INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 29 (2022) 1360–1393
DOI: 10.1111/itor.13030

Exact models for the flying sidekick traveling salesman problem

Mauro Dell'Amico, Roberto Montemanni and Stefano Novellani*

Dipartimento di Scienze e Metodi dell'Ingegneria (DISMI), Università di Modena e Reggio Emilia (UNIMORE), via Amendola 2, Reggio Emilia 42122, Italy
E-mail: mauro.dellamico@unimore.it [Dell'Amico]; roberto.montemanni@unimore.it [Montemanni]; stefano.novellani@unimore.it [Novellani]

Received 6 February 2020; received in revised form 19 June 2021; accepted 23 June 2021

Abstract

This paper presents three enhanced formulations for the flying sidekick traveling salesman problem, where a truck and a drone cooperate to deliver parcels to customers minimizing the completion time. The drone can leave and must return to the truck after visiting one customer, performing flights not exceeding its battery endurance while the truck can serve other customers. The new formulations allow to decrease the number of "big-M" constraints with respect to literature models and improve previous results by solving to optimality several benchmark instances for which an optimal solution was previously unknown. This paper also shows how to modify the new models to include several variants of the problem from the literature.

Keywords: aerial drones; routing; parcel deliveries; mixed integer linear programming (formulations); branch and cut

1. Introduction

The use of aerial drones or unmanned aerial vehicles is gaining more and more relevance in several nonmilitary fields, from precision agriculture, to logistics operations, to catastrophic events management, etc. Their use in a large and diverse set of applications is due to the advantages that result from their flexibility, agility, and usability mainly due to their small size and the fact that no human is needed on board. Drone applications can be divided, roughly, into two: (a) the collection of data and information and (b) the movement of goods. This last case is the one that interests us. The boom of e-commerce and the promise of faster deliveries has provided a challenging task for e-commerce and express delivery companies. Among the first companies exploring the use of drones in parcel deliveries, one can mention Alibaba, Alphabet, Amazon, and JD.com. We address the reader to a recent survey on optimization approaches for drones in the civil sector by Otto et al. (2018).

*Corresponding author.

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies
This is an open access article under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

In this paper, we consider the *flying sidekick traveling salesman problem* (FSTSP), a problem in which parcels are delivered to customers either by a truck or (if possible) by a drone. The two vehicles' work coupled such that the drone can leave and must return to the truck after visiting one customer, performing flights not exceeding its battery endurance. The synchronization among the two vehicles is essential and the completion time at the end of the operations should be minimized. The FSTSP is a generalization of the *traveling salesman problem* (TSP) and the *vehicle routing problem* (VRP), and thus is an NP-hard problem.

The contributions of this work are as follows:

- We propose novel formulations with a set of variables that account for the drone presence on the truck (used to avoid infeasible drone flights). We also reduce timing variables (used for synchronization). The new formulations are consequently more compact and efficient than those previously appeared (for details, see Section 4).
- We show how to modify the proposed formulations to accommodate features typical of variants proposed in the related literature, therefore providing novel formulations also for those variants.
- We solve to optimality for the first time all the Murray and Chu (2015) benchmark instances with 10 customers. For 14 of them, optimality is proven here for the first time.
- We solve many instances of FSTSP with 20 customers to optimality. In the literature, only instances with up to 12 or 13 customers were solved by Yurek and Ozmutlu (2018) and Dell'Amico et al. (2019) and some with 20 customers were solved before by Roberti and Ruthmair (2020) and Schermer et al. (2020) with compact formulations (CFs).
- We solve to optimality all instances of the variants from the literature with 9 and 14 customers and many instances with 20 customers.

The paper is organized as follows. In Section 2, we discuss upon the most relevant related literature, while in Section 3 we formally describe the problem. The enhanced mathematical formulations and their implementations are described in Sections 4. The modification to the formulations to accommodate several variants of the problem are shown in Section 5. Extensive computational results are presented in Sections 6 and 7. Section 8 concludes the paper.

2. Related literature

The amount of literature that considers optimization problems related to drones, or trucks and drones, has been experiencing a boom in the last few years. A recent survey by Otto et al. (2018) reports more than 300 papers. Otto et al. (2018) classify the planning of combined operational drones and other vehicles in four categories: (i) vehicles supporting operations of drones; (ii) drones supporting operations of vehicles; (iii) drones and vehicles performing independent tasks; (iv) drones and vehicles as synchronized working units. The FSTSP belongs to the last category and in particular refers to a single drone and truck.

Murray and Chu (2015) were the first to define and study the FSTSP. In such a problem, the truck and the drone can cooperate to serve customers. The drone can be launched from the depot or from the truck when stationed at a customer node, performs a delivery to exactly one customer, and returns to the depot or to the truck (rendezvous) stationed at a different node. The truck and

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

the drone must be synchronized and thus wait for each other. The objective function is to minimize the completion time. In the FSTSP, customers can be serviced only once by the truck or by the drone; however, some customers can be visited only by the truck because their request cannot be fulfilled by the drone. In this form of the problem, the drones cannot return at the launching point and each drone flight is limited by a finite endurance. In their work, Murray and Chu propose a mixed integer linear programming (MILP) formulation that is not able to solve to optimality the 10-customer instances they have generated, in 30 minutes. They also propose three heuristic methods based on the well-known nearest neighbor, savings, and sweep algorithms, modified to include the drone flights in the solution. Another problem that shares the main characteristics with the FSTSP is the *TSP with drone* (TSP-D), presented by Agatz et al. (2018) for the first time. The differences between the two problems are: in the TSP-D the customers can be visited more than once by the truck if it is convenient for drone launching and return; the launching and rendezvous places of the same flight may coincide (the so-called loops can be performed); endurance is unlimited and launching and rendezvous times are considered negligible. The authors present an integer linear programming (ILP) model and propose route first-cluster second heuristics. They first build a TSP solution with Concorde, a well-known TSP solver (see, e.g., Applegate et al., 2006) and then partition the solution to accommodate drone flights with a heuristic and an exact procedures based on dynamic programming (DP). The heuristic algorithm improves iteratively the solutions by applying a set of local search (LS) procedures. They solve instances with up to 10 customers. Bouman et al. (2018) extend the work by Agatz et al. (2018) by solving the TSP-D with DP that consist of three phases. First, it enumerates the shortest paths that a truck can do, subsequently it combines the truck paths with the drone flights, eventually it combines the components of the solutions with truck and drone and those with only truck paths to provide the optimal solution. The last step of the approach is then extended to an A* algorithm. Instances with up to 15 customers could be solved to optimality in less than one hour. To shorten the computing times, the authors also propose a restriction on the number of nodes visited by the truck during a drone flight: this leads to heuristic quality solutions in shorter times.

Note that visiting customers more than once to collect the drone and allowing to launch and collect the drone in the same location cannot always be possible: the fact that the truck could return to and stop by customer locations that have already been visited or that will be visited subsequently may not be accepted as a viable option by the customers.

Ha et al. (2015) solve an FSTSP. They propose two heuristic algorithms: a route first-cluster second and a cluster first-route second. An MILP formulation is used to solve the cluster step. Two sets of instances have been used, with 10 and 100 customers. In Ha et al. (2017), the same authors solve a similar problem with a different objective function, made of four cost components that depend on the total distance traveled by the truck, the one traveled by the drone, and the waiting time of the truck and the drone. They present an MILP formulation based on Murray and Chu's one and two heuristics. The first heuristic starts with a TSP solution and includes the drone flights in the solution due to LS procedures. The second heuristic is a greedy randomized adaptive search procedure (GRASP) that splits a TSP solution in order to have drone flights, the solution is thus improved with LS procedures. The larger instances solved exactly count 10 customers, the heuristics solve instances with up to 100 customers. Ha et al. (2018) propose a hybrid GA improved with LS procedures to solve both minimum time and minimum cost version of the FSTSP. They solve instances with up to 100 customers. In a technical report, Liu (2018) solves an FSTSP in which the endurance

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

is expressed in kilometers and only one or two customers can be visited by the truck while the drone is in service. The author proposes a GA that is driven by two objective functions: the minimum time and the minimum consumed energy. The proposed instances reach up to 40 customers in size. In his thesis, Ponza (2016) tackles the FSTSP proposing a modified MILP formulation with respect to the Murray and Chu's (2015) one and solves it with a simulated annealing algorithm. Ponza's formulation does not allow the drone to wait on the ground at customer nodes to save battery, so the time spent in waiting for the truck is included in the computation of the used energy. Instances with up to 9 customers could be solved exactly in reasonable time, while the heuristic is tested on instances with up to 200 customers. de Freitas and Penna (2018) propose a randomized variable neighborhood descent for the FSTSP that starts from a TSP solved with Concorde. Some customers are then allocated to the drones and the obtained solution is improved by applying five neighborhoods in a variable neighborhood descent framework. They use TSPLIB-based (see, e.g., Reinelt, 1991) instances with up to 100 customers. In de Freitas and Penna (2020), the same authors propose a hybrid general variable neighborhood search algorithm for the FSTSP proposed by Murray and Chu (2015) and the TSP-D proposed by Agatz et al. (2018). In their algorithm, the authors first obtain the TSP solution due to the Concorde solver, the solution is then modified due to the greedy approach presented by Murray and Chu and improved by applying seven neighborhoods in a VNS scheme. They solve instances with up to 250 customers, proposing also a new set of instances based on the TSPLIB (Reinelt, 1991).

Poikonen et al. (2019) propose a branch and bound (B&B) for a TSP-D version that considers a maximum endurance and drone flights starting and ending in the same node (loops), but not multiple visits by the truck. The studied problem is basically an FSTSP with loops in which no times for launching and collecting the drone are considered. The dedicated B&B begins at the root node with a truck route starting and returning to the depot. The children nodes are made by inserting a new customer in different positions of the sequence. The computation of the lower bound at each branching node is made by considering the partitioning between drone and truck nodes of the given sequence. They also propose three heuristic algorithms, two derived from the B&B and one that is a divide-and-conquer heuristic. This last algorithm first solves the TSP, then divides the solution in several smaller groups, and solves the TSP-D for each group with the B&B. It eventually composes these subsolutions into a TSP-D subsolution. They optimally solve instances with up to 9 customers and heuristically solve instances with up to 200 customers. Yurek and Ozmutlu (2018) propose an iterative algorithm based on a decomposition approach for the FSTSP. In the first phase, they determine the truck route and the customers to be visited by the drone by complete enumeration. In the second phase, after fixing the truck route and the assignment of the first phase, an MILP is solved to determine a feasible solution for the problem. The heuristic algorithm, instead of enumerating all the routes in the first phase, takes the route provided by applying the nearest neighbor algorithm on a limited set of customers. They work on instances with up to 20 customers, being the exact methods able to solve instances with up to 12 customers.

Dell'Amico et al. (2019) solve two variants of the FSTSP, one where the drones can wait on the ground at customer nodes and the waiting time is not included in the drone flight, and the one in which the drone can wait only if hovering. They call these two versions of the problem "wait" and "no wait". They propose two MILP formulations that improve Murray and Chu's one, one where flights are represented with 3-indexed variables and one in which the flights are represented with 2-indexed variables. The authors consider an objective function that is equivalent to Murray

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

and Chu's one, but that provides higher lower bounds, they also propose a set of inequalities to be separated in a branch-and-cut (B&C) algorithm in order to be able to decrease the large number of variables and constraints needed to represent synchronization. They solve instances with up to 13 customers. The same authors propose a random restart local search matheuristc for the FSTSP "no-wait" version in Dell'Amico et al. (2021a) improving some best-known solutions for instances from the literature. In a more recent paper, the same authors propose a B&B and a B&B-based heuristic for the FSTSP "no-wait" version. The exact algorithm provides good results for small instances even with large endurance values and the heuristic improves on the related literature.

