Received: 22 May 2020 | Accepted: 22 May 2020 | Published on: 22 June 2020

DOI: 10.1002/net.21958

SPECIAL ISSUE ARTICLE

A branch-and-cut approach and alternative formulations for the traveling salesman problem with drone

Daniel Schermer | **Mahdi Moeini** | **Oliver Wendt**

Business Information Systems and Operations Research (BISOR), Technische Universität Kaiserslautern, Kaiserslautern, Germany

Correspondence
Daniel Schermer, Business Information Systems and Operations Research (BISOR), Technische Universität Kaiserslautern, Gottlieb-Daimler-Straße, 67663 Kaiserslautern, Germany.
Email: daniel.schermer@wiwi.uni-kl.de

Abstract

In this paper, we are interested in studying the *traveling salesman problem with drone* (TSP-D). Given a set of customers and a truck that is equipped with a single drone, the TSP-D asks that all customers are served exactly once and minimal delivery time is achieved. We provide two compact mixed integer linear programming formulations that can be used to address instances with up to 10 customer within a few seconds. Notably, we introduce a third formulation for the TSP-D with an exponential number of constraints. The latter formulation is suitable to be solved by a branch-and-cut algorithm. Indeed, this approach can be used to find optimal solutions for several instances with up to 20 customers within 1 hour, thus challenging the current state-of-the-art in solving the TSP-D. A detailed numerical study provides an in-depth comparison on the effectiveness of the proposed formulations. Moreover, we reveal further details on the operational characteristics of a drone-assisted delivery system. By using three different sets of benchmark instances, consideration is given to various assumptions that affect, for example, technological drone parameters and the impact of distance metrics.

KEYWORDS

branch-and-cut, drones, last-mile delivery, logistics, mixed integer linear programming, traveling salesman problem

1 | INTRODUCTION

Probing drones and other autonomous units in last-mile delivery is a current trend that is investigated by retailers and logistic service providers such as Amazon.com, Inc. or Deutsche Post DHL Group (see [20] and references therein). The integration of these unmanned units into existing logistic architectures might yield synergetic benefits that could emerge in terms of time or cost savings, both of which might be crucial for effectively carrying out cost-efficient same-day deliveries in the future (refer to [2]). In fact, drones have already become a proven technology in many related public and private sectors including energy, agriculture and forestry, environmental protection, and emergency response (see [21] and references therein).

Generally, we might accept that a drone can move faster between two locations than a truck as a drone has a larger average velocity, can move as the crow flies, and is not subjected to the limitations of a road network such as, for example, congestion. In addition, drones are far more lightweight than trucks, which might cause them to consume less energy when moving from one point to another. However, a drone's payload weight and cargo capacity in terms of volume are inherently limited. Moreover, as multirotor drones rely on comparatively small batteries for powering their flight, their range of operation is quite restricted compared to a delivery truck.

Consequently, due to the complementary nature of both vehicles, combining a truck with a drone gives rise to several new problems in logistics. In recent years, these problems have received a rising interest from the optimization community. Indeed, several new problems have been proposed in the academic literature in which different possibilities of integrating drones are studied (see, e.g., [1,17,20,30]). In an attempt to classify these problems, Otto et al. [21] proposed the following criteria that allow for a differentiation (for more details, we point to the aforementioned paper and the references therein):

1. *Drones as the main performance drivers*, that is, trucks supporting the operations of drones: this might be the case if, for example, a truck merely serves as a (passive) carrier for a drone.
2. *Trucks as the main performance drivers*, that is, drones supporting the operations of trucks: this case might occur if, for example, a truck is merely resupplied by a drone during its tour.
3. *Drones and trucks performing independent tasks*: in such scenarios, a truck and a drone might be located at a common distribution center but perform deliveries without interacting with one another.
4. *Drones and trucks as synchronized working units*: these approaches have received significant attention in the literature and might be further classified as follows:

   - *Sidekick-based approaches* (see, e.g., [1,20]): in these cases, a drone is assumed to be carried along by a delivery truck. However, the truck is not just a passive carrier, that is, customers can be served by either truck or drone.
   - *Station-based approaches* (see, e.g., [17,29]): here, the truck and drone(s) might be located at distinct distribution centers resembling multiple echelons. While the truck could initially be at a central depot, the drone(s) might be located at a remote *drone station*. A drone station could be a low-cost infrastructure located in close proximity to demand centers that contains one or more drones, for example, unmanned (aerial or ground) vehicles.

This paper is concerned with the *traveling salesman problem with drone* (TSP-D) that is a sidekick-based approach. As initially identified by Murray and Chu [20] and confirmed in many subsequent works (see, e.g., [1,3,22]), finding optimal solutions to these kinds of problems poses a significant computational challenge. In fact, these problems are strongly involved with several aspects of *synchronization* (refer to [10]). Consequently, from an algorithmic point of view, currently, no efficient method exists for computing optimal solutions even to small problems in short computation time. While it might be reasonable to accept (unless $\mathcal{P} = \mathcal{NP}$) that no such method might exist, given the $\mathcal{NP}$-hard nature of the underlying problem, we still believe that there is meaning in accelerating the search process for provably optimal solutions to small instances. Therefore, the contributions of our work are as follows:

1. As an alternative to existing formulations in the literature (see, e.g., [1,20]), we introduce two novel *mixed integer linear programming* (MILP) formulations as well as some effective *valid inequalities* (VIEQs) (see [27,29]). At this point, instances with just 9 to 10 customers are considered to be quite difficult to be solved optimally within reasonable time. By themselves, using a state-of-the-art solver, our proposed formulations can be used to solve such instances within a few seconds.
2. Moreover, we propose a third formulation with an exponential number of constraints that is suitable be integrated in a branch-and-cut solver. Arguably, motivated by the VIEQs, this method leverages the strengths of existing and *compact* MILP formulations (e.g., [20,27,29]) by merging them with the basic idea of the *set-covering* based *integer linear programming* (ILP) approach proposed by Agatz et al. [1]. Through this combined approach, we can address several instances with up to 20 customers in less than 1 hour. As a consequence, some further instances are solved up to optimality.
3. In any case, through our numerical study, we confirm previous observations that a drone-assisted truck might allow for a significantly reduced delivery time, and provide some further insights into the operational characteristic of such a delivery system. Furthermore, we show that, using our formulations, different benchmark instances and their associated assumptions have a noticeable influence on the performance of the MILP solver and the effectiveness of the truck-drone delivery system.

The remainder of this paper is organized as follows. We begin in Section 2 with a brief overview of the related literature. Afterwards, Section 3 introduces the notation, two compact mathematical models, and the proposed valid inequalities for the TSP-D. As an alternative, a third formulation with an exponential number of constraints is discussed in Section 4. We present the detailed results of our numerical studies in Section 5. Finally, concluding remarks on this work and future research implications are drawn in Section 6.

