]	Sec_{tot}	Sec_{bst}	bounds
15-r-e	92	0.32	92	0.95	[92, 92]	8.65	[92, 92]	1215.79	850.00	[92, 92]	0.84	0.44	[92, 92]	1.14	1.10	92
15-rc-c	44	0.43	44	1.46	[31.74, 44]	–	[**44**, 44]	1837.95	1790.09	[40, 44]	–	28.79	[**44**, 44]	12.00	11.94	44
16-c-c	60	0.52	60	2.14	[60, 60]	2.61	[**60**, 60]	3.66	3.63	[**60**, 60]	2.95	2.90	[60, 60]	2.10	2.05	60
16-r-e	112	0.55	112	1.52	[112, 112]	63.86	[98.20, 112]	–	250.10	[112, 112]	2.00	1.94	[112, 112]	1.61	1.57	112
18-c-c	56	0.42	56	1.86	[38.72, 56]	–	[**56**, 56]	1258.06	50.89	[44, 56]	–	302.76	[**56**, 56]	42.22	42.13	56
18-r-e	96	0.59	96	1.79	[86.94, 96]	–	[87.86, 96]	–	2338.15	[92, 96]	–	55.83	[**96**, 96]	4.70	4.63	96
18-rc-c	58	0.54	57	2.40	[37.78, 57]	–	[56, 57]	–	3060.21	[38, *60*]	–	399.43	[**57**, 57]	1803.93	6.57	57
19-c-c	44	0.48	44	1.81	[28.06, 44]	–	[40.85, 44]	–	3302.58	[32, 44]	–	64.50	[**44**, 44]	24.91	11.00	44
20-c-c	43	0.60	43	2.54	[30.99, 44]	–	[39.42, 43]	–	2374.04	[**40**, 43]	–	102.82	[**40**, 43]	–	147.26	[40, 43]
20-r-c	64	0.43	64	1.91	[55.35, 64]	–	[61.80, 64]	–	3326.72	[56, 64]	–	276.15	[**64**, 64]	73.39	73.29	64
20-r-e	82	0.62	80	2.00	[62.59, 88]	–	[72.80, 82]	–	3237.98	[72, 80]	–	606.13	[**80**, 80]	38.11	38.00	80
20-rc-c	96	0.41	96	2.37	[96, 96]	0.90	[96, 96]	460.30	1.14	[96, 96]	6.48	6.42	[96, 96]	6.23	6.17	96
20-rc-e	100	0.46	100	1.09	[100, 100]	88.78	[90, 100]	–	292.98	[100, 100]	6.92	6.87	[100, 100]	4.70	4.65	100
21-c-c	62	0.48	62	2.25	[41.80, 64]	–	[44, 64]	–	2281.04	[*36*, *64*]	–	18.02	[**60**, 62]	–	58.44	[60, 62]
21-r-e	85	0.59	85	1.74	[59.52, 100]	–	[75.25, 88]	–	3572.98	[*49*, *88*]	–	1512.11	[**85**, 85]	940.55	128.94	85
23-c-e	80	0.60	80	2.49	[58.15, 80]	–	[58.15, 80]	–	2698.31	[**80**, 80]	0.84	0.78	[**80**, 80]	1.31	1.25	80
23-r-c	88	0.42	88	1.83	[88, 88]	3293.16	[84, 88]	–	3042.55	[88, 88]	218.07	217.97	[88, 88]	8.97	8.90	88
24-c-e	84	0.79	84	2.50	[78.4, 84]	–	[78.40, 84]	–	3567.08	[**84**, 84]	14.07	13.98	[**84**, 84]	11.05	10.97	84
24-r-e	112	0.52	112	1.57	[91.05, 112]	–	[101, 112]	–	1819.74	[108, 112]	–	4.76	[**112**, 112]	955.94	3.60	112
24-rc-c	72	0.73	71	4.06	[69.58, 88]	–	[69.58, 72]	–	14.13	[*68*, 71]	–	2245.17	[**70**, **70**]	190.03	189.86	70
25-c-c	56	0.60	56	2.92	[37.33, 56]	–	[37.83, 56]	–	694.26	[*35*, 56]	–	764.50	[**56**, 56]	44.85	38.62	56
25-r-e	106	0.96	104	3.14	[76.11, 120]	–	[95.73, 108]	–	3533.88	[*58*, *108*]	–	60.58	[**104**, 104]	288.19	288.04	104
25-rc-e	92	0.71	92	2.25	[66.99, 100]	–	[83.60, 96]	–	2636.83	[*60*, *97*]	–	90.45	[**92**, 92]	113.76	113.63	92
26-r-c	103	0.53	103	2.58	[95.26, 128]	–	[100.18, 103]	–	3409.09	[*84*, *104*]	–	2746.86	[**101**, 103]	–	107.06	[101, 103]
27-c-c	84	0.49	84	2.18	[83.23, 84]	–	[64.72, 84]	–	2672.84	[84, 84]	1.91	1.85	[**84**, 84]	1.80	1.75	84
27-c-e	68	0.72	68	6.27	[42.04, 68]	–	[42.04, 68]	–	1429.86	[*31*, 68]	–	1.76	[**68**, 68]	33.23	1.36	68
27-rc-c	100	0.77	100	5.30	[100, 100]	721.25	[85.34, 100]	–	2915.17	[100, 100]	394.35	394.19	[100, 100]	71.17	71.06	100
27-rc-e	84	0.79	84	2.70	[59.52, 100]	–	[64.29, 88]	–	1202.70	[*42*, *88*]	–	2066.04	[**84**, 84]	576.56	54.8	84
29-rc-e	116	0.72	116	1.69	[97.71, 124]	–	[109.75, 116]	–	2214.71	[**116**, 116]	654.58	654.40	[**116**, 116]	8.60	8.53	116
30-c-c	96	0.65	96	3.76	[83.78, 96]	–	[83.78, 96]	–	1827.64	[**96**, 96]	2.30	2.22	[**96**, 96]	3.22	3.14	96
Average	81.17	0.58	80.97	2.44	[68.69, 84.83]	–	[72.42, 81.63]	–	2013.71	[69.77, 81.70]	–	421.69	[80.70, 80.93]	–	48.01	[80.70, 80.93]

[a] CPU AMD Ryzen 3700X - 4 × 3.6 GHz, 4 × 4.4 GHz, 16 threads; RAM 32 GB; best results over 30 runs.

[b] CPU AMD Ryzen 3700X - 4 × 3.6 GHz, 4 × 4.4 GHz, 16 threads; RAM 32 GB; CPLEX 12.1; 3600 s time limit.