Roberti and Ruthmair (2020) propose exact solution approaches, MILP and branch and price (BP), to solve TSP-D in which no loops, no endurance, and only single visits are allowed. Then they also show how to include these variants into their method. The MILP could solve instances with up to 9 customers in the basic TSP-D form, while the BP could solve most of the instances with 29 customers. Schermer et al. (2020) propose MILP formulations and B&C algorithms to solve the TSP-D in which loops are allowed, endurance is limited, only single visits are allowed, and drones can wait on the ground without consuming battery. The authors also explain how to accommodate the other features in their methods but the feature that considers that nodes can be visited multiple times. In the first formulation that they propose, the authors use two 2-indexed binary arc variables for the sorties, one for the launch and one for the return of the drone flights, a separate binary variable for the loops, and two times variables: one for the first time a node is reached by a vehicle and one for the earliest possible departure. In their second formulation, they use only one binary 2-indexed arc variable to represent drone flights, which is coupled with a binary variable that defines if a node is the point where the drone returns to the truck. These formulations include "big-M" constraints to compute times, while the third formulation that they propose is based on their first formulation in which no "big-M" constraints are used. They include a binary variable that defines the position of a vertex in the sequence and three linear variables for the times, one for the waiting times of the truck, one for the waiting time of the drone, and one to account for the time in which both vehicles are traveling in parallel. The times are computed by including inequalities in a B&C fashion. In our work, instead, we propose three formulations, one in which sorties are represented as 3-indexed binary variables and two formulations with two sets of 2-indexed binary variables. We use one set of linear time variables and another to compute waiting times. Moreover, we use a binary variable to represent the position of the drone with respect to the truck in two of three formulations, while in the third formulation we separate the infeasible drone flights in a B&C fashion. Their best methods could solve all the problems with 9 and 14 customers but only few with 20. Note that the last two papers and this one have been developed independently and simultaneously.

The papers that we analyzed in this section solve problems that use one truck and one drone. Most of the times these problems are called FSTSP and TSP-D, even if not every time the same name represents the same problem. To have a better picture of the different characteristics of the treated problems and to identify the solving methods that have been used we head the reader to Table 1. For each paper we report, in the column *Formulation*, if it proposes one or more (M)ILP formulations for the problem and, in such a case, we summarize their main characteristics. The column *Loops* is used to identify if the treated problem allows the drone to return to the same node from where it has been launched. In the column *Multivisits*, letter *y* shows if the truck can visit the same node multiple times. In the column *Truck only*, letter *y* shows if in the treated problem a subset of vertices can be visited only by the truck. In column E, we show if a certain problem considers a

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Table 1
Classification of single truck and single drone problems

Paper	Formulation	Loops[a]	Multivisits	Truck only	Wait[b]	E[c]	Obj. funct.[d]	Solving methods	Instance size[e]
Murray and Chu (2015)	MILP 2-index (truck arcs), 3-index (sorties)			y	y	y	T	MILP, heuristic (nearest neighbor, savings, sweep + savings for UAV routes definition)	E(10: could not be solved to optimality in 30 minutes), H(10)
Agatz et al. (2018)	ILP set covering	y	y	y			T	Heuristic (route first cluster second)	E(10) H(100)
Bouman et al. (2018)	None	y	n.a.	y			T	DP (and limited DP that provides heuristic solutions)	E(15 in 1 h TL) H(20)
Ha et al. (2015)	MILP for clustering, set packing			n.a.	n.a.	y	T/D	Heuristic (route first cluster second, cluster first route second)	H(100)
Ha et al. (2017)	MILP 2-index (truck arcs), 3-index (sorties)			y	y	y	D	MILP and Heuristic (TSP + LS, and GRASP + LS)	E(10), H(100)
Ha et al. (2018)	None			y		y	T and D	Hybrid GA + LS	H(100)
Liu (2018)	None			n.a.	n.a.	y	T and En	GA	H(40)
Ponza (2016)	MILP 2-index (truck arcs), 3-index (sorties)			y		y	T	MILP, SA	E(9 in 1h TL), H(200)

Continued

International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies
© 2021 The Authors.

Table 1
(Continued)

Paper	Formulation	Loops[a]	Multivisits	Truck only	Wait[b]	E[c]	Obj. funct.[d]	Solving methods	Instance size[e]
de Freitas and Penna (2018)	None			y		y	T	TSP solved with Concorde + LS to serve some customers with the drone + RVND + LS.	H(100)
de Freitas and Penna (2020)	None	y/n	y/n	y	n.a.	y/n	T	HGVNS	H(250)
Poikonen et al. (2019)	None	y		y	n.a.	y	T	B&B and B&B-based heuristics, and divide-and-conquer heuristic	E(10), H(200)
Yurek and Ozmutlu (2018)	MILP to determine drone sorties			y	y	y	T	Decomposition-based iterative algorithm (exact and heuristic)	E(12), H(20)
Dell'Amico et al. (2019)	MILP: 2-index (truck arcs), 3-index (sorties); 2-index (truck arcs), 2-index (sorties)			y	y/n	y	T	MILP, B&C	E(13)
Dell'Amico et al. (2021a)	MILP: 2-index (truck arcs), 2-index (sorties)			y		y	T	RRLS	H(152)

Continued

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Table 1
(Continued)

Paper	Formulation	Loops[a]	Multivisits	Truck only	Wait[b]	E[c]	Obj. funct.[d]	Solving methods	Instance size[e]
Roberti and Ruthmair (2020)	MILP: 2-index (truck arcs), 2-index (sorties); ILP set partitioning	y/n		y/n	y/n	y/n	T	MILP, branch and price	E(MILP 9, BP 29)
Schermer et al. (2020)	MILP: 2-index (truck arcs), 2-index (sorties)	y/n		y/n	y/n	y	T	MILP, B&C	E(20)
Dell'Amico et al. (2021b)	None			y		y	T	B&B and B&B heuristic	E(19), H(200)
This paper	Enhanced MILP: 2-index (truck arcs), 3-index (sorties); 2-index (truck arcs), 2-index (sorties)	y/n		y/n	y/n	y/n	T	MILP, B&C	E(20)

HGVNS, hybrid general variable neighborhood search; RRLS, random restart local search; RVND, randomized variable neighborhood descent; SA, simulated annealing.
[a] y: the drone is allowed to return to the same node from where it has been launched.
[b] y: drone can land and wait on the ground to save energy.
[c] y: finite endurance is considered.
[d] *T*: minimize the last completion time; *D*: minimize the overall distance/time (both truck and drone) depending cost; *En*: minimize the overall used energy.
[e] *E*: indicates the size of the largest instance solved within one hour by the proposed exact; *H*: indicates the largest used instances for the proposed heuristic.

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

finite endurance for the drones. Under the *Wait* column, a letter *y* indicates if the problem allows the drone to wait for the truck on the ground, instead that in a flying mode. This is linked to the maximum endurance of the drone: if the drone can wait only in flying mode, the battery represents a more binding constraint. Note that wait has meaning only when the endurance is not unlimited. The column *Obj. funct.* shows the used objective function, where T indicates that the problem wants to minimize the completion time of all the operations, D if the objective function's aim is to minimize the overall distance/time (of both truck and drone) depending cost, and *En* the overall used energy. Note that the use of both letters *y* and *n* means that the paper considers two versions, with and without the single characteristic. We use *n.a.* when the information is not available or when the paper is not clear about one characteristic. In the last two columns, we report a summary of the solving methods and the size of the used instances. The number followed by letter E indicates the size of the largest instance solved within one hour by the proposed exact algorithms, the number followed by letter H indicates the largest used instances for the heuristic algorithms.

Other problems that consider a single drone and a single truck are the following. Jeong et al. (2019) solve the *FSTSP with energy consumption and no fly zones*. This problem accounts for the parcel weight on drone energy consumption and the drone cannot fly over some restricted flying areas. They propose an MILP formulation based on Murray and Chu's one and an evolutionary-based heuristic. Marinelli et al. (2017) study the *en route TSP-D*, in which the drone can start and return on the arcs traveled by the truck. The launch and return points are decided to minimize the waiting times. They solve the proposed problem with a GRASP based on the method by Ha et al. (2017) where the initial solution is obtained with the Lin–Kernighan algorithm (see, e.g., Lin and Kernighan, 1973).

3. Problem description

The FSTSP, first defined by Murray and Chu (2015), is the problem of serving a set of customers $C = \{1, \ldots, c\}$ with either a truck or a drone. The truck starts from the depot 0 and returns to the final depot $c + 1$, and is equipped with a flying drone that can be used to serve one customer at a time, in parallel to the truck. A drone service is called *sortie*, defined by a launching node, a served customer, and a rendezvous node. All customers of C can be served by the truck, but only a subset $C' \subseteq C$ can be served by the drone with a sortie. The problem is built on digraph $G = (N, A)$, where the set $N = \{0, 1, \ldots, c + 1\}$ represents all the nodes, while we define $N_0 = \{0, 1, \ldots, c\}$ and $N_+ = \{1, \ldots, c + 1\}$. Let $n = |N|$ in the following. Let A be the set of all the arcs (i, j), $i \in N_0$, $j \in N_+$, $i \neq j$. Each arc (i, j) is associated with two nonnegative traveling times, τ_{ij}^T and τ_{ij}^D, which represent the time for traveling that arc by the truck and by the drone, respectively. The travel time matrices of the drone and the truck are normally different. Nodes 0 and $c + 1$ represent the same physical point, the depot, and the traveling time between them is set to 0. Service times at customers for both drone and truck are included in the travel times, while the time for preparing the drone at launch is given by σ^L and the rendezvous time is given by σ^R. No launch time is considered when the sortie starts from the depot. The drone has a battery limit (endurance) of E time units, which constraints its use for each sortie. Rendezvous time σ^R contributes to the endurance computation while σ^L does not, since the drone lies on the truck when it is prepared for the launch.

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

A sortie is formally defined by a triplet $\langle i, j, k\rangle$, ($i \neq j, k$, and $j \neq k$), where $i \in N_0$ is the launching node, $j \in C'$ the customer to serve, and $k \in N_+$ the rendezvous node. Let F be the set of all sorties that can be performed within the endurance time E ($\tau^D_{ij} + \tau^D_{jk} + \sigma^R \leq E$) that is the maximum time within each sortie must be completed. After each sortie, the battery is swapped. Note that a sortie $\langle i, j, k\rangle \in F$ can still become infeasible along the route in case the truck that is traveling from i to k exceeds the endurance E, because the drone must wait for the truck while hovering and thus consuming its battery. Two sorties $\langle i, j, k\rangle$ and $\langle i', j', k'\rangle$ are called *crossing* if i' is visited by the truck after i but before k, thus inducing an infeasible solution.

The drone can be launched from the truck only when the truck is stationary at a customer or at the depot; the drone cannot leave the depot before the truck starts its route. The truck can keep serving customers while the drone is performing a sortie. A synchronization is required: the vehicle (drone or truck) that arrives first at a rendezvous point has to wait for the other. The objective of the optimization is to minimize the completion time, which is the moment when the last vehicle arrives at the depot.

4. Formulations

In the following, we propose three new formulations for the FSTSP. These formulations represent an enhancement with respect to those proposed by Dell'Amico et al. (2019), which already improved upon the MILP formulation proposed by Murray and Chu (2015). In the first formulation, sorties are represented as 3-indexed variables, in the last two as 2-indexed variables. The novelties of these formulations are as follows: (a) we use only one set of time variables to synchronize truck and drone instead of two sets, one for the truck and one for the drone; (b) we include a binary variable that states the position of the drone with respect to the truck. This helps in avoiding crossing sorties and thus infeasible solutions.

The new variables allow to write more compact formulations, decreasing the number of constraints from $2n^3 + O(n^2)$ to $n^3 + O(n^2)$. In Table 2, we compare CFs of the literature with the one proposed in this work. We show the number of integer, binary, and linear variables and the number of constraints. Note that routing constraints are not reported in the table because they do not differ in number among the several formulations. Some formulations include exponentially many constraints that are separated in a B&C fashion. The "big-M" constraints are highlighted in bold. We also show the number of inequalities that can be added to the new formulations but are not needed to obtain the optimal solution (for instance, our final implementation of 2IF-BC [where 2IF-BC is 2-indexed formulation B&C] only uses the time-constraint inequalities).