2 | RELATED LITERATURE

For the purpose of this paper, we restrict ourselves to works that consider the *sidekick-based* approach (see Figure 1) under a minimal delivery time objective (refer to [1,20]) as these problems have intensive *synchronization* requirements (refer to [10]). In

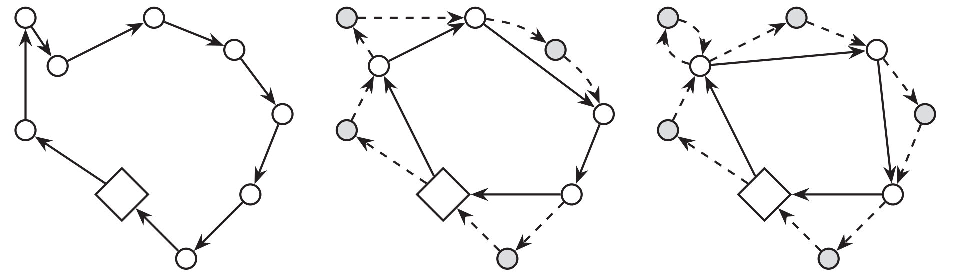

Figure 1 An illustration of a possible TSP, FSTSP, and TSP-D solution (from left to right). The paths of the truck and drone are indicated by solid and dashed lines, respectively. In each figure, the depot is indicated by the square shape

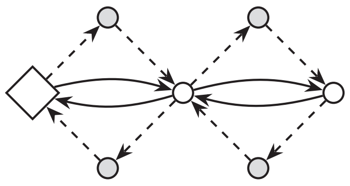

Figure 2 In the TSP-D, it might be useful to visit vertices repeatedly to increase the utilization of the drone

Section 2.1, we provide a brief account on the development of novel (M)ILP formulations or any other exact method that comes with an optimality certificate. Afterwards, in Section 2.2, we highlight some additional works that are primarily concerned with heuristics. The works presented in these sections have some overlaps. In fact, many of the authors that introduced a novel MILP formulation, also developed heuristics for solving the problem (see, e.g., [1,20]). More general literature reviews on the potentials of utilizing drones in a civil manner in and outside the context of vehicle routing problems can be found in [21,25].

2.1 | Model-based literature

Murray and Chu [20] introduced the *flying sidekick traveling salesman problem* (FSTSP) that aims at finding a drone-assisted *traveling salesman problem* (TSP) tour. In this problem, it is assumed that a depot and a set of customers are given, each of whom must be served exactly once (see Figure 1). Initially, the truck-drone tandem is located at a central depot. The objective is to find a feasible routing of the truck and drone such that all customers are served and both vehicles have returned to the depot as quickly as possible, that is, with minimal *makespan*. The authors propose a MILP formulation for solving the FSTSP and specify a drone *sortie* by a triple (i, j, k): the drone is launched at vertex i, serves customer j, $i \neq j$, and is retrieved by the truck at a third vertex k, $j \neq k$. Notably, in their model, *round trips* are forbidden, that is, for every sortie (i, j, k) the following condition must hold: $i \neq k$. Using a state-of-the-art MILP solver, the authors cannot find any provably optimal solution within a 30-minute time limit for instances with just 10 customers. As an alternative, Murray and Chu [20] propose a heuristic for solving the problem.

The TSP-D was proposed by Agatz et al. [1] and shares most of the assumptions of the FSTSP. However, there are a few key differences to the FSTSP. First, the drone can be retrieved at the same location from which it was launched (see Figure 1). Second, the truck may visit customers more than once as this could be beneficial in operating the drone (see Figure 2).

Agatz et al. [1] propose a set-covering based ILP formulation for solving the TSP-D. Within this approach, they use an *operation* as the essential building block (see also Figure 4): an operation is characterized by an ordered sequence of at least two (or more) vertices as visited by the truck. Within this sequence, at most one customer can be served by the drone through a sortie that connects the start and end of the sequence. Using the ILP formulation, the authors can tackle instances with nine customers within no more than 40 seconds. However, due to the exponential number of operations w.r.t. the number of customers, they also report that this approach does not scale well. In fact, on instances with just 11 customers, several hours are required to obtain optimal solutions. Notably, some theoretical insights and heuristics that leverage local search and *dynamic programming* (DP) are also provided in this work.

In a follow up paper, Bouman et al. [3] provide another method that is based on DP. The authors highlight that the new DP method is able to find optimal solutions much faster compared to the ILP approach presented in [1]. In fact, the speedup is about fortyfold on instances with nine customers. However, it also becomes apparent that the performance degrades as the instance size is increased: for provably optimal solutions, instances with 12 customers already take 1 minute, an increase to 15 customers requires more than 2 hours of computation time, and no solution is found within 12 hours beyond this size. To overcome this issue, they propose a heuristic variant of their DP approach where only a subset of feasible operations is considered. The authors can show that a considerable speedup can be achieved through this procedure with just a slight loss on the solution quality w.r.t. optimal solutions.

Es Yurek and Ozmutlu [11] provide an exact method for solving the TSP-D that is based on a decomposition of the problem into two components. First, all feasible tours to the TSP are generated that visit every possible subset of customers. Each tour provides a *lower bound* (LB) on a TSP-D solution in which the customers that are not covered by the tour must be served by the drone. Starting with the shortest tour (smallest LB), mathematical programming is used to identify the optimal drone sorties. This procedure continues until the best remaining LB is worse than the incumbent TSP-D solution. The authors compare their results to [1,20] and highlight that they can solve instances with 10 customers within 5 minutes and instances with 12 customers within 30 minutes. Although, the authors stress that performance depends largely on the type of instance as well as parameters (e.g., speed of the drone). In particular, their approach performs poorly on densely clustered instances: here, for the case of 12 customers, the algorithm rarely terminates within 1 hour.

In the work of Poikonen et al. [22], a *branch-and-bound* method is introduced for solving the TSP-D. Each node in the search tree represents a delivery sequence as visited by the truck that might contain only a subset of customers. At each node, for the customers not covered by this sequence, the optimal TSP-D solution can be computed through the DP algorithm proposed in [1]. During the search, child nodes are created by inserting customers into the parental sequence. Poikonen et al. [22] stress that, in the TSP-D, the objective value could also improve if an additional customer is inserted into the sequence. Therefore, as we climb the tree, for some children, the objective value obtained through the DP can in fact be smaller than that of its ancestors. The authors introduce the *tree exploration ratio* as a parameter that provides a tradeoff on optimality guarantee vs speed. With an optimality guarantee, they can solve instances with nine customers in just 1 minute. However, the authors also report that the solution time becomes intractably large when the number of customers is increased to 14.

Dell'Amico et al. [7] propose a modified FSTSP model that builds on the foundation laid by Murray and Chu [20]. In particular, they propose a new formulation in which some explicit constraints are replaced with exponentially many constraints that can be added in a cutting plane fashion. The affected constraints concern restrictions regarding, for example, the availability of the drone and the correct orientation of sorties (w.r.t. the route of the truck). Dell'Amico et al. [7] can show that these changes lead to an improvement in performance as they can solve several instances with 10 customers optimally within 1 hour. Moreover, in a subsequent work, Dell'Amico et al. [8] propose a more compact formulation. This is achieved by explicitly accounting for the drone's availability at each vertex (a similar approach is used in [27]). The authors can show that the branch-and-cut formulation yields favorable results as instances from [20] with 10 customers can be solved in about 15 minutes on average. Notably, the endurance of the drone has a significant influence. Moreover, a few instances with up to 20 customers can be solved optimally within 1 hour of runtime.