[c] CPU Intel Core i7 12700F - 4 × 3.6 GHz, 8 × 4.9 GHz, 20 threads; RAM 32 GB; Gurobi 10.0; 3600 s time limit.

[d] CPU Intel Core i7 12700F - 4 × 3.6 GHz, 8 × 4.9 GHz, 20 threads; RAM 32 GB; OR-Tools CP-SAT 9.6; 3600 s time limit.

Table 2
Experimental results on the PDSTSP-c. Large instances.

Instance	RnR fast (Nguyen and Hà, 2023)[a]		RnR (Nguyen and Hà, 2023)[a]		*CP*1[b]			*CP*1+(16)[b]			Best
	UB	Sec_{bst}	UB	Sec_{bst}	[LB, UB]	Sec_{tot}	Sec_{bst}	[LB, UB]	Sec_{tot}	Sec_{bst}	bounds
50-r-e	120	2.45	116	20.00	[49, *140*]	–	2123.27	[**116** , *128*]	–	2083.72	116
53-r-e	136	2.23	132	15.17	[68, *160*]	–	2631.50	[**132**, *140*]	–	3177.59	132
66-rc-e	128	3.13	124	16.61	[80, *168*]	–	393.29	[**124**, *132*]	–	2927.03	124
67-c-c	76	2.32	76	29.05	[68, *80*]	–	133.82	[**73**, *80*]	–	403.51	[73, 76]
68-rc-c	79	2.12	76	20.45	[72, *112*]	–	2488.01	[**76**, 76]	3089.06	3088.37	76
76-c-c	52	2.15	52	10.46	[40, 52]	–	26.87	[**52**, 52]	379.86	31.99	52
82-c-e	64	2.63	64	18.07	[**64**, 64]	33.30	30.55	[**64**, 64]	59.21	58.42	64
82-rc-c	108	2.65	104	27.42	[100, *140*]	–	2373.65	[**104**, *144*]	–	3269.15	104
88-c-e	108	3.56	108	9.62	[**108**, 108]	51.03	50.09	[**108**, 108]	92.47	91.56	108
91-r-c	128	3.64	124	72.49	[116, *188*]	–	2887.23	[**120**, *164*]	–	3302.76	[120, 124]
99-rc-c	108	3.64	100	51.21	[80, *168*]	–	1640.06	[**98**, *164*]	–	2575.79	[98, 100]
101-r-c	124	3.98	120	99.71	[96, *176*]	–	3456.53	[**114**, *180*]	–	3338.22	[114, 120]
103-rc-c	128	4.38	124	94.25	[108, *176*]	–	1967.67	[**120**, *164*]	–	2147.63	[120, 124]
105-rc-e	124	5.78	120	62.71	[80, *184*]	–	903.40	[**109**, *132*]	–	1831.25	[109, 120]
108-rc-e	144	5.56	136	61.71	[112, *188*]	–	3088.15	[**134**, *188*]	–	2521.07	[134, 136]
114-rc-c	100	4.69	96	62.73	[68, *152*]	–	2591.56	[**94**, *148*]	–	2821.57	[94, 96]
121-rc-e	128	6.41	124	68.74	[108, *180*]	–	2865.31	[**121**, *160*]	–	1201.60	[121, 124]
126-r-c	161	5.89	160	120.93	[104, *228*]	–	1457.72	[**151**, *216*]	–	3028.00	[151, 160]
126-rc-e	148	7.38	144	71.04	[124, *196*]	–	2945.63	[**136**, *188*]	–	1053.63	[136, 144]
144-rc-r	132	6.52	128	172.75	[120, *188*]	–	2801.00	[**122**, *204*]	–	3289.31	[122, 128]
154-c-c	72	7.17	72	62.96	[**68**, 72]	–	63.64	[**68**, 72]	–	61.68	[68, 72]
165-r-c	176	7.61	164	280.18	[118, *292*]	–	2386.33	[**140**, *312*]	–	3528.39	[140, 164]
167-r-e	200	10.55	188	228.31	[72, *296*]	–	3309.32	[**160**, *304*]	–	2209.10	[160, 188]
173-r-c	180	8.64	164	373.43	[92, *312*]	–	2439.90	[**141**, *280*]	–	2184.87	[141, 164]
173-rc-r	144	9.22	133	135.20	[51, *208*]	–	2539.43	[**115**, *208*]	–	3556.41	[115, 133]
181-r-e	232	11.20	224	196.09	[125, *332*]	–	3252.35	[**199**, *348*]	–	3566.96	[199, 224]
185-c-c	96	11.26	96	61.53	[**96**, 96]	1279.96	1276.68	[**96**, 96]	622.29	619.07	96
187-rc-e	200	12.67	196	119.95	[78, *284*]	–	2464.47	[**167**, *288*]	–	3228.65	[167, 196]
198-c-c	64	11.38	64	94.12	[**64**, *68*]	–	82.40	[**64**, *68*]	–	155.40	64
200-r-e	224	13.88	212	368.97	[40, *324*]	–	3541.42	[**162**, *328*]	–	2132.93	[162, 212]
Average	129.47	6.16	124.70	100.86	[85.63, 177.73]	–	1940.38	[116.00, 171.20]	–	2116.19	[116.00, 124.70]

[a] CPU AMD Ryzen 3700X - 4 × 3.6 GHz, 4 × 4.4 GHz, 16 threads; RAM 32 GB; best results over 30 runs.

[b] CPU Intel Core i7 12700F - 4 × 3.6 GHz, 8 × 4.9 GHz, 20 threads; RAM 32 GB; OR-Tools CP-SAT 9.6; 3600 s time limit.

Table 3
Experimental results on the PDSVRP-c. Small instances, 2 trucks.