4.1. A 3-indexed formulation

The first formulation we present, 3IF, is built on the formulation DMN proposed by Dell'Amico et al. (2019), where the truck route is represented using the variable $x_{ij} = 1$ if the node $j \in N_+$ is visited immediately after node $i \in N_0$, $j \neq i$, and 0 otherwise. The drone sorties are represented by a 3-indexed variable y_{ijk}, $\langle i, j, k\rangle \in F$, that equals 1 if the sortie is performed, 0 otherwise. Nonnegative variables w_i represent the time that the truck waits for the drone at node $i \in N$. With respect to DMN, we include nonnegative variables t_i, $i \in N$, which are used to represent the time

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Table 2
The number of variables and constraints of different formulations for the FSTSP

	MC	DMN	DMN2	3IF	2IF	2IF-BC
Variables						
Binary variables	$n^3 + 2n^2$	$n^3 + n^2$	$3n^2$	$n^3 + n^2 + n$	$3n^2 + n$	$3n^2$
Integer variables	n					
Linear variables	$2n$	$3n$	$3n$	$2n$	$2n$	$2n$
Constraints						
Sorties-route links	$n^3 + 2n$	$n^3 + 2n$	$3n$		n	$3n$
Time constraints	$3n^2 + 4n$	$4n^2 + 4n$	$4n^2 + 4n$	$3n^2$ (+$2n$ ineq.)	$4n^2$ (+$2n$ ineq.)	$4n^2$ (+$2n$ inequalities)
No crossing sorties	$n^3 + 5n^2 + n$	Path-based constraints separated in a B&C fashion	Path-based constraints separated in a B&C fashion	$n^2 + 2n$ (+$2n$ inequalities)	$3n^2 + 2n$ (+$2n^2 + n$ inequalities)	$2n^2$+ Path-based constraints separated in a B&C fashion (+$2n^2$ inequalities)
Endurance constraints	n^3	n^3	n^3	n^2	n^3 (+n inequalities)	n^3 (+n inequalities)

Note. Routing constraints are not reported. The "big-M" constraints are reported in bold. MC is the formulations by Murray and Chu (2015), DMN and DMN2 are those proposed by Dell'Amico et al. (2019), while 3-indexed formulation 3IF, 2IF, and 2IF-BC are the formulations presented in this paper.

synchronization and identify one of the novelties of this formulation (that halves the time variables with respect to DMN). We also include in 3IF a new variable z_i that equals 1 if the drone is on the truck at node $i \in N$ or if the drone returns to the truck at node $i \in N$, 0 otherwise, which is used to avoid crossing sorties and allows to diminish constraints.

In the following sections, we describe the formulation 3IF step by step.

4.1.1. Objective function

The objective function (1) aims at minimizing the arrival of truck and drone at the final depot. We clarify that the drone can arrive at the depot after the truck and this is described by the waiting time variable w_{c+1}. As it has been shown by Dell'Amico et al. (2019), the completion time can be decomposed into the following components: the truck route traveling time, the time needed for launching and collecting a drone for each sortie, and the time the truck waits for the drone:

$$\min \sum_{(i,j)\in A} \tau_{ij}^T x_{ij} + \sigma^R \sum_{\langle 0,j,k\rangle \in F} y_{0jk} + (\sigma^L + \sigma^R) \sum_{\substack{\langle i,j,k\rangle \in F \\ i \neq 0}} y_{ijk} + \sum_{i\in N_+} w_i. \tag{1}$$

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Note that the value of this objective function is equivalent to the minimization of the maximum completion time (objective function used by Murray and Chu, 2015); however, this decomposition may lead to a substantial improvement in the lower bounds. Dell'Amico et al. (2019) have shown that by changing only the objective function, the average lower bound gap at the root node can decrease from an 81.5%, with the classical Murray and Chu's objective function, to a 22.5% gap with the decomposed one.

4.1.2. Customer covering

Constraints (2) enforce one of the two vehicles to serve each customer exactly once:

$$\sum_{i|(i,j)\in A} x_{ij} + \sum_{i,k|\langle i,j,k\rangle\in F} y_{ijk} = 1 \quad j \in C. \tag{2}$$

4.1.3. Truck routing constraints

In (3), we impose that the truck starts and finishes its journey at the depots:

$$\sum_{j\in N_+} x_{0j} = \sum_{i\in N_0} x_{i,c+1} = 1. \tag{3}$$

We also need to impose the truck flow conservation constraints:

$$\sum_{i|(i,j)\in A} x_{ij} = \sum_{i|(j,i)\in A} x_{ji} \quad j \in C. \tag{4}$$

4.1.4. Timing constraints

In order to guarantee the timing constraints, we impose constraints (5) to ensure that, if arc (i, j) is traveled by the truck, then the time in j is at least the time in i plus the time needed for traveling the arc. In (6), we assure that if there is a sortie $\langle i, k, j\rangle$ then the time t_j should be at least t_i plus the time for performing the sortie. If the drone arrives at the rendezvous point after the truck, then the truck must wait stationary at that node for at least an amount of time that is the difference between the arrival time of the drone and the arrival time of the truck in that node, that is imposed by constraint (7). Note that if the drone arrives before the truck, it waits while flying, this waiting time is absorbed by time t_j of constraint (5), which is included in the truck route in the objective function:

$$t_j \geq t_i + \tau^T_{ij} - M(1 - x_{ij}) \quad (i, j) \in A \tag{5}$$

$$t_j \geq t_i + \sum_{k|\langle i,k,j\rangle\in F} (M + \tau^D_{ik} + \tau^D_{kj}) y_{ikj} - M \quad (i, j) \in A \tag{6}$$

$$w_j \geq t_j - t_i - \tau^T_{ij} - M(1 - x_{ij}) \quad (i, j) \in A. \tag{7}$$

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Note that the launching and rendezvous service times, which are explicitly considered in the objective function, are not considered in those constraints because it is not necessary to include them in our variables t. Indeed, launching and rendezvous service times should be included in both truck and drone timing constraints (5) and (6), however they have no effect in the computation of the waiting times (7) since they appear as a constant in both truck and drone times. For instance, let us consider a solution in which the truck travels the arc $(i, j) \in A$ and the drone operates the sortie $\langle i, k, j\rangle \in F$. In such a case t_j should be greater than or equal to $t_i + \sigma^L + \tau_{ij}^T + \sigma^R$ and also $t_i + \sigma^L + \tau_{ik}^D + \tau_{kj}^D + \sigma^R$ (note that σ^L is not included in the drone flight, but the drone must wait σ^L to start operating). To compute the waiting time, one must compute the difference between those two values: the two terms σ^L and σ^R appear in both terms and can be omitted. In constraint (7), we include this difference by considering two cases: if the drone time is less than the truck time then waiting time is included in the truck travel time added in the objective function, if the drone flight takes longer than the truck travel time then we need to consider this difference in the variable w_j that is included in the objective function. The same rationale can be applied to cases in which paths made of more than one arc link the launch and the rendezvous of sorties. To conclude the reasoning, one should then remember to add the rendezvous time when imposing the maximum endurance to the drone flight as we do in constraint (8), because this is not included in variables t.

Constraints (5) and (6) also have the side effect of avoiding backward sorties, that is, sorties $\langle i, k, j\rangle \in F$ such that node j is visited by the truck before node i. These constraints also have the side effect of avoiding subtours, if $\tau_{ij}^T \neq 0$, $(i, j) \in A$; otherwise one must take care of avoiding subtours among coincident nodes.

In contrast with the other models in the literature, t variables do not represent the exact time of visiting one node, but they allow the model to respect the truck–drone timing, to compute the waiting times, and allow us to obtain a more compact formulation.

4.1.5. Drone battery endurance constraints

Through constraints (8) we assure that if a sortie $\langle i, k, j\rangle$ is performed, the elapsed time from the launch to the rendezvous respects the drone endurance E. The rendezvous service time σ^R is included in the drone time:

$$t_j - t_i + \sigma^R - M\left(1 - \sum_{k|\langle i,k,j\rangle \in F} y_{ikj}\right) \leq E \quad (i, j) \in A. \tag{8}$$

Note that we include constraints (8) to impose that the drone battery endurance does not exceed during a sortie because the drone must wait for the truck in flying mode. We do not allow the drone to wait for the truck on the ground for several practical reasons: it is not always possible to have a place available where the drone can land, for example, the customers could not allow the drone to wait in their private property, and even so, the drone could incur safety and security problems while left unattended. In case the drone was allowed to wait on the ground, this constraint appears superfluous as the infeasible sorties can be avoided in preprocessing.

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

4.1.6. x–z and y–z linking constraints

Constraints (9) state that a drone can be on the truck at node i if the truck enters node i or, in turn, using (4), that the truck exits node i. Constraints (10) state that a sortie can start in i if variable z_i equals 1. Those constraints, in addition to constraints (11), are also used to avoid crossing sorties. Constraints (11) regulate z variables, modeling the presence of the drone over the truck route. Consider the first node i on the truck route from which starts a sortie $\langle i, k, l\rangle \in F$ and arc $(i, j) \in A$ such that $x_{ij} = 1$. No sortie can arrive in node j (apart from the one starting in node i) since (5) and (6) forbid backward sorties. Hence, given $z_i = 1$ (because of (10)), (11) imposes $z_j = 0$ if the sortie returns in $l \neq j$. If, instead, the sortie that starts in i returns in j, z_j can take the value 1, which means that a new sortie can start in j. Going back to the general case, where the sortie does not return in j, the z variables along the path after the sortie launch in i and before the rendezvous in l remain at value 0, imposing that no other sortie can start until the truck reaches node l, where the drone returns. After that, the z variables can take value 1, which means that a new sortie can start:

$$z_i \leq \sum_{j|(j,i)\in A} x_{ji} \quad i \in N_+ \tag{9}$$

$$\sum_{j,k|\langle i,j,k\rangle\in F} y_{ijk} \leq z_i \quad i \in N_0 \tag{10}$$

$$z_j \leq z_i - x_{ij} + \sum_{l,k|\langle l,k,j\rangle\in F} y_{lkj} - \sum_{k,l|\langle i,k,l\rangle\in F} y_{ikl} + 1 \quad (i, j) \in A. \tag{11}$$

4.1.7. Variable bounds

$$t_0 = 0 \tag{12}$$

$$t_i, w_i \in \mathbb{R}^+ \quad i \in N_+ \tag{13}$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in A \tag{15}$$

$$y_{ijk} \in \{0, 1\} \quad \langle i, j, k\rangle \in F. \tag{16}$$

In contrast to Murray and Chu and DMN, 3IF does not need $n^3 + 2n$ constraints to link sorties and the truck route because this is guaranteed by $n^2 + 2n$ crossing sortie elimination constraints (9)–(11). Moreover, the Murray and Chu formulation needs n^3 "big-M" constraints, and $5n^2 + n$ other constraints to avoid crossing sorties, while DMN uses an exponentially large number of path-based constraints, separated in a B&C fashion. In 3IF formulation, only the $n^2 + 2n$ constraints (9)–(11) are necessary to avoid crossing sorties and link truck routes and sorties. Concerning the time constraints (that are "big-M" constraints) Murray and Chu use $3n^2 + 4n$ constraints, DMN $4n^2 + 4n$ constraints, while 3IF uses only $3n^2$ constraints. Finally, we decreased the endurance "big-M" constraints from n^3 of Murray and Chu and DMN to n^2.