In Schermer et al. [29] a variant of the TSP-D named the TSP-D *with robot stations* (TSP-D-RS) was proposed. This variant merges the sidekick-based with the station-based approaches (refer to Section 1). The compact MILP formulation presented in this paper also covers the truck-drone as a special case. Using a state-of-the-art solver, the proposed formulation can be used to solve these special cases in relatively short computation time. Indeed, problem instances with ten customers can be solved within 20 seconds and instances with up to 15 customers in less than 15 minutes.

Even though most assumptions are commonly accepted, the existing literature is in fact quite divergent[1]: several further subtle nuances differentiate the various problem statements from one another, making a comparison only valid to a limited extent. Apart from the problem instances and concrete choice of parameters (e.g., the drone's velocity), details of the problem statements differ in whether *round trips* are permitted, if vertices can be *revisited* by the truck, and whether both vehicles operate on the same type of *network*. Moreover, there is no general consensus on if an *overhead time* is incurred for launching and recovering the drone and how such penalties should be applied concretely. Furthermore, there is no agreement on if a drone is permitted to wait at a customer after making a delivery, thus conserving energy before the return flight, or if, instead, it must hover while waiting for the truck to pick it up again. In any case, there is no generally recognized way of specifying a drone's *endurance* precisely.

As will be unveiled in much more detail in the following sections, in this paper we provide three new formulations; in particular, one of them can be used to design a branch-and-cut scheme for solving the TSP-D effectively. Indeed, we can optimally solve instances with $n = 9$ or $n = 10$ customers in just a few seconds, instances with $n = 14$ customers within around 5 minutes, and several instances with $n = 19$ or $n = 20$ customers within an hour.

2.2 | Methodology-based literature

Several further works exist that either study variants of the problem (multiple trucks, drones, or different objective functions) or are primarily concerned with heuristics. In what follows, due to the reasons listed at the beginning of this section, we just mention them briefly.

[1]As an example, besides this work, there are four different problem statements, all of which are called "TSP-D" by the respective authors (refer to [1,11,15,22]).

In his thesis, Ponza [24] discusses the possibilities of solving the FSTSP heuristically using metaheuristics such as *simulated annealing* (SA) or *ant colony optimization*. Of these possible algorithms, SA is implemented and evaluated. For small-scale instances with at most 10 customers, on which optimal solutions can be obtained through a MILP formulation (that follows the one proposed by Murray and Chu [20]), the SA algorithm can obtain optimal solutions in most cases. After the investigation of these small instances, a more detailed parameter study is performed on larger instances with up to 200 customers.

In [6], de Freitas and Penna propose a *hybrid general variable neighborhood search* (VNS) for solving the FSTSP and TSP-D (the authors present a similar approach in [5]). In this algorithm, the first step consists in determining the optimal tour to the TSP. Next, a greedy improvement procedure is proposed that takes customers from the truck's route and assigns them to drone sorties. Finally, VNS serves as an in-depth improvement procedure. In particular, the authors propose several problem-specific neighborhood structures. The authors provide a detailed numerical study and show the effectiveness of their approach by testing on the problem instances proposed in [1,20,24].

The work by Ha et al. [15] studies a variant of the TSP-D with a minimal cost (as opposed to time) objective. The authors provide a MILP formulation that is based on the one provided in [20]. However, the primary contribution of the work is the design of an effective *greedy randomized adaptive search procedure* (GRASP) that can be used to insert drone operations into an existing TSP solution as well as some local search operators for refining the solution quality afterwards. In a subsequent work, the same authors investigate the performance of a *hybrid genetic algorithm* as an alternative to their GRASP [16]. The authors show that this novel algorithm outperforms existing approaches in terms of solution quality on instances with up to 100 customers.

In a similar manner, Sacramento et al. [26] study a variant of the TSP-D that involves multiple vehicles from a cost-per-mile-oriented perspective. For this purpose, they extend the MILP formulation that was proposed in [20]. In particular, they propose an *adaptive large neighborhood search* (ALNS) procedure that is capable of solving large-scale instances with up to 250 customers.

Following [23,30], Schermer et al. [27] investigate a variant of the TSP-D that features multiple vehicles and drones per vehicle named the *vehicle routing problem with drones* (VRPD). A MILP formulation and some effective VIEQs are introduced. However, the primary contribution is the design of a *Matheuristic* that embeds the *drone assignment and scheduling problem*. This heuristic is evaluated on instances with up to 100 customers. Related to the this work, Schermer et al. [28] also study a variant of the VRPD where *en route* operations are permitted, that is, the possibility of launching or recovering drones at some discrete points on each arc (see also [18]). Even though a MILP formulation is introduced, which can be used to address small-scale instances, the primary contribution is the design of a heuristic based on the concept of *variable neighborhood search* (VNS).

3 | THE TRAVELING SALESMAN PROBLEM WITH DRONE

In this section, we provide a formal specification of the TSP-D. We begin in Section 3.1 with a concise problem description. Afterwards, in Sections 3.2 and 3.3, we provide two compact MILP formulations of the problem.

3.1 | Problem definition

Suppose that there is a set of customers, each of whom must be served exactly once. In the TSP-D, we assume that there is a single truck that is equipped with a single drone for performing the deliveries. Both vehicles are initially located at a central depot and we look for a feasible route of the truck and admissible use of the drone such that all customers are served and both vehicles have returned to the depot as quickly as possible, that is, minimal makespan is achieved. Furthermore, consistent with most of the previous assumptions in the literature, we accept the following limitations in regard to the nature of the drone [1,20,30]:

- When launched from the truck, the drone can serve exactly one customer before it needs to return.
- We assume that the drone's *flight time* is limited by $\mathcal{E}$ *time units* per sortie. Furthermore, after returning to the truck, the battery of the drone is recharged (or swapped) instantaneously.
- Launching and recovering the drone might take $\bar{t}_l$ and $\bar{t}_r$ time units, respectively.
- It is only admissible to operate the drone at vertices (for a relaxation, refer to, e.g., [18,28]).

In the TSP-D, the locations of the depot and the customers are defined by a complete graph $\mathcal{G}(V, A)$. In this graph, V is the set of vertices and A the set of arcs. The vertex set V contains n vertices associated with the customers, labeled $V_N = \{1, \dots, n\} \subset V$ and two extra vertices 0 and $n+1$ that represent the depot location at the start and end of the tour, respectively. To sum up, $V = \{0, 1, \dots, n, n+1\}$, where $0 \equiv n+1$. We define $V_L = V \backslash \{n+1\}$ and $V_R = V \backslash \{0\}$.

A *drone-sortie* is characterized by a triple (i, w, j) as follows: the drone is launched from the truck at $i \in V_L$, performs a delivery to $w \in V_N$, and is retrieved at vertex $j \in V$ ($i \neq w$, $w \neq j$). If $j = i$, that is, when the sortie is (i, w, i), we call it a *round trip*; otherwise, if $j \in V_R$, $i \neq j$, we call it a *multileg* sortie. Finally, by the parameters d_{ij} and $\overline{d}_{ij}$ we define the distance required to travel from i to j by truck and drone, respectively. In a similar manner, we use t_{ij} and $\bar{t}_{ij}$ to define the time that is spent when traveling from i to j by either vehicle.