Instance	$MILP$+(16)[a]			$CP2$+(16)[b]			$CP3$+(16)[b]			Best
	[LB, UB]	Sec_{tot}	Sec_{bst}	[LB, UB]	Sec_{tot}	Sec_{bst}	[LB, UB]	Sec_{tot}	Sec_{bst}	bounds
15-r-e	[**92** , **92**]	2654.11	745.82	[**92**, **92**]	157.10	0.21	[**92**, **92**]	2.49	0.45	92
15-rc-c	[**33**, **33**]	9.42	7.46	[**33**, **33**]	0.92	0.52	[**33**, **33**]	5.14	2.92	33
16-c-c	[**40**, **40**]	50.04	3.42	[**40**, **40**]	6.43	0.29	[**40**, **40**]	7.17	2.26	40
16-r-e	[104, **108**]	–	783.12	[**108**, **108**]	83.49	0.69	[**108**, **108**]	5.03	1.75	108
18-c-c	[**44**, **44**]	22.76	22.72	[**44**, **44**]	3.09	2.40	[**44**, **44**]	7.45	5.98	44
18-r-e	[**92**, **92**]	1153.19	197.42	[**92**, **92**]	43.36	1.57	[**92**, **92**]	6.92	4.52	92
18-rc-c	[44, **46**]	–	3569.88	[**46**, **46**]	459.47	161.29	[**46**, **46**]	401.32	45.46	46
19-c-c	[34, **36**]	–	3220.87	[**36**, **36**]	7.85	4.15	[**36**, **36**]	15.41	3.84	36
20-c-c	[**40**, **40**]	907.04	5.25	[**40**, **40**]	4.35	2.23	[**40**, **40**]	4.26	1.75	40
20-r-c	[**48**, **48**]	2023.70	1886.70	[**48**, **48**]	454.46	9.19	[**48**, **48**]	62.72	43.56	48
20-r-e	[63, 76]	–	2707.00	[**72**, **72**]	462.23	331.95	[**72**, **72**]	27.42	23.27	72
20-rc-c	[63, **64**]	–	1977.44	[58, **64**]	–	1.15	[**64**, **64**]	14.57	8.09	64
20-rc-e	[72, **80**]	–	3031.82	[**64**, **80**]	–	1.49	[**80**, **80**]	24.93	7.82	80
21-c-c	[**40**, **40**]	59.12	8.87	[**40**, **40**]	9.94	1.36	[**40**, **40**]	11.60	2.01	40
21-r-e	[51, **76**]	–	3348.26	[**76**, **76**]	475.63	4.08	[**76**, **76**]	82.46	9.41	76
23-c-e	[44, **80**]	–	0.98	[42, **80**]	–	0.98	[**80**, **80**]	17.85	0.67	80
23-r-c	[**60**, **60**]	1536.90	1379.17	[57, **60**]	–	9.64	[**60**, **60**]	475.01	10.69	60
24-c-e	[56, **60**]	–	505.00	[**60**, **60**]	116.75	34.53	[**60**, **60**]	54.04	53.77	60
24-r-e	[68, **100**]	–	0.89	[67, **100**]	–	3.09	[**100**, **100**]	68.82	35.52	100
24-rc-c	[49, **52**]	–	42.34	[**52**, **52**]	1350.35	25.78	[**52**, **52**]	307.80	82.07	52
25-c-c	[**40**, **40**]	2523.46	1988.78	[**40**, **40**]	899.80	26.80	[**40**, **40**]	79.98	63.91	40
25-r-e	[72, 92]	–	693.55	[85, **88**]	–	35.41	[**88**, **88**]	175.75	44.11	88
25-rc-e	[64, 80]	–	239.90	[59, **76**]	–	80.16	[**76**, **76**]	189.81	3.42	76
26-r-c	[68, **70**]	–	1381.35	[65, **70**]	–	20.87	[**70**, **70**]	2701.81	700.35	70
27-c-c	[40, **52**]	–	83.87	[42, **52**]	–	2.32	[**52**, **52**]	37.07	16.67	52
27-c-e	[7, **68**]	–	6.18	[**68**, **68**]	2741.90	0.65	[**68**, **68**]	130.71	1.56	68
27-rc-c	[64, **72**]	–	58.13	[**64**, **72**]	–	8.96	[**72**, **72**]	79.95	53.96	72
27-rc-e	[36, 80]	–	171.86	[47, **76**]	–	20.22	[**76**, **76**]	198.05	46.71	76
29-rc-e	[62, **100**]	–	3092.20	[65, **100**]	–	13.71	[**100**, **100**]	59.26	48.57	100
30-c-c	[36, **64**]	–	2935.06	[48, **64**]	–	2.09	[**64**, **64**]	39.64	2.99	64
Average	[54.20, 66.17]	–	1136.51	[58.33, 65.63]	–	26.93	[65.63, 65.63]	176.48	44.27	65.63

[a] CPU Intel Core i7 12700F - 4 × 3.6 GHz, 8 × 4.9 GHz, 20 threads; RAM 32 GB; Gurobi 10.0; 3600 s time limit.
[b] CPU Intel Core i7 12700F - 4 × 3.6 GHz, 8 × 4.9 GHz, 20 threads; RAM 32 GB; OR-Tools CP-SAT 9.6; 3600 s time limit.

drones going from customer i to j must be lower than m, only if there is a flow from i to j. Inequalities (59) regulate completion of the service time for the customers visited by the drones. Finally, constraints (60)–(65) define the domain of the variables.

3.4.1. *Separation of the subtour elimination constraints*

The subtour elimination constraints (57) are dynamically added to the MILP as *Lazy constraints* (available in the most popular MILP solvers). Specifically, the solver starts by disregarding the constraints declared *lazy* and once a feasible integer solution is found it invokes a user defined separation procedure. In our case, since the solution on hand is integer, the separation is a simple $O(|E|)$ exploration of the graphs $G^k = (V, E^k)$ with $E^k = \{(i,j) \in E : w^k_{ij} = 1\}$ to look for subtours not involving the depot.

4. Experimental results

All the models presented in previous sections have been coded in Python 3.11.2. The Constraint Programming models of Sections 2 and 3 have been solved via the CP-SAT solver of Google OR-Tools 9.6 (Perron and Furnon, 2023) while the Mixed Integer Linear model of Section 3 has been solved with Gurobi 10.0 (Gurobi Optimization, 2023).

The outcome of the experimental campaign is discussed in the remainder of this section and is organized according to the different problems tackled. Tables 1–8 report, for each instance: (i) the instance name; (ii) the lower bound eventually produced and the best heuristic solution ([LB, UB]); (iii) the computing time to find the best heuristic solution (Sec_{bst}); (iv) the eventual computing time to prove optimality (Sec_{tot}); (v) a final summary column (Best bounds) containing the current state-of-the-art results of each instance for easing future research.

In addition, we use a dash whenever a result is not retrieved or the time limit is reached, and we mark in *italics* the results of our models not matching nor improving best-known bounds while in **bold** those producing new best bounds. A line with the average of the relevant column is present in the bottom of each table, to ease the interpretation of the results. Hardware configurations, solvers used, experimental settings and time limits are finally reported in the notes of the tables for each approach.

4.1. *Benchmark instances*

To evaluate the performance of the proposed models for both the PDSTSP-c and the new PDSVRP-c, we consider the instances originally introduced in Nguyen and Hà (2023) for the PDSTSP-c, and available at http://orlab.com.vn/home/download. The number n of customers varies from 15 to 200 (first number of the instance name) and the instances are divided into small ($n \leq 30$) and large ($n > 30$). The number m of drones available varies in the range $[3, 6]$ for the small instances and $[5, 10]$ for the large ones. The traveling distances for trucks are computed using Manhattan distances and a speed of 30 km/h, while drones follow the Euclidean distance and the optimal travel times (rounded up to the nearest integer) are pre-calculated for

Table 4
Experimental results on the PDSVRP-c. Small instances, 3 trucks.