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

4.1.8. Inequalities

We can strengthen 3IF by adding the following constraints: no crossing sorties inequalities such as (17)–(19) that link more tightly variables x and y to variable z, and inequalities (20) and (21) that limit variable w, whose value is set to 0 for those customers served by the drone or for those nodes in which no drone returns to the truck:

$$z_0 \leq \sum_{j|(0,j)\in A} x_{0j} \tag{17}$$

$$\sum_{k,j|\langle k,j,i\rangle\in F} y_{kji} \leq z_i \quad i \in N_+ \tag{18}$$

$$\sum_{j,i|\langle j,i,k\rangle\in F} y_{jik} \leq 1 - z_i \quad i \in C' \tag{19}$$

$$w_j \leq E \sum_{i,k|\langle i,k,j\rangle\in F} y_{ikj} \quad j \in N_+ \tag{20}$$

$$w_j \leq E(1 - \sum_{i,k|\langle i,j,k\rangle\in F} y_{ijk}) \quad j \in C'. \tag{21}$$

4.2. A 2-indexed formulation

The 2IF is based on the formulation DMN2 presented in Dell'Amico et al. (2019). Similar to the explanation given for the previous formulation, the main difference with DMN2 is that 2IF uses only one set of time variables to represent synchronization, halving the number of time variables with respect to DMN2 and diminishing the number of "big-M" constraints required. The other main difference is the use of a binary variable z representing the presence or absence of the drone on the truck along the route in the same manner as in formulation 3IF. We model the truck route as in the 3IF; however, in this formulation, we model the sorties with a pair of 2-indexed binary variables as in DMN2: one for the launch and one for the rendezvous. Variable $\vec{g}_{ij}$ takes value 1 if the drone is launched in $i \in N_0$ and serves the customer $j \in C'$, and $\overleftarrow{g}_{jk}$ is 1 if the drone returns to node $k \in N_+$ after visiting customer $j \in C'$. To reduce the variables, we preliminary fix to zero all those corresponding to arcs with flying time exceeding the battery limit, that is, we set $\vec{g}_{ij} = 0$ for all $(i, j) \in A : \tau^D_{ij} > E$ and $\overleftarrow{g}_{jk} = 0$ for all $(j, k) \in A : \tau^D_{jk} + \sigma_R > E$. We also fix to zero variables that do not allow to complete a feasible drone fly: $\vec{g}_{ij} = 0, (i, j) \in A, j \notin C'$; $\overleftarrow{g}_{jk} = 0, (j, k) \in A, j \notin C'$; $\vec{g}_{i,c+1} = 0\ i \in N$; and $\overleftarrow{g}_{j0} = 0\ j \in N$. We clarify that this type of preprocessing is implicit for formulation 3IF by simply not including infeasible sorties in the set of feasible sorties F. Variables x, t, w, z are similar to the variables used for the previous formulation.

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

In formulation 2IF, truck routing constraints are similar to the constraints for 3IF ((3) and (4)). We also import from 3IF the timing constraints (5) and (7), constraints (9), and the variable bounds (12)–(15). The remaining components are as follows.

4.2.1. Objective function

The objective function (22) is the adjustment of (1) to the new sets of variables:

$$\min \sum_{(i,j)\in A} \tau_{ij}^T x_{ij} + \sigma^L \sum_{\substack{(i,\,j)\,\in\, A \\ i \neq 0}} \vec{g}_{ij} + \sigma^R \sum_{(j,k)\in A} \overleftarrow{g}_{jk} + \sum_{i\in N_+} w_i. \tag{22}$$

4.2.2. Customer covering

Constraints (23) impose that all customers must be served either by the truck or by the drone:

$$\sum_{i|(i,j)\in A} x_{ij} + \sum_{i|(i,j)\in A} \vec{g}_{ij} = 1 \quad j \in C. \tag{23}$$

4.2.3. Timing constraints

In addition to constraints (5) and (7), we impose constraints (24) and (25) to update the times of the two components of a sortie (launch and rendezvous) separately:

$$t_j \geq t_i + \tau_{ij}^D - M(1 - \vec{g}_{ij}) \quad (i,\, j) \in A \tag{24}$$

$$t_k \geq t_j + \tau_{jk}^D - M(1 - \overleftarrow{g}_{jk}) \quad (j,\, k) \in A. \tag{25}$$

4.2.4. Drone battery endurance constraint

If a sortie is performed then its total time should respect the battery endurance, as in constraints (26):

$$t_k - t_i + \sigma^R - M(2 - \vec{g}_{ij} - \overleftarrow{g}_{jk}) \leq E \quad i \in N_0,\, j \in C',\, k \in N_+,\, \langle i,\, j,\, k\rangle \in F. \tag{26}$$

4.2.5. Sortie congruence

Constraints (27) impose that if a drone sortie enters in customer $j \in C'$ then a drone sortie must exit the same node and return to the truck. Note that the number of sorties entering in each node is limited to one by constraint (23):

$$\sum_{i|(i,j)\in A} \vec{g}_{ij} = \sum_{k|(j,k)\in A} \overleftarrow{g}_{jk} \quad j \in C'. \tag{27}$$

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

4.2.6. x–z and g–z linking constraints

Similar to the previous formulation and in addition to constraints (9), we use (28) to impose that a sortie can start in node i only if $z_i = 1$ and (29) to avoid crossing sorties:

$$\sum_{j \in C'} \vec{g}_{ij} \leq z_i \quad i \in N_0 \tag{28}$$

$$z_j \leq z_i - x_{ij} + \sum_{k \in C'} (\overleftarrow{g}_{kj} - \vec{g}_{ik}) + 1 \quad (i, j) \in A. \tag{29}$$

4.2.7. Variable bounds

In addition to (12)–(15), we introduce the constraints on variables g:

$$\vec{g}_{ij} + \overleftarrow{g}_{ij} \leq 1 \quad (i, j) \in A \tag{30}$$

$$\vec{g}_{ij} + \overleftarrow{g}_{ji} \leq 1 \quad (i, j) \in A \tag{31}$$

$$\vec{g}_{ij}, \overleftarrow{g}_{ij} \in \{0, 1\} \quad (i, j) \in A \tag{32}$$

Note that we decrease the number of linking constraints from $n^3 + 2n$ and $3n$ constraints of Murray and Chu and DMN2, respectively, to n. Some of the linking constraints are not needed anymore because they are guaranteed by the crossing sorties avoiding constraints. The time constraints (that are "big-M" constraints) change from $3n^2 + 4n$ of Murray and Chu, and decreased from $4n^2 + 4n$ of DMN2, to the $4n^2$ constraints of 2IF. The constraints that avoid the crossing sorties diminish from $n^3 + 5n^2$ "big-M" constraints plus the other n non-"big-M" constraints of Murray and Chu to the $n^2 + n$ non-"big-M" constraints of 2IF. DMN2, instead, avoids crossing sorties with exponentially many path-based constraints separated in a B&C fashion.

4.2.8. Inequalities

The following inequalities (33) are a relaxed version of constraint (26), they avoid sorties longer than the maximum endurance. Notwithstanding this version of the constraint does not take into account the drone waiting time in case it arrives to the rendezvous point before the truck, it has a strong impact on the solution time of the formulation, since it is not affected by large constants (M):

$$\sum_{i \in N_0} \tau^D_{ij} \vec{g}_{ij} + \sum_{k \in N_+} \tau^D_{jk} \overleftarrow{g}_{jk} + \sigma^R \leq E \quad j \in C'. \tag{33}$$

These constraints are used for the experiments reported in Section 6 to strengthen (26).

Other inequalities that we can include in formulation 2IF, in addition to (17), are as follows: no crossing sortie inequalities (34) and (35) that tighten the link between variables g and z, and

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

inequalities (36) and (37) that avoid infeasible combinations of variables$g$. Inequalities (38)–(39) limit the waiting variables and impose them to be zero when not needed:

$$\sum_{j \in C'} \overleftarrow{g}_{j,c+1} \leq z_{c+1} \tag{34}$$

$$\sum_{j \in N_0} \vec{g}_{ji} \leq 1 - z_i \quad i \in C' \tag{35}$$

$$\vec{g}_{ij} + \vec{g}_{ji} \leq 1 \quad (i, j) \in A \tag{36}$$

$$\overleftarrow{g}_{ij} + \overleftarrow{g}_{ji} \leq 1 \quad (i, j) \in A \tag{37}$$

$$w_j \leq E \sum_{k \in C'} \overleftarrow{g}_{kj} \quad j \in N_+ \tag{38}$$

$$w_j \leq E \left(1 - \sum_{i \in N_0} \vec{g}_{ij} \right) \quad j \in C'. \tag{39}$$

4.3. A modified 2-indexed formulation

In this section, we present a mathematical formulation built upon the formulation 2IF with the difference that variables z are not necessary anymore to avoid infeasible sorties. For doing so, we use the crossing sorties elimination constraints proposed by Dell'Amico et al. (2019). These constraints are exponentially many and we separate them in a B&C (BC) fashion. We refer to this formulation as 2IF-BC. Note that 2IF-BC uses less constraints than the formulation proposed by Murray and Chu and less "big-M" constraints than DMN2. With respect to 2IF, 2IF-BC has the same number of "big-M" constraints but needs more constraints to guarantee the feasibility of the solution.

The set of variables used is the same as in 2IF, with the exclusion of variables z. The following constraints are added to (3)–(5), (7), (12), (13), (15), (22)–(27), (30)–(32). Moreover inequalities (33), (36)–(39) can also be included to strengthen 2IF-BC.

4.3.1. x–g Coupling constraints

Since this formulation does not use z variables, we need to impose that a sortie can start and end in one node only if the truck is there, respectively:

$$\sum_{j|(i,j) \in A} \vec{g}_{ij} \leq \sum_{h|(i,h) \in A} x_{ih} \quad i \in N_0 \tag{40}$$

$$\sum_{i|(i,j) \in A} \overleftarrow{g}_{ij} \leq \sum_{h|(h,j) \in A} x_{hj} \quad j \in N_+. \tag{41}$$

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

4.3.2. Crossing sorties elimination constraints

To avoid crossing sorties we need to impose the inequalities (42), which we report in its tournament version hereafter (for details, see Dell'Amico et al., 2019). Let $i \in N_0, l \in C$ be the starting and ending vertices of the truck path P from i to l and $P = \{v(1), v(2), \ldots, v(q)\}$ with $v(1) = i, v(q) = l$. Along the path P, we assume that two sortie launches defined by $\vec{g}_{ij} > 0$ and $\vec{g}_{lm} > 0$ occur and that there is no node $k \in P \setminus \{i, l\}$ such that $\overleftarrow{g}_{jk} > 0$. In this case, the second sortie starts before the first sortie is terminated and the following "tournament" crossing sorties elimination holds:

$$\sum_{r=1}^{|P|-1} \sum_{s=r+1}^{|P|} x_{v(r)v(s)} + \sum_{\substack{(i,j) \in A, \\ j \notin P}} \vec{g}_{ij} + \sum_{\substack{(l,m) \in A, \\ m \notin P}} \vec{g}_{lm} \leq |P| \quad P \in \mathcal{P}, \tag{42}$$

where $\mathcal{P}$ defines the set of all the paths with the described characteristics.

5. Variants

In this section, we explain how to modify our formulations to include several variants that could be found in the related literature that considers the following features: loops, unlimited drone endurance, and the case in which the drone is allowed to wait on the ground.

5.1. Loops

We recall that we refer to *loops* to those sorties whose launch and rendezvous points coincide. Loops are infeasible sorties in the FSTSP but they are a feasible feature for several related problems, and thus we show how our formulations can include them. Let us define a set F' that includes set F of sorties feasible for the FSTSP and the feasible loops $\langle i, j, i \rangle$, $i \in N_+$, $j \in C'$ such that $\tau^D_{ij} + \tau^D_{ji} + \sigma^R \leq E$. We allow loops to be performed from the depot, but only once and from the final depot; however, this can be easily avoided by modifying set F' to avoid such loops.