3.2 | A MILP formulation using a disjoint representation of multilegs

For the MILP formulation of the TSP-D, we use the binary decision variables according to Table 1 and the continuous variables according to Table 2. Using these variables, we can formulate the TSP-D as in (1) to (16), where (1) is the objective function and the constraints are given by (2) to (16). In the following, we refer to this model as *MILP-A* and it is inspired by the one presented in [29].

$$\min \quad s_{n+1}, \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in V_R} x_{0j} = \sum_{i \in V_L} x_{i,n+1} = 1, \tag{2}$$

$$\sum_{\substack{i \in V_L,\\ i \neq h}} x_{ih} = \sum_{\substack{j \in V_R,\\ h \neq j}} x_{hj} \quad : \quad \forall h \in V_N, \tag{3}$$

$$\sum_{\substack{i \in V_L,\\ i \neq w}} y_{iw} = \sum_{\substack{j \in V_R,\\ w \neq j}} \widetilde{y}_{wj} \quad : \quad \forall w \in V_N, \tag{4}$$

$$\sum_{\substack{i \in V_L,\\ i \neq w}} (x_{iw} + y_{iw} + z_{iw}) = 1 \quad : \quad \forall w \in V_N, \tag{5}$$

$$M(x_{ij} - 1) + s_i + t_{ij} \leq a_j \quad : \quad \forall i \in V_L,\ j \in V_R,\ i \neq j, \tag{6}$$

$$M(y_{iw} - 1) + s_i + \bar{t}_{iw} \leq a_w \quad : \quad \forall i \in V_L,\ w \in V_N,\ i \neq w, \tag{7}$$

$$M(\widetilde{y}_{wj} - 1) + s_w + \bar{t}_{wj} \leq a_j \quad : \quad \forall w \in V_N,\ j \in V_R,\ w \neq j, \tag{8}$$

$$a_i + \bar{t}_r \sum_{\substack{w \in V_N,\\ w \neq i}} \widetilde{y}_{wi} + \bar{t}_l \sum_{\substack{w \in V_N,\\ i \neq w}} y_{iw} + \sum_{\substack{w \in V_N,\\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) z_{iw} \leq s_i \quad : \quad \forall i \in V, \tag{9}$$

$$\sum_{\substack{i \in V_L,\\ i \neq w}} \bar{t}_{iw} y_{iw} + \sum_{\substack{j \in V_R,\\ w \neq j}} \bar{t}_{wj} \widetilde{y}_{wj} \leq \mathcal{E} \quad : \quad \forall w \in V_N, \tag{10}$$

$$\sum_{\substack{i \in V_L,\\ i \neq w}} (\bar{t}_{iw} + \bar{t}_{wi}) z_{iw} \leq \mathcal{E} \quad : \quad \forall w \in V_N, \tag{11}$$

$$(1 - x_{ij}) + v_i - \sum_{\substack{w \in V_N,\\ w \neq i,\\ w \neq j}} y_{iw} + \sum_{\substack{w \in V_N,\\ w \neq i,\\ w \neq j}} \widetilde{y}_{wj} \geq v_j \quad : \quad \forall i \in V_L,\ j \in V_R,\ i \neq j, \tag{12}$$

$$\sum_{\substack{i \in V_L,\\ i \neq j}} x_{ij} \geq v_j \quad : \quad \forall j \in V_R, \tag{13}$$

$$\sum_{\substack{w \in V_N,\\ i \neq w}} y_{iw} \leq v_i \quad : \quad \forall i \in V_L, \tag{14}$$

Table 1 Binary decision variables used in the TSP-D formulation

Variable	Explanation
$x_{ij} \in \{0,1\}$ $\forall i \in V_L, j \in V_R, i \neq j$	Equal to 1, iff arc (i, j) is used by the truck
$y_{iw} \in \{0,1\}$ $\forall i \in V_L, w \in V_N, i \neq w$	Equal to 1, iff arc (i, w) is the start of a multileg sortie
$\widetilde{y}_{wj} \in \{0,1\}$ $\forall w \in V_N, j \in V_R, i \neq w$	Equal to 1, iff arc (w, j) is the end of a multileg sortie
$z_{iw} \in \{0,1\}$ $\forall i \in V_L, w \in V_N, i \neq w$	Equal to 1, iff the drone performs a round trip (i, w, i)
$v_i \in \{0,1\}$ $\forall i \in V$	Equal to 1, iff the drone is available for a sortie at i

Table 2 Continuous decision variables used in the TSP-D formulation

Variable	Explanation
$a_i \in \mathbb{R}^{\geq 0}$ $\forall i \in V$	The time at which i is reached by either truck or drone. If the drone is retrieved at i, this marks the lower bound at which the last vehicle (truck or drone) has arrived
$s_i \in \mathbb{R}^{\geq 0}$ $\forall i \in V$	The earliest possible departure time from i

$$\sum_{\substack{w \in V_N, \\ w \neq j}} \widetilde{y}_{wj} \leq v_j \quad : \quad \forall j \in V_R, \tag{15}$$

$$\sum_{\substack{w \in V_N, \\ i \neq w}} z_{iw} \leq v_i \quad : \quad \forall i \in V_L. \tag{16}$$

In the TSP-D, the truck and the drone should serve all customers in shortest possible time and then return to the depot, which is stated in (1). Constraints (2) and (3) guarantee that the flow of the truck is conserved. Moreover, constraints (4) sustain the flow of the drone during each multileg. As is stated through constraints (5), every customer must be served exactly once.[2]

The temporal constraints are defined through (6) to (9). More precisely, (6) as well as (7) and (8) define the transitions of the truck and drone, respectively. Note that any subtour would contradict constraints (6) to (8). Moreover, constraints (9) account for the overhead times as well as the time spent performing round trips for every vertex.

The endurance constraints are given by (10) for multileg and (11) for round trip sorties and incorporate the time during which the drone is in flight. Constraints (12) and (13) account for the availability of the drone at each vertex. In particular, if an arc (i, j) is used by the truck, constraints (12) provide an upper bound on the availability of the drone at j, that is, v_j. If the drone is available at i, that is, $v_i = 1$, the upper bound $v_j \leq 1$ will only hold if either the drone is not launched at i or it is immediately recovered at j; otherwise, v_j will be bound to zero. In contrast, if the drone is unavailable at i, that is, $v_i = 0$, then there must be a recovery at j (from a sortie that began before i) for the drone to be available again. Moreover, constraints (13) are logical constraints: the drone can only be available at vertices that are visited by the truck. Finally, with these limits on the availability of the drone in place, constraints (14) to (16) guarantee that the drone can only be launched from and recovered at vertices where it is also available.