Instance	$MILP$+(16)[a]			$CP2$+(16)[b]			$CP3$+(16)[b]			Best
	[LB, UB]	Sec_{tot}	Sec_{bst}	[LB, UB]	Sec_{tot}	Sec_{bst}	[LB, UB]	Sec_{tot}	Sec_{bst}	bounds
15-r-e	[72, **92**]	–	3401.20	[**92, 92**]	331.05	0.19	[**92, 92**]	4.44	0.36	92
15-rc-c	[**32, 32**]	11.08	11.01	[**32, 32**]	1.43	1.27	[**32, 32**]	5.95	3.66	32
16-c-c	[**36, 36**]	2.20	2.18	[**36, 36**]	1.21	1.17	[**36, 36**]	6.01	5.41	36
16-r-e	[68, **108**]	–	2332.97	[**108, 108**]	290.70	1.05	[**108, 108**]	6.55	1.15	108
18-c-c	[**44, 44**]	25.27	25.21	[**44, 44**]	3.02	1.61	[**44, 44**]	6.37	2.12	44
18-r-e	[88, **92**]	–	614.09	[**92, 92**]	22.3	0.92	[**92, 92**]	12.00	2.10	92
18-rc-c	[**40, 40**]	18.60	18.53	[**40, 40**]	5.22	5.09	[**40, 40**]	30.65	24.69	40
19-c-c	[29, **36**]	–	3554.00	[**36, 36**]	2.26	0.86	[**36, 36**]	19.25	3.59	36
20-c-c	[**40, 40**]	383.19	139.78	[**40, 40**]	1.90	0.72	[**40, 40**]	7.35	1.27	40
20-r-c	[**37, 37**]	1202.72	582.24	[**37, 37**]	8.30	3.81	[**37, 37**]	99.17	36.74	37
20-r-e	[44, **72**]	–	45.40	[**72, 72**]	443.37	7.40	[**72, 72**]	17.12	9.27	72
20-rc-c	[**48, 48**]	604.42	8.91	[**48, 48**]	99.53	3.15	[**48, 48**]	94.88	26.53	48
20-rc-e	[60, **68**]	–	595.63	[**68, 68**]	122.18	42.00	[**68, 68**]	45.74	8.25	68
21-c-c	[**36, 36**]	28.87	6.92	[**36, 36**]	4.96	4.75	[**36, 36**]	16.50	7.81	36
21-r-e	[34, **76**]	–	689.44	[**76, 76**]	891.56	11.30	[**76, 76**]	427.99	28.05	76
23-c-e	[36, **80**]	–	3.88	[66, **80**]	–	0.94	[**80, 80**]	25.44	0.93	80
23-r-c	[**48, 48**]	203.22	44.94	[**48, 48**]	20.92	14.24	[**48, 48**]	590.10	135.03	48
24-c-e	[56, **60**]	–	2000.96	[**60, 60**]	135.61	3.35	[**60, 60**]	28.56	11.45	60
24-r-e	[47, **100**]	–	8.28	[80, **100**]	–	1.72	[**100, 100**]	64.88	8.77	100
24-rc-c	[41, **44**]	–	3456.94	[**44, 44**]	227.22	77.94	[**44, 44**]	926.73	387.43	44
25-c-c	[30, 40]	–	97.39	[**37, 37**]	70.58	31.83	[**37, 37**]	107.30	42.94	37
25-r-e	[57, 96]	–	3298.63	[59, **85**]	–	44.60	[**85, 85**]	404.43	242.54	85
25-rc-e	[52, 69]	–	2601.65	[65, **66**]	–	141.25	[**66, 66**]	356.83	61.26	66
26-r-c	[**56, 56**]	180.88	67.14	[52, **56**]	–	1399.21	[55, **56**]	–	70.15	56
27-c-c	[**36, 36**]	1367.86	936.48	[**36, 36**]	577.74	3.69	[**36, 36**]	121.70	39.48	36
27-c-e	[8, **68**]	–	4.92	[**68, 68**]	2315.01	1.31	[**68, 68**]	437.75	1.71	68
27-rc-c	[**60, 60**]	223.20	213.85	[**60, 60**]	6.79	6.01	[**60, 60**]	74.79	63.90	60
27-rc-e	[28, **76**]	–	155.68	[56, **76**]	–	8.02	[**76, 76**]	520.40	14.61	76
29-rc-e	[53, 108]	–	3337.59	[72, **100**]	–	23.67	[**100, 100**]	56.98	35.42	100
30-c-c	[26, **38**]	–	3182.76	[**38, 38**]	96.12	5.63	[**38, 38**]	145.11	15.99	38
Average	[44.73, 61.20]		1047.95	[56.60, 60.37]	–	61.62	[60.33, 60.37]	–	43.09	60.37

[a] CPU Intel Core i7 12700F - 4 × 3.6 GHz, 8 × 4.9 GHz, 20 threads; RAM 32 GB; Gurobi 10.0; 3600 s time limit.

[b] CPU Intel Core i7 12700F - 4 × 3.6 GHz, 8 × 4.9 GHz, 20 threads; RAM 32 GB; OR-Tools CP-SAT 9.6; 3600 s time limit.

each collaborative cluster of k drones. The interested reader can find all the details of the instances in Nguyen and Hà (2023).

For generating PDSVRP-c instances we used the same set of benchmarks and added the number s of trucks chosen in the range $[2, 3]$ for small instances and in $[2, 5]$ for large instances.

4.2. PDSTSP-c

In this section, we aim at comparing the results obtained by solving the $CP1$ model described in Section 2.2, with and without the valid inequality (16). The results are summarized in Table 1 for the small instances and in Table 2 for the large ones. We compare $CP1$ with the methods introduced in Nguyen and Hà (2023), namely a Mixed Integer Linear Programming (MILP) model solved with IBM CPLEX 12.1 (IBM ILOG, 2023) and two versions of a *Ruin&Recreate* metaheuristic: RnR fast and RnR. Notice that the results of the MILP model in Nguyen and Hà (2023) are only available for small instances and those reported for the *Ruin&Recreate* methods are the best over 30 runs. To fully understand the impact of inequality (16), we also considered the MILP model described in Section 3.4 for the PDSVRP-c and run it with $s = 1$ (one truck only) as well as the inequality (16). This method is run on small instances only, since a previous study (Nguyen and Hà, 2023) showed MILP models are not suitable for the large instances.