5.1.1. 3IF with loops

To accommodate loops in formulation 3IF, we must recall that the time variables we used do not represent the visit time in each node, but they are used mainly to allow us to compute the waiting times due to the synchronization, and thus we do not need to change the timing constraints in the formulation with loops. However, we need to change the objective function (1) to include the new time spent in a loop as in (43); indeed, no return to the same node is allowed to the truck, and thus the loops $\langle i, j, i \rangle$ must be performed when the truck is stopped at node i. Hence, if the loop is feasible, we do not need to check that the endurance is satisfied, because the truck is waiting and we recall that the considered loops are always feasible because of the definition of set F'. Consistent with the original model we do not count the launching time from the depot, but this can be easily

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

changed in the objective function:

$$\min \sum_{(i,j)\in A} \tau_{ij}^T x_{ij} + \sigma^R \sum_{\langle 0,j,k\rangle \in F} y_{0jk} + (\sigma^L + \sigma^R) \sum_{\substack{\langle i,j,k\rangle \in F \\ i \neq 0}} y_{ijk} + \sum_{i\in N_+} w_i + \sum_{\langle i,j,i\rangle \in F'} (\sigma^L + \tau_{ij}^D + \tau_{ji}^D + \sigma^R) y_{iji}. \tag{43}$$

Constraints (2) must be modified to allow loops to serve customers as in (44) and we must include constraints (45) to avoid loops starting from nodes served by sorties, to avoid multiple loops from the same node and to avoid loops starting when the drone is not in the truck. Finally, the binary variable y should also include the loops as in (46):

$$\sum_{i|(i,j)\in A} x_{ij} + \sum_{i,k|\langle i,j,k\rangle \in F'} y_{ijk} = 1 \quad j \in C \tag{44}$$

$$\sum_{j|\langle i,j,i\rangle \in F'} y_{iji} \leq z_i \quad i \in N_+ \tag{45}$$

$$y_{ijk} \in \{0,1\} \quad \langle i,j,k\rangle \in F'. \tag{46}$$

Formulation 3IF with loops is thus made of (6)–(15) and (43)–(46). Inequalities (18)–(21) can be included in the loop variant if slightly modified by considering the set of sorties F' instead of F, while inequality (17) can be included as it is.

5.1.2. 2IF with loops

To accommodate loops in formulation 2IF we need to use another rationale, because in such formulation two variables describe sorties and also loops ($\vec{g}_{ij}$ and $\overleftarrow{g}_{ji}$) and thus we cannot benefit from a 3-indexed variable describing the loops to include into the objective function as in 3IF. First of all we define a new set of arcs A_+, where arcs $(i,j) \in A_+$, $i \in N$, $j \in N_+$, $i \neq j$ include loops starting from the final depot (however loops from the depot can be easily avoided by restricting set A_+). A nonnegative variable is needed to account for the time in which the truck waits for the drone to perform the loop as in constraints (48), this variable will then be included in the objective function. Instead of using a new set of variables, one can use variables w for customers served with a loop. We recall that for a customer served with the drone the wait variable w is set to zero in the original 2IF, because its value is irrelevant, while in this variant it can be used to account for the time needed to perform a loop to serve that customer. We must take care of not considering inequalities (38)–(39), to avoid infeasibilities. Note that the launch and rendezvous times are already included in the objective function also for the loops but those starting from the final, which we include in the

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

new objective function (47):

$$\min \sum_{(i,j)\in A} \tau_{ij}^T x_{ij} + \sigma^L \sum_{\substack{(i,j)\in A_+ \\ i\neq 0}} \vec{g}_{ij} + \sigma^R \sum_{(j,k)\in A} \overleftarrow{g}_{jk} + \sum_{i\in N_+} w_i \tag{47}$$

$$w_j \geq \tau_{ij}^D + \tau_{ji}^D - M(2 - \vec{g}_{ij} - \overleftarrow{g}_{ji}) \quad i \in N_+, j \in C', i \neq j. \tag{48}$$

The formulation 2IF with loops must not include constraints (31) that avoid loops. Constraints (33) must be improved as in (50) that it is now needed in the formulation to avoid infeasible loops, while for the other sorties it remains a valid inequality. Constraints (24) and (25) must be substituted with constraints (49) to avoid including the time of a loop in the computation of variables t. Constraints (23), (27), and (28) must be modified as in (51), (52), and (53), respectively, to account for loops starting in the final depot. Constraint (54) must be added to avoid infeasible loops from the final depot and the variable domain (32) must be changed as in (55) and (56):

$$t_k \geq t_i + \tau_{ij}^D + \tau_{jk}^D - M(2 - \overrightarrow{g}_{ij} - \overleftarrow{g}_{jk}) \quad \langle i, j, k\rangle \in F \tag{49}$$

$$\sum_{i\in N_+} \tau_{ij}^D \vec{g}_{ij} + \sum_{k\in N_+} \tau_{jk}^D \overleftarrow{g}_{jk} + \sigma^R \leq E \quad j \in C' \tag{50}$$

$$\sum_{i|(i,j)\in A} x_{ij} + \sum_{i|(i,j)\in A_+} \vec{g}_{ij} = 1 \quad j \in C \tag{51}$$

$$\sum_{i|(i,j)\in A_+} \vec{g}_{ij} = \sum_{k|(j,k)\in A} \overleftarrow{g}_{jk} \quad j \in C' \tag{52}$$

$$\sum_{j\in C'} \vec{g}_{ij} \leq z_i \quad i \in N \tag{53}$$

$$\vec{g}_{nj} + \sum_{h\in N_0} \overleftarrow{g}_{jh} \leq 1 \quad j \in C' \tag{54}$$

$$\vec{g}_{ij} \in \{0, 1\} \quad (i, j) \in A_+ \tag{55}$$

$$\overleftarrow{g}_{ij} \in \{0, 1\} \quad (i, j) \in A. \tag{56}$$

Note that constraint (29), if included in the formulation as it is, may lead to suboptimal solutions when loops are an available sortie for the drone. Let us consider that, in an optimal solution, the truck travels along an arc (i, j) and the drone performs a loop from node i and starts a new sortie from node j. In such a case both variables z_i and z_j must equal 1 because a drone launch occurs in both i and j. However, if we compute (29) we can see that at most 1 between z_i and z_j can equal 1

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

($z_j \leq z_i - 1$), and thus the optimal solution is discarded. To overcome this, we need to introduce a new set of binary variables, namely l_i, that equals 1 if a loop is performed from node i, 0 otherwise. We can thus change constraint (29) to (57) and include constraints (58)–(61) to ensure that variables l work as by their definition:

$$z_j \leq z_i - x_{ij} + \sum_{k \in C'} (\overleftarrow{g}_{kj} - \vec{g}_{ik}) + l_i - l_j + 1 \quad (i, j) \in A \tag{57}$$

$$l_i \geq \vec{g}_{ij} + \overleftarrow{g}_{ji} - 1 \quad i \in N_+, j \in C', i \neq j \tag{58}$$

$$l_i \leq \sum_{j \in C'} \vec{g}_{ij} \quad (i, j) \in A_+ \tag{59}$$

$$l_i \leq \sum_{j \in C'} \overleftarrow{g}_{ji} \quad (j, i) \in A \tag{60}$$

$$l_i \in \{0, 1\} \quad i \in N. \tag{61}$$

Formulation 2IF with loops is then given by (3)–(5), (7), (12)–(15), (26), (30), and (47)– (61). Inequalities (17) and (34)–(37) can be used to strengthen the formulation.

5.1.3. 2IF-BC with loops

One can derive formulation 2IF-BC with loops from 2IF with loops as we built 2IF-BC from 2IF. The obtained 2IF-BC with loops is then given by (3)–(5), (7), (12), (13), (15), (26), (30), (40)–(42), and (47)–(56). Inequalities (36) and (37) can be used to strengthen the formulation.

5.2. Unlimited endurance

Another variant of the problem is the one for which the drone can fly indefinitely. This version can be achieved by imposing infinite endurance in the previously presented formulations; however, we can simplify the formulations to account for this feature. First of all, we need to redefine the set of feasible sorties F: for this variant a feasible sortie is a triplet $\langle i, j, k \rangle \in F, i \in N_0, j \in C', k \in N_+$ with no restriction on the endurance. Given the new set F, formulation 3IF with unilimited endurance is similar to formulation 3IF without constraints (8). Moreover, inequalities (20) and (21) can still be applied if using a "big-M" instead of E. Formulations 2IF with unilimited endurance and 2IF-BC with unilimited endurance can be obtained by formulation 2IF and 2IF-BC, respectively, by removing constraint (26), having defined the new set F as in the 3IF with unilimited endurance, and allowing variables $\vec{g}_{ij}$ for all $(i, j) \in A : \tau^D_{ij} > E$ and $\overleftarrow{g}_{jk}$ for all $(j, k) \in A : \tau^D_{jk} + \sigma_R > E$ to equal 1. Inequality (33) cannot be applied, while inequalities (38) and (39) can be applied if modified by substituting a "big-M" value to E.

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

5.3. Wait

A common variant in the literature is the one that allows the drone to wait on the ground. This feature can be easily included in the proposed formulations. Formulation 3IF wait is derived from formulation 3IF by removing constraint (8). Inequalities (20) and (21) can still be applied if using a "big-M" instead of E. Note that all the infeasible sorties are avoided with the preprocessing expressed by the definition of set F. Formulations 2IF wait and 2IF-BC wait can be obtained by formulation 2IF and 2IF-BC, respectively, by removing constraint (26). Inequality (33) becomes necessary, while inequalities (38) and (39) can be applied if modified by substituting a "big-M" value to E.

6. Computational experiments

The algorithms have been implemented in ANSI C. The mathematical models have been solved by Gurobi 8.1 on an Intel Core i3-2100 CPU, with 3.10 GHz and 8.00 GB of RAM. All computation times are obtained with single thread runs. All the extended results and the used instances could be found at the website www.or.unimore.it/site/home/online-resources.html.

Formulation 3IF and 2IF have been solved directly by the solver including the new inequalities, while 2IF-BC required a B&C implementation, since it includes an exponential number of constraints (42) that are incorporated dynamically. To separate these constraints, we have considered the residual graph $G' = (N, A')$ obtained from G by selecting the only arcs associated with a nonzero variable $(x, \vec{g}, \overleftarrow{g})$ in the continuous relaxation of the model. For the crossing sorties elimination constraints we explore the graph starting from depot 0, until a truck path violating one of the constraints is identified, if any. After preliminary computational tests, we have observed that formulation 2IF-BC benefited from separating these constraints only for integer solutions. In such a case, the overall cut separation procedure has a time complexity $O(|A'|)$. We also noted that 2IF-BC does not benefit from the use of inequalities (36)–(39) and thus we do not include them in the final implementation. As an initial upper bound, we provide the solver with a pure TSP solution computed by the Lin–Kernighan algorithm (see, e.g., Lin and Kernighan, 1973), implemented by Helsgaun Helsgaun (2000).

6.1. 10-Customer instances

We first run our methods on the 36 randomly generated benchmark instances proposed by Murray and Chu (2015), with 10 customers and the endurance E set to either 20 or 40, giving 72 instances. The customers are randomly distributed across an eight-mile square region, while the depot is located in four different positions: "a" indicates that the depot is near the center of gravity of the customers; while "b," "c" and "d" have the following (x, y) coordinates, respectively: (4.0,2.7), (4.0,0.0), and (4,−2.7). The drone times are based on Euclidean distances, while the truck times are based on Manhattan distances, and the ratio $|C'|/|C|$ is set between 80% and 90%. The values $\sigma^L = \sigma^R$ are set to 1.