Even though MILP (1) to (16) formulates the TSP-D, some effective valid inequalities exist that significantly strengthen the linear relaxation (refer to [7,27,29]). For the purpose of this work, we integrate them into *MILP-A* as follows:

$$\sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} t_{ij} x_{ij} + \sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} ((\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) z_{iw} + \bar{t}_l y_{iw}) + \sum_{w \in V_N} \sum_{\substack{j \in V_R, \\ w \neq j}} \bar{t}_r \widetilde{y}_{wj} \leq s_{n+1}, \tag{17}$$

$$\sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} ((\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) z_{iw} + (\bar{t}_l + \bar{t}_{iw}) y_{iw}) + \sum_{w \in V_N} \sum_{\substack{j \in V_R, \\ w \neq j}} (\bar{t}_{wj} + \bar{t}_r) \widetilde{y}_{wj} \leq s_{n+1}. \tag{18}$$

Here, constraints (17) and (18) account for the time spent traveling on arcs by the truck and drone, respectively. Moreover, these constraints incorporate the time where the truck and drone perform their actions in a *synchronized* manner, that is, when the drone performs round trips while the truck remains stationary (waiting for the drone to return) and whenever the overhead times $\bar{t}_l$ and $\bar{t}_r$ are applied.

[2]Without loss of generality, we assume that every customer is eligible to be served by the drone. If this does not apply, one can simply bind the respective variables y_{iw}, y_{wj}, and z_{iw} to zero for each customer $w \in V_N \setminus V_D$, where $V_D \subseteq V_N$ is the subset of customers that are eligible to be served by drone.

3.3 | A MILP formulation using a joint representation of multilegs

In *MILP-A*, we use the variables y_{iw} and $\widetilde{y}_{wj}$ (according to Table 1) to differentiate between drone arcs that are part of outbound and inbound flights (each part of a multileg). In principle, instead of using two disjoint sets, it is also possible to use a single set of variables that simply state if an arc is used at all during a multileg flight. Hence, a smaller number of columns is required for formulating the problem. To the best of our knowledge, at this point, there are no models using a two-index formulation for the TSP-D (or related problems) that investigated this possibility.

To this end, we might use the binary decision variables x_{ij}, z_{iw}, and v_i as well as the continuous decision variables a_i and s_i according to Tables 1 and 2, respectively. Moreover, we introduce the additional binary decision variables according to Table 3. Then, the alternative MILP formulation of the TSP-D is given by (1) to (3), (6), (11), (13), (16), (19) to (31) and we denote this formulation by *MILP-B*.

$$\sum_{\substack{i \in V_L, \\ i \neq w}} (x_{iw} + (y_{iw} + z_{iw})) - r_w = 1 \quad : \quad \forall w \in V_N, \tag{19}$$

$$\sum_{\substack{i \in V_L, \\ i \neq w}} y_{iw} \leq \sum_{\substack{j \in V_R, \\ w \neq j}} y_{wj} + v_w \quad : \quad \forall w \in V_N, \tag{20}$$

$$v_w + \sum_{\substack{i \in V_L, \\ i \neq w}} y_{iw} \geq \sum_{\substack{j \in V_R, \\ w \neq j}} y_{wj} \quad : \quad \forall w \in V_N, \tag{21}$$

$$\sum_{\substack{i \in V_L, \\ i \neq j}} (x_{ij} + y_{ij}) - 1 \leq r_j \quad : \quad \forall j \in V_R, \tag{22}$$

$$\sum_{\substack{i \in V_L, \\ i \neq j}} x_{ij} \geq r_j \quad : \quad \forall j \in V_R, \tag{23}$$

$$\sum_{\substack{i \in V_L, \\ i \neq j}} y_{ij} \geq r_j \quad : \quad \forall j \in V_R, \tag{24}$$

$$r_j \leq v_j \quad : \quad \forall j \in V_R, \tag{25}$$

$$M(y_{ij} - 1) + s_i + \bar{t}_{ij} \leq a_j \quad : \quad \forall i \in V_L,\ j \in V_R,\ i \neq j, \tag{26}$$

$$a_i + \bar{t}_r r_i + \bar{t}_l \left(v_i - 1 + \sum_{\substack{w \in V_N, \\ i \neq w}} y_{iw} \right) + \sum_{\substack{w \in V_N, \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) z_{iw} \leq s_i \ : \forall i \in V, \tag{27}$$

$$\sum_{\substack{i \in V_L, \\ i \neq w}} \bar{t}_{iw} y_{iw} + \sum_{\substack{j \in V_R, \\ w \neq j}} \bar{t}_{wj} y_{wj} \leq \mathcal{E} + M r_w \quad : \quad \forall w \in V_N, \tag{28}$$

$$2 \cdot (1 - x_{ij}) + v_i - \sum_{\substack{w \in V_N, \\ w \neq i, \\ w \neq j}} y_{iw} + r_j \geq v_j \quad : \quad \forall i \in V_L,\ j \in V_R,\ i \neq j, \tag{29}$$

$$y_{ij} \leq v_i + r_j \quad : \quad \forall i \in V_L,\ j \in V_R, \tag{30}$$

$$\sum_{\substack{j \in V_R, \\ i \neq j}} y_{ij} \leq 1 \quad : \quad \forall i \in V_L. \tag{31}$$

Through the objective function (1) and constraints (2) and (3), we still aim at minimizing the makespan while preserving the flow of the truck. Moreover, the updated model requires several new constraints that restrict the binary variables r_i. Constraints (19) state that every customer must be served exactly once. If the drone is recovered at a vertex j, that is, $r_j = 1$, there are

Table 3 Additional binary decision variables used in the joint drone arc formulation of the TSP-D

Variable	Explanation
$y_{ij} \in \{0,1\}$ $\forall i \in V_L, j \in V_R, i \neq j$	Equal to 1, iff arc (i,j) is part of a multileg sortie (either outbound or inbound)
$r_i \in \{0,1\}$ $\forall i \in V$	If the drone is recovered after a multileg sortie at i

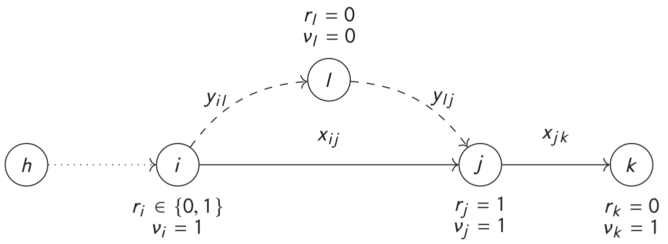

Figure 3 A part of a feasible TSP-D solution: the variables r_i are used to indicate if a drone is recovered. If the drone is available, that is, where $v_i = 1$, flow must not necessarily be conserved: in these situations, there can be an incoming (outgoing) drone arc without an outgoing (incoming) drone arc. Note that the case of $r_i = 1$, $v_i = 0$ does not exist $\forall i \in V$

two incoming arcs at j: one for the truck and drone, respectively. In this case, we subtract the binary variable r_j to achieve an equilibrium.

Constraints (20) and (21) provide conservation of flow for the drone arcs. More precisely, if the drone is not available at some vertex w, that is, $v_w = 0$, then the flow must be conserved and both constraints force the number of incoming (drone) arcs to be equal to the number of outgoing arcs. On the other hand, flow must not necessarily be conserved if the drone is available at w, that is, if $v_w = 1$ (see also Figure 3). Constraints (22) to (24) determine the value on the variables r_j: these constraints force the variables to take value 1 if there is exactly one incoming truck and drone arc and to value 0 otherwise. Moreover, constraints (25) are valid cuts that are logically implied by the remaining model. These constraints state that the drone can only be recovered if it also available at the same vertex.