We are not aware of other existing methods to deal with this problem.

From the results displayed in Table 1, we see that inequality (16) is very effective in improving the performance of the models, both $CP1$ and $MILP$. Given this evidence, we will always consider inequalities (16) for the next experiments.

Table 1 reveals that the CP-based approach matches (or improves in the case of instance 24-rc-c) all the best-known heuristic solutions and outperforms the exact MILP method, both in terms of quality and times. Additionally, we observe how the $CP1$+(16) improves several lower bounds and closes all but three instances.

To better highlight the differences between the MILP and the CP methods, we report in Fig. 2 their percentage optimality gaps, calculated as $100 \cdot \frac{UB-LB}{UB}$, and in Fig. 3 their required time to find the best solution (Sec_{bst}). Fig. 2 shows that the $CP1$ model clearly leads to lower optimality gaps than the $MILP$ model with a time limit of 3600 s, the latter also demonstrating scalability issues on larger instances as remarked by its linearly increasing trend (dashed line). Whereas Fig. 3 shows that the $CP1$ model is substantially faster in retrieving the best heuristic solution. These results suggest that the Constraint Programming-based approach has great potential for the PDSTSP-c problem.

Moving to the larger instances reported in Table 2 we observe that $CP1$+(16) is able to provide, for the first time, valid lower bounds for all instances. Moreover 10 over 30 bounds equal to the best known solution, hence proving for the first time the optimality of these solutions. In the remaining instances the gaps, between the lower bound and the heuristic solution is generally small. However the upper bound provided by the CP models is not competitive with respect to that of the metaheuristic methods. Also the running times are larger, although it is worth to observe once again that for the *RnR* methods the best results over 30 runs is provided, making the timing presented less fair.

Table 5
Experimental results on the PDSVRP-c. Large instances, 2 trucks.

Instance	$CP2$+(16)[a]		$CP3$+(16)[a]		Best
	[LB, UB]	Sec_{bst}	[LB, UB]	Sec_{bst}	bounds
50-r-e	[**65**, **116**]	206.57	[63, 120]	168.48	[65, 116]
53-r-e	[77, **112**]	894.09	[**82**, 128]	1756.80	[82, 112]
66-rc-e	[72, **112**]	1829.73	[**73**, 136]	866.28	[73, 112]
67-c-c	[**38**, **52**]	22.33	[31, **52**]	827.01	[38, 52]
68-rc-c	[50, **56**]	3332.51	[**52**, 104]	3088.50	[52, 56]
76-c-c	[**26**, **36**]	20.60	[16, 40]	185.95	[26, 36]
82-c-e	[**32**, **64**]	25.41	[17, **64**]	73.68	[32, 64]
82-rc-c	[**62**, **116**]	2974.84	[56, 132]	2615.62	[62, 116]
88-c-e	[54, **84**]	298.18	[**58**, 112]	49.28	[58, 84]
91-r-c	[**75**, **152**]	405.02	[**75**, 160]	2249.67	[75, 152]
99-rc-c	[**63**, **96**]	2083.65	[51, 144]	564.95	[63, 96]
101-rc	[**71**, 164]	2921.49	[53, **152**]	1731.45	[71, 152]
103-rc-c	[**69**, **124**]	2603.95	[52, 128]	2912.93	[69, 124]
105-rc-e	[**65**, **136**]	2170.84	[57, 148]	1383.74	[65, 136]
108-rc-e	[**79**, 172]	1683.07	[70, **160**]	831.13	[79, 160]
114-rc-c	[**58**, **124**]	3417.23	[49, 140]	411.62	[58, 124]
121-rc-e	[**70**, 156]	647.12	[56, **152**]	2088.27	[70, 152]
126-rc-e	[**87**, 220]	3115.59	[67, **184**]	1956.96	[87, 184]
126-r-c	[**78**, 160]	2679.11	[56, **156**]	1448.65	[78, 156]
144-rc-c	[**67**, 272]	2610.83	[47, **168**]	3103.46	[67, 168]
154-c-c	[**35**, –]	–	[8, **72**]	279.16	[35, 72]
165-r-c	[**88**, –]	–	[67, **224**]	3544.74	[88, 224]
167-r-e	[**100**, –]	–	[74, **256**]	3151.22	[100, 256]
173-r-c	[**85**, **204**]	2929.34	[59, 240]	2251.40	[85, 204]
173-rc-c	[**79**, –]	–	[48, **180**]	1797.98	[79, 180]
181-r-e	[**112**, –]	–	[78, **252**]	3388.32	[112, 252]
185-c-c	[**48**, –]	–	[24, **96**]	316.31	[48, 96]
187-rc-e	[**100**, 308]	3391.71	[65, **212**]	1567.38	[100, 212]
198-c-c	[**32**, –]	–	[12, **64**]	271.52	[32, 64]
200-r-e	[**105**, –]	–	[68, **324**]	2072.94	[105, 324]
Average	[68.07, –]	–	[52.80, 150.00]	1565.18	[68.47, 141.20]

[a] CPU Intel Core i7 12700F - 4 × 3.6 GHz, 8 × 4.9 GHz, 20 threads; RAM 32 GB; OR-Tools CP-SAT 9.6; 3600 s time limit.

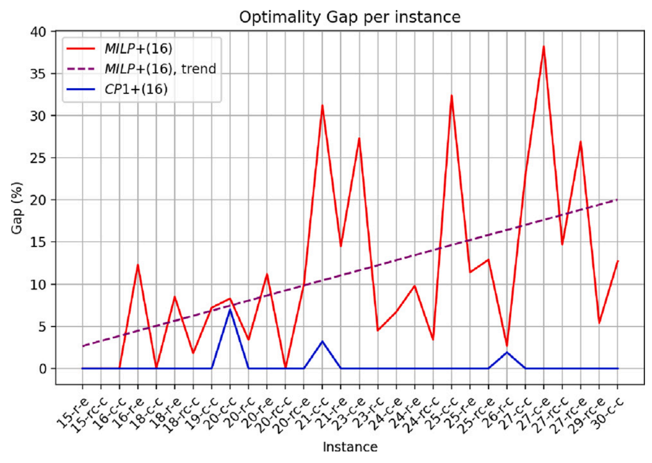

Fig. 2. The optimality gap in percentage for the $MILP$+(16) and $CP1$+(16) on small PDSTSP-c instances.