In Table 3 one can find the results of our methods for $E = 20$, and in Table 4 the results for $E = 40$. All the instances are solved to the proven optimum. In these tables we show the in-

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Table 3
Results on the 10-customer instances from Murray and Chu (2015) with endurance $E = 20$

Instances					3IF			2IF			2IF-BC			DMN*
Name	Depot	*%sorties*	*opt*	*#sorties*	$\%gap_{RN}$	*#nodes*	Seconds	$\%gap_{RN}$	*#nodes*	Seconds	$\%gap_{RN}$	*#nodes*	Seconds	Min Seconds
37v1	"a"	29.9	*57.45*	0	8.20	470	0.8	7.24	424	1.3	11.06	1379	1.0	2.9
37v2	"b"	27.0	53.79	1	14.02	174	0.6	14.64	278	0.8	15.05	155	0.6	1.2
37v3	"c"	23.8	54.66	0	15.32	2191	1.9	19.05	3517	2.3	18.38	3201	2.2	12.6
37v4	"d"	21.5	67.46	0	12.53	1189	4.1	14.77	2702	3.5	14.25	2126	1.2	14.9
37v5	"a"	75.9	**51.78**	2	42.27	69,066	374.6	42.74	38,075	137.6	42.37	69,343	156.3	*t.l.*
37v6	"b"	75.5	48.60	2	50.30	39,797	189.8	50.47	18,762	71.9	50.33	21,050	53.7	1905.7
37v7	"c"	71.3	49.58	2	45.74	9963	30.6	45.90	4633	13.1	45.74	4863	9.6	204.3
37v8	"d"	65.3	62.38	2	36.30	6913	28.9	36.48	6110	18.5	36.48	6278	30.2	195.8
37v9	"a"	95.1	**43.48**	2	24.96	138,729	917.6	32.94	221,364	776.1	31.59	221,226	617.6	*t.l.*
37v10	"b"	95.1	41.91	3	37.85	30,741	193.3	44.06	35,165	102.1	42.19	39,940	104.8	2087.2
37v11	"c"	93.7	42.90	2	37.18	5559	113.1	37.22	1525	4.1	37.15	3600	64.5	46.5
37v12	"d"	89.7	56.85	3	30.16	4240	17.2	30.26	1964	10.8	30.10	2391	5.7	74.4
40v1	"a"	30.8	49.43	2	22.38	1686	2.2	22.38	1509	5.5	22.38	1178	3.0	42.1
40v2	"b"	30.9	51.71	1	26.93	2396	2.5	26.08	1994	2.2	27.63	1915	1.6	25.2
40v3	"c"	27.3	57.10	1	22.69	1701	1.7	22.69	954	1.1	24.04	1343	1.1	13.0
40v4	"d"	20.9	69.90	1	17.11	2572	1.9	18.20	1016	1.2	18.20	1475	1.1	2.0
40v5	"a"	81.7	45.46	2	16.59	31,215	179.5	15.60	25,124	104.8	15.60	20,832	57.7	1505.1
40v6	"b"	79.9	44.51	2	18.36	7245	27.2	18.36	2753	6.4	17.07	1855	6.0	72.5
40v7	"c"	79.0	49.90	2	11.54	2562	12.6	12.48	1692	6.8	11.54	1687	4.9	39.1
40v8	"d"	72.0	62.70	2	9.90	2271	7.3	9.18	1362	4.9	9.18	2139	5.0	25.6
40v9	"a"	99.5	42.53	4	9.80	1504	2.3	9.80	3023	2.8	9.80	4565	2.7	149.3
40v10	"b"	99.5	43.08	4	15.65	945	7.7	15.65	690	1.8	15.65	603	1.2	20.3
40v11	"c"	97.9	49.20	3	10.29	345	1.1	10.28	248	1.4	9.79	277	1.2	1.0
40v12	"d"	96.4	62.00	3	8.16	1262	6.3	7.61	152	1.2	7.82	197	1.0	1.1
43v1	"a"	26.0	69.59	0	5.42	87	0.6	5.59	71	0.5	5.44	81	0.5	0.6
43v2	"b"	22.3	72.15	0	4.11	77	0.5	4.06	41	0.5	4.06	199	0.6	0.5
43v3	"c"	17.9	77.34	0	4.63	131	0.5	4.63	82	0.5	4.79	82	0.5	0.9
43v4	"d"	17.3	90.14	0	4.25	169	0.4	4.26	157	0.5	3.97	79	0.4	0.4
43v5	"a"	66.6	58.71	2	21.23	120,395	351.7	21.22	59,222	137.9	24.13	83,928	164.6	1788.5
43v6	"b"	66.3	59.09	2	30.96	34,052	122.3	30.96	31,202	104.0	30.96	25,905	48.8	694.3
43v7	"c"	61.8	65.52	2	7.68	2188	8.0	7.68	2959	10.2	7.69	3015	7.2	22.3
43v8	"d"	53.3	84.81	1	13.59	32,780	67.9	13.59	12,503	29.0	13.59	13,963	26.8	276.5
43v9	"a"	88.7	**46.93**	3	23.30	132,709	674.1	23.30	160,449	446.6	23.16	219,372	705.2	*t.l.*
43v10	"b"	89.3	47.93	3	14.90	23,516	84.0	14.88	14,253	36.0	14.79	41,054	49.9	421.9
43v11	"c"	87.6	57.38	3	6.60	423	1.1	6.60	618	2.0	6.60	1972	6.8	5.0
43v12	"d"	81.7	69.20	3	4.05	447	1.0	4.05	156	0.7	4.05	971	1.0	2.4
Average					19.03	19,770	95.5	19.58	18,243	57.0	19.63	22,340	59.6	568.2

International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies
© 2021 The Authors.

Table 4
Results on the 10-customer instances from Murray and Chu (2015) with endurance $E = 40$

Instances				3IF				2IF			2IF-BC			DMN*
Name	Depot	*%sorties*	*opt*	*#sorties*	$\%gap_{RN}$	*#nodes*	Seconds	$\%gap_{RN}$	*#nodes*	Seconds	$\%gap_{RN}$	*#nodes*	Seconds	Min Seconds
37v1	"a"	89.7	**50.57**	1	8.20	413,169	3018.1	7.24	204,053	848.0	36.93	209,601	640.8	*t.l.*
37v2	"b"	89.5	47.31	1	14.02	90,548	418.1	14.64	29,786	134.8	46.93	28,718	73.5	1853.5
37v3	"c"	87.5	53.69	1	15.32	122,510	399.6	19.05	17,159	57.8	50.03	15,890	37.6	1099.3
37v4	"d"	81.6	67.46	0	12.53	397,706	1079.4	14.77	35,926	121.7	41.10	28,344	66.3	3204.1
37v5	"a"	100.0	**45.84**	2	42.27	252,415	1804.6	42.74	175,474	641.9	35.58	327,852	888.3	*t.l.*
37v6	"b"	100.0	**44.60**	2	50.30	200,119	1029.8	50.47	60,180	205.9	44.32	93,722	216.1	*t.l.*
37v7	"c"	100.0	47.62	2	45.74	115,932	532.6	45.90	37,616	156.4	42.85	38,399	92.0	2638.8
37v8	"d"	99.3	**60.42**	2	36.30	173,356	623.1	36.48	34,729	181.3	34.22	31,995	136.5	*t.l.*
37v9	"a"	100.0	**42.42**	2	24.96	248,823	769.2	32.94	89,273	282.9	29.54	215,608	510.5	*t.l.*
37v10	"b"	100.0	**41.91**	3	37.85	96,618	332.8	44.06	38,334	136.6	42.40	65,690	138.0	*t.l.*
37v11	"c"	100.0	42.90	2	37.18	3634	83.0	37.22	6668	15.7	37.16	4276	10.3	222.8
37v12	"d"	100.0	55.70	2	30.16	3940	70.7	30.26	3707	12.7	28.85	9535	59.4	235.1
40v1	"a"	95.3	46.89	1	22.38	69,697	398.5	22.38	37,933	164.9	19.17	41,963	111.3	2221.7
40v2	"b"	95.3	46.42	1	26.93	30,992	158.1	26.08	8089	26.2	21.73	9882	25.5	232.6
40v3	"c"	93.5	53.93	1	22.69	59,649	280.6	22.69	25,460	124.6	19.40	37,254	103.9	1820.6
40v4	"d"	89.4	**68.40**	1	17.11	152,859	599.8	18.20	60,789	202.4	17.73	35,781	88.9	*t.l.*
40v5	"a"	100.0	43.53	2	16.59	5168	22.0	15.60	6614	17.9	12.95	11,295	27.8	503.7
40v6	"b"	100.0	44.08	2	18.36	4998	23.0	18.36	1560	2.4	17.56	2762	5.4	104.8
40v7	"c"	100.0	49.23	3	11.54	2101	2.7	12.48	1064	1.3	11.26	2316	5.1	8.5
40v8	"d"	99.9	62.03	3	9.90	1479	8.5	9.18	1576	5.9	8.20	1983	4.4	9.7
40v9	"a"	100.0	42.53	4	9.80	1619	16.7	9.80	3495	3.6	9.80	3060	6.5	127.2
40v10	"b"	100.0	43.08	4	15.65	1441	2.4	15.65	384	0.6	14.29	1524	1.7	63.7
40v11	"c"	100.0	49.20	3	10.29	2480	2.8	10.28	439	0.7	10.22	1190	3.0	2.3
40v12	"d"	100.0	62.00	3	8.16	653	1.6	7.61	201	0.5	8.16	378	0.6	3.3
43v1	"a"	80.8	**57.01**	1	5.42	336,820	2019.4	5.59	162,276	652.3	36.85	211,273	749.0	*t.l.*
43v2	"b"	80.8	58.05	1	4.11	415,412	2078.0	4.06	82,508	350.0	29.73	138,585	397.8	3470.7
43v3	"c"	78.6	69.43	2	4.63	91,984	370.4	4.63	30,824	138.6	22.81	60,187	143.0	1271.4
43v4	"d"	70.9	83.70	1	4.25	111,234	369.3	4.26	41,640	135.9	20.68	49,460	107.9	1100.6
43v5	"a"	100.0	**52.09**	2	21.23	1,332,919	10,299.3	21.22	540,968	2268.2	30.89	895,968	4968.4	*t.l.*
43v6	"b"	100.0	**52.33**	2	30.96	420,700	2173.8	30.96	191,203	506.4	22.05	503,676	1466.6	*t.l.*
43v7	"c"	100.0	61.88	3	7.68	148,490	546.8	7.68	90,802	237.7	13.39	82,782	151.3	1625.9
43v8	"d"	100.0	73.73	3	13.59	63,154	238.2	13.59	44,568	124.5	9.95	108,904	198.5	1731.7
43v9	"a"	100.0	**46.93**	3	23.30	254,628	1009.6	23.30	194,000	499.6	23.29	461,916	1849.8	*t.l.*
43v10	"b"	100.0	47.93	3	14.90	33,197	125.9	14.88	71,665	140.1	14.90	49,459	95.9	1235.1
43v11	"c"	100.0	56.40	3	6.60	1454	2.4	6.60	778	1.0	4.97	2850	6.8	11.0
43v12	"d"	100.0	69.20	3	4.05	1156	2.8	4.05	588	0.8	4.05	615	0.8	0.8
Average					19.03	157,307	858.7	19.58	64,787	233.4	24.28	105,130	371.9	1788.9

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Table 5
Results on the 10-customer instances from Murray and Chu (2015) with respect to depot location

$E = 20$ Depot	3IF Seconds	2IF Seconds	2IF-BC Seconds	DMN* Min Seconds
"a"	278.2	179.2	189.8	1587.6
"b"	69.8	36.2	29.7	581.0
"c"	19.0	4.6	10.9	38.3
"d"	15.0	7.8	8.0	65.9
$E = 40$ Depot	3IF Seconds	2IF Seconds	2IF-BC Seconds	DMN* Min Seconds
"a"	2150.8	597.7	1083.6	2716.9
"b"	704.7	167.0	268.9	1973.4
"c"	246.8	81.5	61.5	966.7
"d"	332.6	87.3	73.7	1498.4

stance name, the depot location, the percentage of feasible sorties with respect to the total number (%*sorties*), the optimal solution value (*opt*), and the number of sorties in the optimal solution (#*sorties*). For each formulation, we show the %*gap* of the lower bound at the root node ($\%gap_{RN} = 100 \cdot (LB_{RN} - opt)/opt$), the number of visited nodes (#*nodes*), and the time needed by each of the exact methods to certify the optimal solution. Under DMN*, we report the minimum time among all the formulations proposed by Dell'Amico et al. (2019) to solve the same instances, recalling that they imposed one-hour time limit. The optimal solution value is written in bold if it is provided for the first time in this paper (3 values and 11 values for $E = 20$ and $E = 40$, respectively). Note that the solutions having zero sorties are also TSP solutions.