In this model, the temporal constraints for the truck defined in (6) still hold. Moreover, it suffices to use constraints (26) and (27) for the transitions of the drone. The endurance constraints are given by (11) and (28) for round trip and multileg sorties, respectively. Note that, in this model, (28) now requires an additional Mr_w term, that is, the constraint only applies for two consecutive drone arcs that visit a customer (and not the truck) in between (see also Figure 3). Finally, constraints (13), (16), and (29) to (31) are used to determine the availability of the drone and to restrict operations accordingly. If the formulation presented in this section is used, before integrating the VIEQs (17) and (18) into *MILP-B*, slight adjustments are necessary. In particular, due to the nature of the variables y_{ij}, it is no longer possible to associate the overhead time $\bar{t}_l$ and $\bar{t}_r$ with outgoing and incoming drone arcs, respectively. However, during each multileg, the drone is launched and recovered exactly once. Hence, the total number of multileg arcs y_{ij} with value 1 will always be even and thus we can adapt these VIEQs as follows:

$$\sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} t_{ij} x_{ij} + \sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} ((\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) z_{iw}) + \sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} \frac{\bar{t}_l + \bar{t}_r}{2} y_{ij} \leq s_{n+1}, \tag{32}$$

$$\sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} ((\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) z_{iw}) + \sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} \left(\frac{\bar{t}_l + \bar{t}_r}{2} + \bar{t}_{ij} \right) y_{ij} \leq s_{n+1}. \tag{33}$$

3.4 | Discussion

As we will see in Section 5, only small-sized instances can be solved through *MILP-A* and *MILP-B*. In particular, several constraints involve large M-coefficients; hence, the linear relaxation is quite weak, which results in a slow convergence of the LB. Nevertheless, the proposed valid inequalities strengthen the linear relaxation significantly and, according to Lemma 1, these VIEQs are almost *tight*.

Lemma 1. *In an optimal solution, the slack value between left-hand-side of the VIEQs* (17) *and* (32) *and the makespan, that is,* s_{n+1}, *is equal to the total time that the truck remains stationary at vertices while waiting for the drone to arrive* (*from a multileg sortie*).

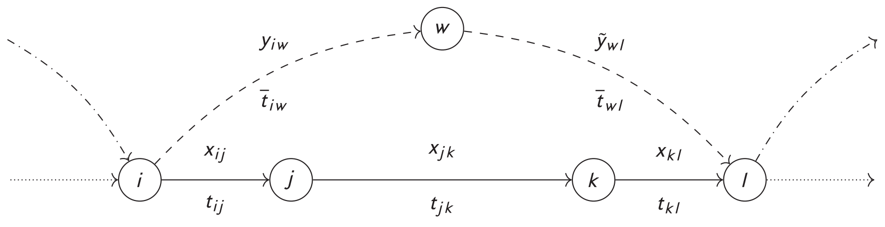

Figure 4 An example of an operation $o = \{x_{ij}, x_{jk}, x_{kl}, y_{iw}, \tilde{y}_{wl}\}$ with a cardinality of $|o| = 5$ that ends at $\delta(o) = l \in V_R$. The directed path P is defined by the solid arcs

Proof. The total operating time of the truck can be decomposed into three components as follows:

1. First, the *travel time*, that is, the time spent moving on arcs.
2. Second, the *synchronized waiting time*, that is, the time shared by the truck and drone in a coordinated manner while performing interdependent actions:
   - during launch and retrieval of the drone, that is, whenever the overhead times are applied,
   - whenever the truck is waiting for a drone to return from a round trip.

- Finally, the *nonsynchronized waiting time*, that is, the time spent at each vertex by the truck while waiting for the drone to arrive (from a multileg).

Based on the problem definition in Section 3, no other mode of operation does exist for the truck. Note that the VIEQs (17) and (32) account for the travel time and the synchronized waiting time. Therefore, the slack must be equal to the nonsynchronized waiting time. This concludes the proof of Lemma 1. ■

It is also worth to mention that, based on VIEQs (18) and (33), a similar lemma might be proposed for the drone. However, compared to Lemma 1, one further mode of operation exists for which these VIEQs do not account for: the time during which the drone is carried along an arc by the truck.

4 | A BRANCH-AND-CUT APPROACH FOR THE TSP-D

The MILP formulations presented in Section 3 might be solved by any standard MILP solver, for example, Gurobi Optimizer, which uses a branch-and-cut approach [14]. However, in order to gain more efficiency, we introduce a third MILP formulation that permits a customized design of the branch-and-cut algorithm. This proposal is motivated by three main observations:

1. In optimal solutions, typically, the travel time and synchronized waiting time have the largest contribution to the makespan and, due to Lemma 1, they can be approximated well through the linear constraints (17) and (32).
2. The temporal constraints in *MILP-A* and *MILP-B*, presented in Section 3, introduce several M-coefficients (as is the case in related models, e.g., [7,8,15,20]). Consequently, the quality of the linear relaxation is quite weak and the convergence of the LB is slow. However, these formulations are *compact*, that is, of polynomial size. In fact, in our formulations, only $\mathcal{O}(|V|^2)$ variables and constraints are required for formulating the problem.
3. In the set-covering based ILP formulation proposed by Agatz et al. [1], the so-called *operations* are used as the essential building block. Operations already explicitly incorporate the *nonsynchronized waiting times*; therefore, no big-M constraints are necessary. However, there is an exponential number of operations for a given instance; hence, a static formulation does not scale very well to a branch-and-cut framework.

Based on these observations, we introduce a MILP formulation that attempts to leverage the strengths of the MILP and ILP approaches listed above through Lemma 1. Before we present the model, we adapt the concept of an *operation o* for our purpose (based on [1]). Let an operation o contain a feasible sequence (w.r.t. Section 3) of one or more truck-arcs that we call a *directed path P*. Each operation o contains exactly one *multileg* drone sortie that starts at the beginning P, targets a customer (not contained in P), and concludes at the end of P. We introduce the notation $\delta(o) \in V_R$ to refer to the vertex at the end of the operation o (the path P). Finally, we denote the cardinality of an operation by $|o|$ and define it as the total number of directed arcs that are contained. Figure 4 illustrates an example of this concept. Let O denote the set of all feasible operations and take note that this set grows exponentially with the instance size, that is, with $|V|$.