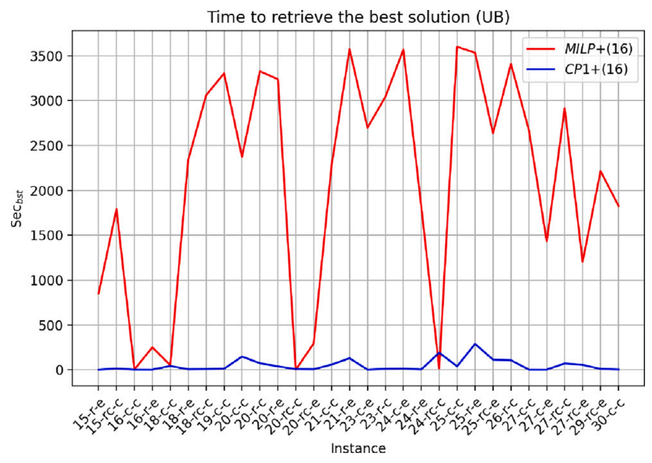

Fig. 3. The time required in seconds by the $MILP$+(16) and $CP1$+(16) to retrieve the best heuristic solution (UB).

4.3. PDSVRP-c

In this section, we aim at comparing the performance of the models $CP2$, $CP3$ and $MILP$ described in Sections 3.2–3.4 for the PDSVRP-c. Their results are summarized in Tables 3 and 4 for the small instances, covering respectively 2 and 3 trucks, and in Tables 5–8 for the large instances, using respectively 2, 3, 4, and 5 trucks. The PDSVRP-c is first introduced in this paper, so no comparison is available with methods from other authors.

Tables 3 and 4 suggest that solving the $MILP$ is less effective than solving the CP models, both in terms of bounds provided and computing time. The only remarkable exception is instance 26-r-c with 3 trucks (Table 4), which is closed by the former but not by the latters.

One can also observe that the performances of $MILP$ degrade with the increasing of the instance size, much more than that of the CP methods. After some test with larger instances (not reported here), and considering the analogous decision made for the PDSTSP-c

Table 6
Experimental results on the PDSVRP-c. Large instances, 3 trucks.

Instance	$CP2$+(16)[a]		$CP3$+(16)[a]		Best
	[LB, UB]	Sec_{bst}	[LB, UB]	Sec_{bst}	bounds
50-r-e	[**48**, **112**]	79.65	[47, 112]	411.11	[48, 112]
53-r-e	[**56**, **96**]	860.00	[51, 112]	2074.85	[56, 96]
66-rc-e	[**53**, **108**]	282.64	[38, 116]	139.49	[53, 108]
67-c-c	[**27**, **52**]	32.82	[9, **52**]	353.35	[27, 52]
68-rc-c	[**39**, **56**]	756.18	[34, 104]	655.52	[39, 56]
76-c-c	[**18**, **24**]	42.28	[12, 52]	81.65	[18, 24]
82-c-e	[**22**, **64**]	21.88	[8, **64**]	26.79	[22, 64]
82-rc-c	[**47**, **80**]	1727.16	[38, 128]	312.65	[47, 80]
88-c-e	[**36**, **76**]	375.27	[32, 104]	587.30	[36, 76]
91-r-c	[**56**, **120**]	3036.11	[42, 148]	726.56	[56, 120]
99-rc-c	[**47**, **64**]	2650.32	[29, 128]	196.67	[47, 64]
101-rc	[**52**, **128**]	2645.43	[36, 144]	2520.98	[52, 128]
103-rc-c	[**49**, **96**]	2229.84	[32, 136]	2332.89	[49, 96]
105-rc-e	[**49**, **120**]	877.50	[34, 132]	907.44	[49, 120]
108-rc-e	[**58**, 184]	1969.45	[37, **160**]	1273.20	[58, 160]
114-rc-c	[**44**, **80**]	1676.32	[32, 112]	466.45	[44, 80]
121-rc-e	[**52**, **124**]	2820.31	[40, 152]	1701.91	[52, 124]
126-rc-e	[**63**, **136**]	2839.24	[44, 164]	2663.93	[63, 136]
126-r-c	[**56**, **140**]	2191.71	[38, 148]	3114.44	[56, 140]
144-rc-c	[**50**, **132**]	3362.32	[35, 160]	2396.79	[50, 132]
154-c-c	[**24**, **36**]	195.44	[8, 68]	1368.67	[24, 36]
165-r-c	[**68**, –]	–	[50, **212**]	3120.16	[68, 212]
167-r-e	[**73**, –]	–	[54, **204**]	2112.65	[73, 204]
173-r-c	[**65**, –]	–	[45, **212**]	2004.93	[65, 212]
173-rc-c	[**58**, 172]	2994.35	[37, **168**]	2592.28	[58, 168]
181-r-e	[**82**, –]	–	[55, **216**]	3342.10	[82, 216]
185-c-c	[**32**, –]	–	[14, **96**]	1280.86	[32, 96]
187-rc-e	[**74**, –]	–	[46, **212**]	2849.33	[74, 212]
198-c-c	[**22**, **36**]	158.92	[8, 68]	108.97	[22, 36]
200-r-e	[**77**, –]	–	[48, **252**]	1817.10	[77, 252]
Average	[49.90, –]	–	[34.43, 137.87]	1451.37	[49.90, 120.40]

[a] CPU Intel Core i7 12700F - 4 × 3.6 GHz, 8 × 4.9 GHz, 20 threads; RAM 32 GB; OR-Tools CP-SAT 9.6; 3600 s time limit.

in Nguyen and Hà (2023), we decided to not consider the $MILP$ for the experiments on large instances (Tables 5–8).

The results of the two CP models suggest that the 3-indices formulation ($CP3$) is superior, being able to close all the instances but one. The 2-indices model appears slower even though the quality of its upper bounds is the same of $CP3$. This highlights that the weakness of the $CP2$ model is in the computation of the lower bound.

The results reported in Tables 5–8 for large instances (notice that the column Sec_{tot} has been omitted, since no optimality is proven) and a varying number of trucks lead to the following observations. The model with 3 indices, which performs the best on small instances (see Tables 3 and 4), is instead performing worse than the 2-indices model on large ones, especially in terms of retrieved lower bounds. This might suggest that handling multiple truck tours with the *MultipleCircuit* command becomes effective when tours are complex.

There are however a few exceptions where the 3-indices model is better either in terms of lower or upper bounds. Specifically, the $CP3$ model appears to be more consistent in instances with many customers and a few trucks, in which the 2-indices model often fails to produce any feasible solution. This might indicate that the models are approaching their natural limit.

5. Conclusions

In this paper, we have discussed several advances for the Parallel Drone Scheduling Traveling Salesman Problem with cooperative drones. In particular, we have proposed a Constraint Programming model coupled with a valid inequality that allows us to find improved lower and upper bounds for the instances proposed in the literature. Additionally, we demonstrated that the proposed valid inequality can be used to enhance the performance of other methods such as MILP models.