The results of Table 3 indicate a ranking of the formulations, with 2IF being able to converge faster than the other models, followed by 2IF-BC, that shows similar results, and 3IF. The inspection of Table 4 substantially supports the same conclusions, although here formulation 2IF-BC converges faster than 2IF for some of the instances, but 2IF appears to be faster when solving harder instances. This can be intuitively explained by considering that with $E = 40$ the drone has more freedom, and therefore more cuts need to be iteratively separated.

Concerning the optimality of the solutions, we recall that Murray and Chu (2015) could not solve, to the proven optimality, any of the instances within 30 minutes. On the other hand, the best method proposed by Dell'Amico et al. (2019) to solve the case in which the drone could not wait on the ground (that is harder than the opposite case), could solve 33 of 36 instances with $E = 20$ in about 570 seconds on average, and 25 of 36 instances with $E = 40$ in about 1800 seconds, on average, conseidring that it was run with an one-hour time limit (*t.l.* in Tables 3 and 4). Note that we have tested our algorithms on the same machine used by Dell'Amico et al. (2019), but we did not set a maximum time limit because all instances but 43v5 could be solved to the proven optimum within one hour by all methods. We can thus confirm that we significantly improved upon their methods.

In Table 5, one can find the average results of the various methods aggregated with respect to the depot location. One can see that the instances with the depot located near the center of gravity ("a" and "b") are the hardest to solve to optimality. We can conclude that, for these instances, the

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

closest the depot is located to the center of gravity, the hardest they are to solve, due to the larger number of feasible sorties.

6.1.1. Comparison with different endurance values

The benchmark instances by Murray and Chu have endurance $E = 20$ and $E = 40$; however, we believe that we can provide an interesting insight by testing the methods showing the best performances on the 10-customer instances (2IF and 2IF-BC) by varying the endurance values. In particular, we test instances with $E = 0$, which makes them TSP instances, and instances with a very large endurance $E = 100$, which is an endurance greater than all the optimal solution values. Between these two extremes, and besides endurance of 20 and 40, we also test $E = 60$ and $E = 80$. In Table 6, one can find these results averaged per depot location. We show the depot location, and, for each endurance, the percentage of feasible sorties with respect to the total number of sorties (*%sorties*), the time needed by 2IF and 2IF-BC to solve the instances to the proven optimum (no time limit has been set), and the number of sorties in the optimal solution (*#sorties*). Both methods solve all the TSP instances in less than one second. The solving times increase with the amount of endurance up to 60 for 2IF-BC, given that more and more sorties become feasible. The instances with endurance that equals 60, 80, and 100 do not provide an improvement in the solution value (as we will discuss in the following) but the solving times slightly worsen with the increase in E. Formulation 2IF shows the best results on average, with a peak in solving times when endurance is $E = 40$, being able to solve all the instances to optimality in less than 40 minutes, in the worst case. 2IF-BC could solve all the instances in less than 40 minutes but instance 43v5 that needs about 2 hours 45 minutes to be solved to optimality when $E = 60$, 80, and 100.

Another interesting insight is to analyze the solutions with different endurance values, in particular to show the percentage improvement with respect to the corresponding TSP solution. Among the instances with $E = 20$ only 7 replicate the TSP solution, and on average the improvement is 12.77%. The improvement obtained by setting $E = 40$ is even more consistent, with an average value of 17.60%. Among these instances, 10 keep the same value obtained with $E = 20$ and 1 replicates the TSP solution. Only six instances with $E = 60$ improve upon those with $E = 40$. From that value of endurance, no instance provides an improvement in the solution value, as one can clearly see in the corresponding box and whiskers graph in Fig. 1.

6.2. 20-Customer instances

Since all 10-customer size instances could be solved to optimality, we challenge our methods by testing them with instances having more than 10 customers. In Murray and Chu (2015), the authors also propose a set of 120 instances with 20 customers for the *parallel drone scheduling TSP*, a problem in which a fleet of drones can serve a set of customers only departing from the depot, the remaining customers are served by a truck. Those instances can be easily adapted to the FSTSP. For $\sigma^L = \sigma^R = 1$, we use the values similar to those used for the 10-customer instances and we set the endurance to 20 and 40 unit times. A total of 120 instances with $E = 20$ and 120 instances with $E = 40$ is thus obtained. Each set of 120 instances is equally divided by three different depot locations: "centered" (in which the depot is centrally located with respect to the customers), "edge"

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Table 6
Solving times to get to the optimal solution with different values of endurance (E) obtained using 2IF and 2IF-BC on the 10-customer instances from Murray and Chu (2015)

Depot	%*sorties*	2IF (seconds)	2IF-BC (seconds)	#*sorties*	%*sorties*	2IF (seconds)	2IF-BC (seconds)	#*sorties*	%*sorties*	2IF (seconds)	2IF-BC (seconds)	#*sorties*
	$E = 0$				$E = 20$				$E = 40$			
"a"	0.0	0.1	0.1	0.0	66.0	179.2	189.8	1.9	96.2	597.7	1083.6	2.0
"b"	0.0	0.1	0.1	0.0	65.1	36.2	29.7	2.0	96.2	167.0	268.9	2.1
"c"	0.0	0.2	0.2	0.0	62.3	4.6	10.9	1.7	95.5	81.5	61.5	2.2
"d"	0.0	0.2	0.2	0.0	57.6	7.8	8.0	1.7	93.4	87.3	73.7	2.0
Average	0.0	0.2	0.1	0.0	62.7	57.0	59.6	1.8	95.3	233.4	371.9	2.1
	$E = 60$				$E = 80$				$E = 100$			
"a"	99.9	345.3	1628.0	2.0	100.0	331.2	1652.2	2.0	100.0	371.5	1667.4	2.0
"b"	99.9	120.8	169.3	2.1	100.0	128.1	178.3	2.1	100.0	120.8	190.2	2.1
"c"	99.9	133.4	197.2	2.2	100.0	153.8	87.4	2.2	100.0	160.2	87.1	2.2
"d"	99.7	229.5	249.1	2.1	99.99	308.3	251.1	2.1	100.0	187.6	178.9	2.1
Average	99.9	207.2	560.9	2.1	100.0	230.4	542.3	2.1	100.0	210.3	530.9	2.1

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

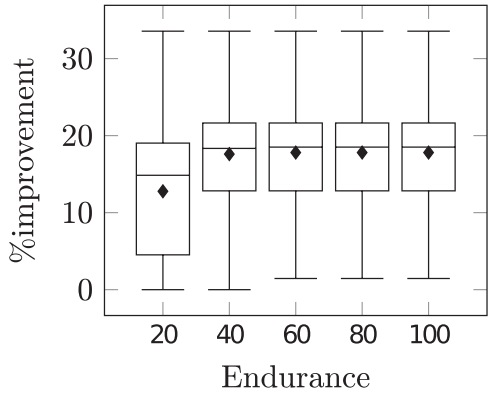

Fig. 1. Comparison between the optimal solution value with $E = 0$ (the TSP solution) and that of instances with different values of endurance averaged on the 10-customer instances. The lower and upper values of the box are the first and the third quartiles, respectively, the central line is the median and the diamond is the average. The whiskers include the minimum and maximum values.

Table 7
Summarized results on the Murray and Chu (2015) 20-customer instances

$E = 20$	2IF (3600 seconds)					2IF-BC (3600 seconds)				
Depot	*#opt*	*#best*	*%gap*	*#nodes*	Seconds	*#opt*	*#best*	*%gap*	*#nodes*	Seconds
"Centered"	18	35	3.87	97,889	819.5	20	29	3.12	164,685	973.5
"Edge"	3	26	13.88	145,415	1323.2	4	24	12.80	244,757	1748.7
"Origin"	31	37	1.53	104,920	796.5	34	37	0.88	181,619	899.1
Sum/average	52	98	6.42	104,823	834.9	58	90	5.60	180,108	984.8
$E = 40$	2IF (3600 seconds)					2IF-BC (3600 seconds)				
Depot	*#opt*	*#best*	*%gap*	*#nodes*	Seconds	*#opt*	*#best*	*%gap*	*#nodes*	Seconds
"Centered"	0	19	16.18	-	-	0	4	15.21	-	-
"Edge"	0	10	23.65	-	-	0	12	21.28	-	-
"Origin"	2	18	17.28	225,486	2847.4	2	21	15.67	197,621	2004.8
Sum/average	2	47	19.04	225,486	2847.4	2	38	17.39	197,621	2004.8

(on the edge of the squared area where the customer are generated), and "origin" (at the origin of the axes of that area).

In Table 7, we report the results achieved by the most promising formulations (2IF and 2IF-BC) on the set of 20-customer instances, with endurance $E = 20$ and $E = 40$, and with depot located centrally, on the edge, or at the origin. We show the number of optima (*#opt*) and best solutions (*#best*) obtained by each of the methods, and the following gaps: $\%gap = 100 \cdot (best_{UB} - LB)/best_{UB}$, where $best_{UB}$ is the best upper bound among the exact methods, LB is the lower bound of the method considered each time. We also show the average solving times (seconds) with 3600 seconds as time limit and the average number of nodes of the enumerations tree (*#nodes*) at the end

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Table 8
Comparison between 2IF, 2IF-BC, and the branch-and-bound (B&B) by Poikonen et al. (2019) on the FSTSP variant with loops

				B&B		2IF				2IF-BC			
$\|C\|$	*#inst.*	*#sorties*	*#loops*	*#opt*	Seconds	*#opt*	$\%gap_{RN}$	*#nodes*	Seconds	*#opt*	$\%gap_{RN}$	*#nodes*	Seconds
9	100	2.99	0.44	100	77.8	100	36.74	2443	4.4	100	34.05	2412	3.6
14	52	5.37	0.17	0	*t.l.*	52	43.16	95745	917.9	52	41.13	96741	602.9

of the iterations. Note that both the solving times and the number of nodes are averaged on the instances that could be solved to optimality.

Table 7 suggests that 2IF and 2IF-BC are able to obtain good results on the 20-customer instances when $E = 20$. In particular, 2IF-BC performs on average better both in terms of best heuristic solution retrieved and gaps, although in some cases 2IF finds better heuristic solutions. Formulation 2IF-BC is able to obtain the proven optimal solution for 58 instances, while 2IF stops at 52. Moreover, 2IF-BC is also faster than 2IF. The results of Table 7, where the endurance E is increased to 40, show that we have reached the limit of the proposed methods. Comparing the two approaches, for several instances 2IF has better heuristic results while 2IF-BC shows slightly better lower bounds.

Concerning the depot location, it is easy to note that the instances having depot located at the center or at the origin of the axes are those that our methods could solve more easily. The depot located at the origin could be compared with the depot "d." On the other side, these instances show that the depot located at the edge of the generation area are harder to solve, in contrast to what shown for the previous instances with depot of type "c".

7. Comparison with similar problems from the literature

The last set of computational results is focused on comparing 2IF and 2IF-BC with methods from the literature that solve other variants of the FSTSP. Poikonen et al. (2019) solve an FSTSP in which loops are allowed, every customer can be visited by the drone, no service time is considered ($\sigma^L = \sigma^R = 0$), and a maximum endurance is fixed. Their instances are made of randomly generated locations in a 50×50 grid in which the coordinates are uniformly distributed; among them one is designated as the depot. They then compute Manhattan distances to be used as truck travel times and Euclidean distances divided by a number α (to represent how fast the drone can be with respect to the truck) as drone travel times. In particular, they have a first set of a hundred 9-customer instances and a second one of fifty-two 14-customer instances on which they test their exact method. Endurance is set to 20 and $\alpha = 2$.