In this section, we introduce a MILP formulation of the TSP-D that features an exponential number of *constraints*. As a result, we propose these constraints to be separated dynamically, that is, through a branch-and-cut framework. For introducing

Table 4 Additional variables used for the formulation *MILP-BC*

Variable	Explanation
$\tau \in \mathbb{R}^{\geq 0}$	The least upper bound on the *travel time* and *nonsynchronized waiting time* for the truck and drone
$\omega \in \mathbb{R}^{\geq 0}$	The total *synchronized waiting time* shared by the truck and drone
$\underset{\forall j \in V_R}{w_j^t \in \mathbb{R}^{\geq 0}}$	The nonsynchronized waiting time of the truck at vertex j (while waiting for the drone to arrive)
$\underset{\forall j \in V_R}{w_j^d \in \mathbb{R}^{\geq 0}}$	The nonsynchronized waiting time of the drone at vertex j (while waiting for the truck to arrive)
$\underset{\forall i \in V}{u_i \in \mathbb{Z}^+}$	Sets the position of vertex i in the delivery sequence

this formulation, we use the decision variables according to Tables 1 and 4. Using this notation, the MILP generally follows *MILP-A* and is given by (2) to (5), (10) to (16), (34) to (39), and the *Miller-Tucker-Zemlin* (MTZ) constraints (42) to (46). Due to the branch-and-cut nature, we call this formulation *MILP-BC*.

$$\min \quad \tau + \omega, \tag{34}$$

$$\text{s.t.} \quad \sum_{i \in V_L} \sum_{\substack{j \in V_R, \\ i \neq j}} t_{ij} x_{ij} + \sum_{r \in V_R} w_r^t = \tau, \tag{35}$$

$$\sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} t_{iw} y_{iw} + \sum_{w \in V_N} \sum_{\substack{j \in V_R, \\ w \neq j}} t_{wj} \widetilde{y}_{wj} + \sum_{r \in V_R} w_r^d \leq \tau, \tag{36}$$

$$\sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} (\bar{t}_l + \bar{t}_{iw} + \bar{t}_{wi} + \bar{t}_r) z_{iw} + \bar{t}_l \sum_{i \in V_L} \sum_{\substack{w \in V_N, \\ i \neq w}} y_{iw} + \bar{t}_r \sum_{w \in V_N} \sum_{\substack{j \in V_R, \\ w \neq j}} \widetilde{y}_{wj} = \omega. \tag{37}$$

Let us explain the model in more detail. Generally, this formulation closely follows *MILP-A*. In fact, we only replace the objective function and drop all of the temporal constraints (and thus, M-coefficients). Hence, in *MILP-BC*, the objective function is given by (34) and consists of two terms: τ and ω (note that a similar objective is used in [7,8]). The value τ has two bounds which are defined through constraints (35) and (36) for the truck and drone, respectively. These constraints account for the time spent traveling on arcs as well as the nonsynchronized waiting times w_r^t and w_r^d at each vertex $r \in V_R$. Note that the bound is binding in case of constraint (35) while (36) only provides a LB on τ (refer to Lemma 1 and the comments thereafter). The total synchronized waiting time ω is defined through Equation (37). This constraint accounts for the time spent performing round trips and the total overhead time.

$$\phi(o) \left(1 + \sum_{x_{ij} \in o} x_{ij} + \sum_{y_{iw} \in o} y_{iw} + \sum_{\widetilde{y}_{wj} \in o} \widetilde{y}_{wj} - |o| \right) \leq w_{\delta(o)}^t \quad : \quad \forall o \in O, \tag{38}$$

$$\theta(o) \left(1 + \sum_{x_{ij} \in o} x_{ij} + \sum_{y_{iw} \in o} y_{iw} + \sum_{\widetilde{y}_{wj} \in o} \widetilde{y}_{wj} - |o| \right) \leq w_{\delta(o)}^d \quad : \quad \forall o \in O. \tag{39}$$

For each operation $o \in O$, constraints (38) and (39) provide LBs on the nonsynchronized waiting times $w_{\delta(o)}^t$ for the truck and $w_{\delta(o)}^d$ for the drone. To explain these constraints, let us assume that an operation o is currently part of the solution, that is, the parentheses in (38) and (39) evaluate to 1; then, for the given operation o, the nonsynchronized waiting times at the vertex $\delta(o) \in V_R$ are defined by the functions $\phi(o)$ and $\theta(o)$ for the truck and drone, respectively. These functions determine the nonsynchronized waiting times at the vertex $\delta(o) \in V_R$ as follows:

- If one vehicle (either the truck or the drone) arrives at $\delta(o)$ at the same time as or after the other one, then the nonsynchronized waiting time (either $\phi(o)$ or $\theta(o)$) is equal to 0 as it does not have to wait for synchronization.
- However, if one vehicle (either the truck or the drone) arrives at $\delta(o)$ before the other one, then the nonsynchronized waiting time (either $\phi(o)$ or $\theta(o)$) is equal to the difference in arrival time, that is, the time spent waiting.

As a result, the functions $\phi(o)$ and $\theta(o)$ are embedded within the scheme shown in Algorithm 1. Indeed, using a state-of-the-art solver, this algorithm can be applied through a *callback* whenever a possibly new MIP feasible incumbent solution is identified. In more detail, the algorithm works as follows for each operation o that is contained in the current solution:

1. First, the travel times of the truck and drone are calculated. Here, we do not account for the overhead times, as they are already included in the term ω in the objective function (see objective (34) and constraint (37)).
2. Afterwards, depending on which vehicle arrives first, either cut (38) or (39) is added using the difference in travel time, that is, the nonsynchronized waiting time at $\delta(o)$ of either the truck or the drone, given that the operation o is part of the solution.

Algorithm 1. Constraint generation scheme (applied at each MIP incumbent node)

```
1: for each operation $o \in O$ contained in the current solution do
2:     Calculate $t_t = \sum_{x_{ij} \in o} t_{ij} x_{ij}$ and $\bar{t}_d = \sum_{y_{iw} \in o} \bar{t}_{iw} y_{iw} + \sum_{\tilde{y}_{wj} \in o} \bar{t}_{wj} \tilde{y}_{wj}$
3:     if $t_t < \bar{t}_d$ then                ▷ In operation $o$, the truck arrives before the drone.
4:         $\phi(o) = (\bar{t}_d - t_t)$
5:         Add cut (38) for operation $o$
6:     else if $t_t > \bar{t}_d$ then           ▷ In operation $o$, the drone arrives before the truck.
7:         $\theta(o) = (t_t - \bar{t}_d)$
8:         Add cut (39) for operation $o$
9:     end if
10: end for
```

Using this algorithm, the corresponding constraints, that is, (38) and (39), are added only as necessary or not before needed to the model. This scheme is guaranteed to converge towards the optimal solution because of Lemmas 1 and 2 as well as Proposition 1, described in what follows.

Lemma 2. *An operation $o \in O$ is never a subset of any other operation $o' \in O$, $o \neq o'$.*

Proof. By contradiction, assume that $\exists o' \in O$ such that $o \subset o'$ and $o \neq o'$ (the case of $o' \subset o$ can be treated analogously; hence, we skip it). We differentiate the following cases:

1. either o' contains at least one more drone sortie,
2. or o' contains at least one more arc that is used by the vehicle.

The first case contradicts the definition of an operation as at most one (multileg) drone sortie can be contained in each operation. For the second case, assume that we were to add an additional arc x_{ij} to the path P. For the path to remain feasible, the only possible location to append one further arc is at the end or the beginning of P. However, this would also contradict the definition of an operation as the drone would no longer be launched (retrieved) at the start (end) of the path. This concludes the proof of Lemma 2. ∎

Proposition 1. *The constraints* (38) *and* (39) *are optimality cuts, that is, they are*:

1. *binding if the operation $o \in O$ is contained in the current solution,*
2. *feasible for all other $o' \in O$, $o' \neq o$, otherwise.*

Proof. The first statement follows directly from the definition of an operation at the beginning of Section 4 and the fact that constraints (38) and (39) are *indicator* constraints, that is, they become active if and only if o is part of the solution. In this case, that is, if the operation o is part of the solution, constraints (38) and (39) provide valid LBs on w_r^t and w_r^d for the truck and drone, respectively.