We have also extended the problem into the new Parallel Drone Scheduling Vehicle Routing Problem with cooperative drones, where several trucks are available. For this new extension, we have proposed two alternative Constraint Programming models and a Mixed Integer Programming model. Experimental results suggest that Constraint Programming guarantees better performance, but seems to have scaling issues on large instances, leaving room for future studies on heuristic approaches tailored to the problem.

CRediT authorship contribution statement

Roberto Montemanni: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Mauro Dell'Amico:** Conceptualization, Methodology, Writing – original draft. **Andrea Corsini:** Formal analysis, Investigation, Methodology, Software, Writing – original draft.

Data availability

Data will be made available on request.

Acknowledgments

The authors are grateful to Minh Hoàng Hà and Minh Anh Nguyen for the useful discussions, and for having provided the values of the optimized travel times for the drones.

We finally thank the anonymous reviewers for their careful reading and precious suggestions.

Table 7
Experimental results on the PDSVRP-c. Large instances, 4 trucks.

Instance	$CP2$+(16)[a]		$CP3$+(16)[a]		Best
	[LB, UB]	Sec_{bst}	[LB, UB]	Sec_{bst}	bounds
50-r-e	[**46**, **104**]	649.55	[35, 112]	213.25	[46, 104]
53-r-e	[**50**, **96**]	1068.12	[38, 112]	548.64	[50, 96]
66-rc-e	[**41**, **104**]	3493.30	[34, 108]	1019.85	[41, 104]
67-c-c	[**21**, **48**]	25.68	[8, 52]	1297.51	[21, 48]
68-rc-c	[**32**, **52**]	410.57	[29, 88]	296.61	[32, 52]
76-c-c	[**14**, **24**]	44.33	[12, 56]	24.41	[14, 24]
82-c-e	[**18**, **64**]	18.15	[8, **64**]	20.44	[18, 64]
82-rc-c	[**38**, **68**]	2275.00	[31, 124]	194.26	[38, 68]
88-c-e	[28, **76**]	76.88	[**32**, 108]	1177.87	[32, 76]
91-r-c	[**45**, **96**]	3019.26	[32, 156]	248.69	[45, 96]
99-rc-c	[**37**, **68**]	1058.95	[24, 120]	322.23	[37, 68]
101-rc	[**42**, **76**]	3171.49	[30, 144]	2589.25	[42, 76]
103-rc-c	[**39**, **80**]	1490.89	[26, 140]	521.14	[39, 80]
105-rc-e	[**39**, **116**]	261.16	[26, 132]	1691.38	[39, 116]
108-rc-e	[**46**, **124**]	454.82	[28, 152]	3163.13	[46, 124]
114-rc-c	[**35**, **88**]	2564.47	[26, 120]	1369.36	[35, 88]
121-rc-e	[**42**, **104**]	3185.34	[29, 144]	410.13	[42, 104]
126-rc-e	[**50**, **132**]	3362.14	[35, 164]	2600.11	[50, 132]
126-r-c	[**45**, **116**]	1094.09	[28, 140]	729.26	[45, 116]
144-rc-c	[**40**, **128**]	3013.06	[25, 144]	2451.13	[40, 128]
154-c-c	[**18**, **40**]	949.21	[8, 72]	63.77	[18, 40]
165-r-c	[**54**, **192**]	243.15	[40, **192**]	3124.08	[54, 192]
167-r-e	[**58**, **176**]	3277.00	[42, 196]	1489.17	[58, 176]
173-r-c	[**54**, 352]	3435.45	[36, **192**]	3070.29	[54, 192]
173-rc-c	[**46**, **116**]	1650.91	[29, 164]	3368.14	[46, 116]
181-r-e	[**65**, 268]	2937.67	[42, **208**]	3048.86	[65, 208]
185-c-c	[**24**, **48**]	2350.91	[14, 100]	161.08	[24, 48]
187-rc-e	[**58**, 216]	2551.50	[37, **204**]	2097.96	[58, 204]
198-c-c	[**16**, –]	–	[8, **68**]	122.39	[16, 68]
200-r-e	[**60**, 308]	3550.81	[38, **228**]	2613.35	[60, 228]
Average	[40.03, –]	–	[27.67, 133.47]	1334.92	[40.17, 107.87]

[a] CPU Intel Core i7 12700F - 4 × 3.6 GHz, 8 × 4.9 GHz, 20 threads; RAM 32 GB; OR-Tools CP-SAT 9.6; 3600 s time limit.

Table 8
Experimental results on the PDSVRP-c. Large instances, 5 trucks.

Instance	$CP2$+(16)[a]		$CP3$+(16)[a]		Best
	[LB, UB]	Sec_{bst}	[LB, UB]	Sec_{bst}	bounds
50-r-e	[**47**, **100**]	227.16	[30, 112]	54.55	[47, 100]
53-r-e	[**50**, **92**]	645.51	[32, 112]	667.24	[50, 92]
66-rc-e	[**35**, **100**]	487.89	[24, 120]	482.57	[35, 100]
67-c-c	[**18**, **52**]	64.31	[8, **52**]	1437.96	[18, 52]
68-rc-c	[**28**, **44**]	1047.89	[23, 80]	1776.83	[28, 44]
76-c-c	[**12**, **24**]	67.04	[**12**, 40]	400.81	[12, 24]
82-c-e	[**15**, **64**]	17.43	[6, **64**]	25.43	[15, 64]
82-rc-c	[**32**, **68**]	592.96	[24, 112]	782.49	[32, 68]
88-c-e	[23, **72**]	218.44	[**32**, 108]	109.66	[32, 72]
91-r-c	[**38**, **88**]	3122.27	[28, 124]	3272.57	[38, 88]
99-rc-c	[**32**, **64**]	597.67	[20, 108]	2532.76	[32, 64]
101-rc	[**36**, 112]	532.65	[26, 144]	505.96	[36, 76]
103-rc-c	[**32**, **80**]	1419.11	[22, 120]	3400.80	[32, 80]
105-rc-e	[**33**, **112**]	1282.90	[21, 124]	410.79	[33, 112]
108-rc-e	[**39**, **120**]	957.98	[24, 136]	1566.66	[39, 120]
114-rc-c	[**30**, **64**]	733.92	[22, 96]	299.50	[30, 64]
121-rc-e	[**34**, 116]	1034.38	[24, 128]	3100.10	[34, 104]
126-rc-e	[**41**, **120**]	2562.32	[29, 148]	2626.65	[41, 120]
126-r-c	[**37**, **116**]	1485.59	[24, 144]	807.31	[37, 116]
144-rc-c	[**34**, **104**]	2325.39	[22, 136]	2332.08	[34, 104]
154-c-c	[**15**, **36**]	1719.34	[6, 68]	669.42	[15, 36]
165-r-c	[**47**, 220]	1614.09	[34, **212**]	3294.70	[47, 212]
167-r-e	[**49**, 204]	1884.33	[34, 204]	1667.48	[49, 196]

(continued on next page)

Table 8 (*continued*).