In Table 8 we show the aggregated results for these two set of instances, in particular we show the number of customers, the number of instances for each set, the number of instances solved to optimality by the B&B proposed by Poikonen et al. (2019), and the average seconds needed by their algorithm to solve to optimality the instances (when possible) using a computer with an i7-6700 processor operating at 3.4 GHz, 16 GB of RAM, and no parallelization. In Table 8, *t.l.* is used to represent the sentences "not a single instance was solved after several hours of testing" reported in

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Table 9
Comparison between 2IF and the compact formulation (CF-RR) by Roberti and Ruthmair (2020) on the FSTSP variant with loops

					CF-RR		2IF				2IF-BC			
$\|C\|$	α	*#inst.*	*#sorties*	*#loops*	*#opt*	Seconds	*#opt*	$\%gap_{RN}$	*#nodes*	Seconds	*#opt*	$\%gap_{RN}$	*#nodes*	Seconds
9	1	25	1.5	0.16	25	2.2	25	12.22	342	0.4	25	10.95	359	0.4
9	2	25	3.1	0.40	25	19.9	25	35.67	1671	2.8	25	32.33	1811	2.5
9	3	25	3.8	0.48	25	47.0	25	48.51	3998	12.7	25	47.17	4775	11.4
19	1	25	4.8	0.13	3	1044.1	23	26.31	142,122	717.2	23	23.97	228,719	724.6
19	2	25	8.3	0.00	0	*t.l.*	3	40.66	25,023	1023.2	3	38.55	42,741	965.1
19	3	25	-	-	0	*t.l.*	0	-	-	*t.l.*	0	-	-	*t.l.*

Poikonen et al. (2019) about their method on the 14-customer instances. Under the columns 2IF and 2IF-BC, we show the time needed for our methods to solve all the 9- and 14-customer instances with our computer. We also show the average number of sorties (*#sorties*) and loops (*#loops*) in the optimal solutions (whose number is given by *#opt*), the *%gap* at the root node, and the average number of nodes of the enumeration tree to get to optimality (*#nodes*). The average number of sorites and loops in teh table follow this rule: when equivalent solutions having different number of sorties and loops where at hand we selected the one with fewer sorties and, in case of an equivalent number of sorties, the one with fewer loops. The number of sorties includes that of loops. Note that both 2IF and 2IF-BC clearly outperform their method on the 9-customer instances because we need only 4.4 and 3.6 seconds, respectively, on a slower computer to solve each of the 9-customer instances, on average. Moreover, 2IF and 2IF-BC could obtain the optimal solution for all the 14-customer instances in about 15 and 10 minutes, respectively, on average and exceed one hour only for two and one instances, respectively. 2IF-BC provides better performances with respect to 2IF.

Poikonen et al. (2019) also define another set of randomly generated instances that they mainly use to compare the results of the heuristic methods they propose. These instances include a set of 25 instances with 9, 19, 29, 39, 49, 59, 69, 79, 89, 99, and 199 customers, with $E = 20$, computing truck times based on Manhattan distances and drone times based on Euclidean distances, these last instances divided by $\alpha = 2$. Roberti and Ruthmair (2020) use the instances with 9, 19, 29, and 39 customers with $\alpha = 1, 2, 3$ as their test bed and also test their methods when solving the FSTSP with loops as proposed by Poikonen et al. (2019). They compare their BP with their CF, concluding that 39 customer is the limit of their best method, BP, which also outperforms 2IF and 2IF-BC; however, we intend to compare our CFs with their CF, that we now call CF-RR. In Table 9 we show the number of customers for each instance $|C|$, the used α, the number of instances of that type (*#inst.*), the number of instances solved to optimality by each of the methods (*#opt*), and the time needed to prove optimality. One can see that 2IF and 2IF-BC are faster than CF-RR when solving the 9-customer instances and that could solve 23 more 19-customer instances in shorter times. When solving these instances, 2IF-BC shows similar or better results than 2IF, on average. In Table 9 we also show the number of nodes of the enumeration tree (*#nodes*), the *%gap* at the root node, and the number of sorties (*#sorties*) and loops (*#loops*) averaged on all the optimal solutions at hand. The number of sorties includes the number of loops. Note that different equivalent solutions with different numbers of sorties and loops were available, we thus reported the smaller

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Table 10
Comparison between 2IF, 2IF-BC, and the compact formulation (CF-RR) by Roberti and Ruthmair on the FSTSP variant with unlimited endurance, no incompatible customers, and no service times

				CF-RR		2IF				2IF-BC			
$\|C\|$	α	*#inst.*	*#sorties.*	*#opt*	Seconds	*#opt*	$\%gap_{RN}$	*#nodes*	Seconds	*#opt*	$\%gap_{RN}$	*#nodes*	Seconds
9	1	25	2.4	25	7.7	25	60.59	62,174	161.9	25	62.80	113,308	270.9
9	2	25	4.0	25	7.4	25	53.82	48,979	100.6	25	54.04	96,743	223.0
9	3	25	4.5	25	5.1	25	47.10	14,870	23.0	25	47.41	34,798	60.2

number of sorties and loops. Their number increases with the number of customers and the speed of the drone, however the use of loops is rare. A similar behavior is shown by the %*gap* at the root node.

Roberti and Ruthmair (2020) use the same test bed to test their CF-RR on their basic version of the problem that they call TSP-D, an FSTSP with unlimited endurance ($E = \infty$) (and thus of course allowing the drone to wait at customers), no launch and rendezvous times ($\sigma^L = \sigma^R = 0$), and no incompatible customers. Note that in this case, all computed distances are subject to the floor function. In Table 10, we compare CF-RR with 2IF and 2IF-BC on this version of the problem. Both methods could solve to optimality all the 9-customer instances and none of the 19-customer instances (that we do not show in the table for this reason). CF-RR outperforms our methods on this version of the problem; however, Roberti and Ruthmair (2020) use a more powerful machine (Intel Xeon E5-2670v2, 2GHz, 8 GB RAM). We can thus conclude that CF-RR appears to be more suitable to solve problems with unlimited endurance and 2IF-BC those with a more binding restriction on the endurance value. If one wants to use a CF one should use CF-RR to solve a basic TSP-D instance, and 2IF and 2IF-BC to solve the FSTSP and the FSTSP with loops. In Table 10 we also show the number of sorties used in the optimal solution averaged on the three set of instances. Note that several solutions with the same optimal value but with different numbers of sorties are available for some instances. In the table, the average is computed on the solutions at hand presenting the minimum number of sorties. The larger the α, the faster is the drone, and, as expected, the larger is the number of sorties performed and the easier is to find the optimal solution. The %*gaps* at the root node ($\%gap_{RN}$) are very large and similar for both 2IF and 2IF-BC, but they decrease with the increment of the speed of the drone.

The best method presented by Schermer et al. (2020) could solve the 10-customer instances by Murray and Chu (2015) (see Section 6.1) within 16 seconds, on average; however, their best method could only solve less than 15% of the 20-customer instances presented in the same paper (those used in Section 6) while our best method could solve to optimality the 25% of them, presenting better %*gaps* and solving times. When solving the TSP-D with loops, their best method can provide slightly better results on instances with 14 customers and on some instances with 20 customers. We recall, however, that they use an Intel Xeon E5-2640v3, 2.6 GHz, 8 GB RAM, that is a much faster machine with respect to ours.

8. Conclusions

In this paper, we proposed novel formulations for the FSTSP. Extensive computational tests showed that the best formulation could solve to optimality for the first time all the benchmark instances

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

from the literature, with 10 customers, in reasonable time. Many instances with 20 customers could be solved to optimality as well, indicating that the proposed formulations could be used effectively for instances of small/medium size. We also applied our methods to variants of the literature being able to solve many instances, showing that our methods can be better than other compact formulations from the literature. In particular, our methods obtained promising results for variants including loops. Future works should explore problems with multiple trucks and drones, taking into account other real-world side constraints.

References

Agatz, N., Bouman, P., Schmidt, M., 2018. Optimization approaches for the traveling salesman problem with drone. *Transportation Science* 52, 4, 965–981.

Applegate, D., Bixby, R., Chvatal, V., Cook, W., 2006. Concorde TSP solver. Available at http://www.tsp.gatech.edu/concorde/.

Bouman, P., Agatz, N., Schmidt, M., 2018. Dynamic programming approaches for the traveling salesman problem with drone. *Networks* 72, 4, 528–542.

Dell'Amico, M., Montemanni, R., Novellani, S., 2019. Drone-assisted deliveries: new formulations for the flying sidekick traveling salesman problem. *Optimization Letters* 15, 1–32.

Dell'Amico, M., Montemanni, R., Novellani, S., 2021a. A random restart local search matheuristic for the flying sidekick traveling salesman problem. ICIEA 2021: The 8th International Conference on Industrial Engineer and Applications. ACM, New York, NY.

Dell'Amico, M., Montemanni, R., Novellani, S., 2021b. Algorithms based on branch and bound for the flying sidekick traveling salesman problem. *Omega* 104, https://doi.org/10.1016/j.omega.2021.102493.

de Freitas, J.C., Penna, P.H.V., 2018. A randomized variable neighborhood descent heuristic to solve the flying sidekick traveling salesman problem. *Electronic Notes in Discrete Mathematics* 66, 95–102.

de Freitas, J.C., Penna, P.H.V., 2020. A variable neighborhood search for flying sidekick traveling salesman problem. *International Transactions in Operational Research* 27, 267–290.

Ha, Q.M., Deville, Y., Pham, Q.D., 2017. On the min-cost traveling salesman problem with drone. *Transportation Research. Part C: Emerging Technologies* 86, 597–621.

Ha, Q.M., Deville, Y., Pham, Q.D., Hà, M.H., 2015. Heuristic methods for the traveling salesman problem with drone. arXiv:1509.08764v1.

Ha, Q.M., Deville, Y., Pham, Q.D., Hà, M.H., 2020. A hybrid genetic algorithm for the traveling salesman problem with drone. *Journal of Heuristics* 26, 2, 219–247.

Helsgaun, K., 2000. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research* 126, 1, 106–130.

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.

Lin, S., Kernighan, B.W., 1973. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research* 21, 2, 498–516.

Liu, L., 2018. Optimization of drone-assisted delivery system. *Optimization Methods in Engineering* 3. Available at https://openscholarship.wustl.edu/mems5001/3.

Marinelli, M., Caggiani, L., Ottomanelli, M., Dell'Orco, M., 2017. En route truck–drone parcel delivery for optimal vehicle routing strategies. *IET Intelligent Transport Systems* 12, 4, 253–261.

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.

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.

Poikonen, S., Golden, B., Wasil, E.A., 2019. A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing* 31, 2, 335–346.

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies

Ponza, A., 2016. Optimization of drone-assisted parcel delivery. Master's thesis, Universita Degli Studi di Padova, Padova.

Reinelt, G., 1991. TSPLIB—a traveling salesman problem library. *ORSA Journal on Computing* 3, 4, 376–384.

Roberti, R., Ruthmair, M., 2020. Exact methods for the traveling salesman problem with drone. *Transportation Science* 55, 2, 315–335.

Schermer, D., Moeini, M., Wendt, O., 2020. A branch-and-cut approach and alternative formulations for the traveling salesman problem with drone. *Networks* 76, 2, 164–186.

Yurek, E.E., Ozmutlu, H.C., 2018. A decomposition-based iterative optimization algorithm for traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies* 91, 249–262.

© 2021 The Authors.
International Transactions in Operational Research published by John Wiley & Sons Ltd on behalf of International Federation of Operational Research Societies