Instance	$CP2$+(16)[a]		$CP3$+(16)[a]		Best
	[LB, UB]	Sec_{bst}	[LB, UB]	Sec_{bst}	bounds
173-r-c	[**43**, –]	–	[32, 196]	2657.03	[43, 192]
173-rc-c	[**39**, **116**]	2955.97	[24, 164]	3203.29	[39, 116]
181-r-e	[**54**, **204**]	3349.71	[35, **204**]	2369.20	[54, 204]
185-c-c	[**20**, **48**]	1216.37	[12, 60]	2561.48	[20, 48]
187-rc-e	[**48**, **128**]	2645.47	[32, 192]	2310.65	[48, 128]
198-c-c	[**16**, **36**]	487.28	[8, 68]	118.12	[16, 36]
200-r-e	[**52**, 288]	2545.74	[32, **216**]	2152.81	[52, 216]
Average	[34.30, –]	–	[23.40, 126.40]	1586.56	[34.60, 101.60]

[a] CPU Intel Core i7 12700F - 4 × 3.6 GHz, 8 × 4.9 GHz, 20 threads; RAM 32 GB; OR-Tools CP-SAT 9.6; 3600 s time limit.

References

Dantzig, G., Fulkerson, R., Johnson, S., 1954. Solution of a large-scale traveling-salesman problem. J. Oper. Res. Soc. Amer. 2 (4), 393–410.

Dell'Amico, M., Montemanni, R., Novellani, S., 2020. Matheuristic algorithms for the parallel drone scheduling traveling salesman problem. Ann. Oper. Res. 289, 211–226.

Dell'Amico, M., Montemanni, R., Novellani, S., 2021. Algorithms based on branch and bound for the flying sidekick traveling salesman problem. Omega 104, 102493.

Dell'Amico, M., Montemanni, R., Novellani, S., 2022. Exact models for the flying sidekick traveling salesman problem. Omega 29 (3), 1360–1393.

Dinh, Q.T., Do, D.D., Hà, M.H., 2021. Ants can solve the parallel drone scheduling traveling salesman problem. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 14–21.

Forbes, 2022. Drone explosion: $5B investment in 2 years, 129 startups, 170 new craft. https://www.forbes.com/sites/johnkoetsier/2022/02/07/drone-innovation-check-up-5b-investment-129-companies-170-craft/ (Accessed: 05 February 2023).

Gurobi Optimization, L., 2023. Gurobi optimizer reference manual. URL https://www.gurobi.com (Accessed 01 July 2023).

IBM ILOG, 2023. User's manual for CPLEX. https://www.cplex.com/ (Accessed 01 July 2023).

Lei, D., Chen, X., 2022. An improved variable neighborhood search for parallel drone scheduling traveling salesman problem. Appl. Soft Comput. 127, 109416.

Liu, Z., Sengupta, R., Kurzhanskiy, A., 2017. A power consumption model for multi-rotor small unmanned aircraft systems. In: Proceedings of the IEEE International Conference on Unmanned Aircraft Systems (ICUAS). pp. 310–315.

Mbiadou Saleu, R.G., Deroussi, L., Feillet, D., Grangeon, N., Quilliot, A., 2018. An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem. Networks 72 (4), 459–474.

Mbiadou Saleu, R.G., Deroussi, D., Grangeon, N., Quilliot, A., 2022. The parallel drone scheduling problem with multiple drones and vehicles. European J. Oper. Res. 300, 571–589.

Montemanni, R., Dell'Amico, M., 2023a. Constraint programming models for the parallel drone scheduling vehicle routing problem. EURO J. Comput. Optimiz. 11, 100078.

Montemanni, R., Dell'Amico, M., 2023b. Solving the parallel drone scheduling traveling salesman problem via constraint programming. Algorithms 16 (1), 40.

Murray, C.C., Chu, A.G., 2015. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transp. Res. C 54, 86–109.

Nguyen, M.A., Dang, G.T.-H., Hà, M.H., Pham, M.-T., 2022. The min-cost parallel drone scheduling vehicle routing problem. European J. Oper. Res. 299, 910–930.

Nguyen, M.A., Hà, M.H., 2023. The parallel drone scheduling traveling salesman problem with collective drones. Transp. Sci. 4 (57), 866–888.

Nguyen, M.A., Luong, H.L., Hà, M.H., Ban, H.B., 2023. An efficient branch-and-cut algorithm for the parallel drone scheduling traveling salesman problem. 4OR 21, 609–637.

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.

Paczan, N.M., Elzinga, M.J., Hsieh, R., Nguyen, L.K., 2022. Collective unmanned aerial vehicle configurations. Patent US 11, 480, 958 B2.

Pasha, J., Elmi, Z., Purkayastha, S., Fathollahi-Fard, A.M., Ge, Y.-E., Lau, Y.-Y., Dulebenets, M.A., 2022. The drone scheduling problem: A systematic state-of-the-art review. IEEE Trans. Intell. Transp. Syst. 23 (9), 14224–14247.

Perron, L., Furnon, V., 2023. Google OR-Tools. https://developers.google.com/optimization/ (Accessed 03 March 2023).

Raj, R., Lee, D., Lee, S., Walteros, J., Murray, C., 2021. A branch-and-price approach for the parallel drone scheduling vehicle routing problem. SSRN Electron. J. 1–47.

Raj, R., Murray, C., 2020. The multiple flying sidekicks traveling salesman problem with variable drone speeds. Transport. Res. Part C 120, 102813.

Statista, 2022. E-commerce. https://www.statista.com/markets/413/e-commerce/ (Accessed: 01 July 2023).

Stuckey, P.J., 2010. Lazy clause generation: Combining the power of SAT and CP (and MIP?) solving. In: Proceedings of the International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming (CPAIOR). pp. 5–9.

Wolleswinkel, R., Lukic, V., Jap, W., Chan, R., Govers, J., Banerjee, S., 2018. An onslaught of new rivals in parcel and express. Travel, Transport and Logistics. Boston Consulting Group.

Wolpert, D., Macready, W., 1997. No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1, 67.

Zhang, J., Campbell, J.F., Sweeney, D.C.I., C., H.A., 2021. Energy consumption models for delivery drones: A comparison and assessment. Transport. Res. Part D 90.