The second statement is true because of Lemma 2. In fact, since o is never a subset of any other o', the cut is nonbinding and feasible for all other operations. This completes the proof of Proposition 1. ∎

As an example, let us consider the operation $o \in O$ shown in Figure 4. Here, the operation is given as $o = \{x_{ij}, x_{jk}, x_{kl}, y_{iw}, \tilde{y}_{wj}\}$ with a cardinality of $|o| = 5$. In this example, we assume that the arrival time of the truck is $t_t = t_{ij} + t_{jk} + t_{kl}$ and the arrival time of the drone is $\bar{t}_d = \bar{t}_{iw} + \bar{t}_{wj}$. As the overhead times $\bar{t}_l$ and $\bar{t}_r$ are shared by both vehicles and incorporated in the synchronized waiting time ω, we do not have to consider them here. If $t_t > \bar{t}_d$, then the drone has to wait for $\theta(o) = (t_t - \bar{t}_d)$ time units at $\delta(o) = l \in V_R$ and the following drone-cut can be added (refer to Algorithm 1):

$$(t_t - \bar{t}_d)(x_{ij} + x_{jk} + x_{kl} + y_{iw} + \tilde{y}_{wj} - 4) \leq w_l^d. \tag{40}$$

Otherwise, if $t_t < \bar{t}_d$, the truck has to wait for $\phi(o) = (\bar{t}_d - t_t)$ time units at l for the drone to arrive. Thus, we can add the following truck-cut:

$$(\bar{t}_d - t_t)(x_{ij} + x_{jk} + x_{kl} + y_{iw} + \widetilde{y}_{wj} - 4) \leq w_l^t. \tag{41}$$

Note that, in the (unlikely) event where $\bar{t}_d = t_t$, the arrival of the truck and drone is perfectly synchronous; hence, no cut must be added as neither vehicle has to wait for the other one.

Finally, the temporal constraints (6) to (9) that also prevent subtours are no longer used in the model *MILP-BC*. As a result, we adapt and use the family of MTZ constraints (refer to [19]) as follows:

$$u_0 = 0, \tag{42}$$

$$1 \leq u_i \leq n+1 \quad : \quad \forall i \in V_R, \tag{43}$$

$$u_i - u_j + 1 \leq (n+1)(1 - x_{ij}) \quad : \quad \forall i \in V_L,\ j \in V_R,\ i \neq j, \tag{44}$$

$$u_i - u_w + 1 \leq (n+1)(1 - y_{iw}) \quad : \quad \forall i \in V_L,\ w \in V_N,\ i \neq w, \tag{45}$$

$$u_w - u_j + 1 \leq (n+1)(1 - \widetilde{y}_{wj}) \quad : \quad \forall w \in V_N,\ j \in V_R,\ w \neq j. \tag{46}$$

This concludes the description of *MILP-BC*. However, in optimal solutions, many operations are of cardinality $|o| = 3$, that is, the truck does not serve any additional customer in between launch and recovery of the drone (see also Figure 7). Even though, the constraint generation scheme proposed in Algorithm 1 also accounts for such operations, it might be beneficial to introduce some additional VIEQs that match to these types of operations. Therefore, we propose constraints (47) and (48) as VIEQs to *MILP-BC* and call the resulting formulation *MILP-BC-V I*:

$$\sum_{\substack{w \in V_N, \\ i \neq w, \\ w \neq j}} (\bar{t}_{iw} y_{iw} + \bar{t}_{wj} \widetilde{y}_{wj}) - t_{ij} x_{ij} + M \cdot \left(x_{ij} + \sum_{\substack{w \in V_N, \\ i \neq w, \\ w \neq j}} (y_{iw} + \widetilde{y}_{wj}) - 3 \right) \leq w_j^t \quad : \quad \forall i \in V_L,\ j \in V_R, \tag{47}$$

$$t_{ij} x_{ij} - \sum_{\substack{w \in V_N, \\ i \neq w, \\ w \neq j}} (\bar{t}_{iw} y_{iw} - \bar{t}_{wj} \widetilde{y}_{wj}) + M \cdot \left(x_{ij} + \sum_{\substack{w \in V_N, \\ i \neq w, \\ w \neq j}} (y_{iw} + \widetilde{y}_{wj}) - 3 \right) \leq w_j^d \quad : \quad \forall i \in V_L,\ j \in V_R. \tag{48}$$

In particular, constraints (47) and (48) account for the waiting time of the truck and drone, respectively, at each vertex j, if a sortie o of cardinality $|o| = 3$ is performed (i.e., if the expression with coefficient M evaluates to 0). Note that, in these cases, a sufficiently large value of M is relatively small; in fact, $M = \arg\ \max_{i,j \in V} 2 \cdot \bar{t}_{ij}$ and $M = \arg \max_{i,\, j \in V} t_{ij}$ are sufficient for the value in (47) and (48), respectively.

Finally, as a general remark, note that the variables w_j^t and w_j^d might also prove useful when some constraints are imposed that might limit the maximum waiting time of the truck or drone or when it is of interest to associate a specific cost with waiting (as it is the case in, e.g., [15]).

5 | COMPUTATIONAL EXPERIMENTS AND THEIR NUMERICAL RESULTS

All experiments were performed on an Intel Xeon E5-2640v3 Processor with access to 8 GB of RAM. As MILP solver, we chose Gurobi Optimizer 9.0.0 [14] with a runtime limit of 1 hour. For the design of the branch-and-cut algorithm introduced in Section 4, a natural choice is to use the *callback* framework of the same state-of-the-art solver, that is, we apply the cut scheme proposed in Algorithm 1 at each new MIP incumbent node in the branch-and-cut tree. Through this process, we rely on an established external framework for primal heuristics, the generation of general MILP cuts, branching-decisions, and so forth (compare with, e.g., Fischetti et al. [12,13]). As we have outlined towards the end of Section 2, there are no commonly accepted problem assumptions. As a consequence, we selected three different types of benchmark instances to conduct our computational experiments. In Section 5.1, we begin with instances introduced by Bouman et al. [4] that were first used in [1]. Afterwards, in Section 5.2, we test on the instances proposed by Poikonen et al. [22]. Finally, in Section 5.3, we conclude our experiments using instances from Murray and Chu [20].

5.1 | Instances proposed by Bouman et al.

In this section, we describe our computational experiments on the instances proposed by Bouman et al. [4], which have been used in, for example, [1,3,22]. We begin in Section 5.1.1 by reporting the experimental setup. Afterwards, the numerical results are discussed in Section 5.1.2.

5.1.1 | Experimental setup

Bouman et al. [4] provide a wide range of instances with different numbers of vertices. For our experiments, we use instances with $n \in \{9, 19\}$ customers (i.e., $n+1 \in \{10, 20\}$ vertices), where the coordinates were drawn uniformly and independently from $\{0, \ldots, 100\}$. It is assumed that both vehicles follow the Euclidean distance metric and the drone's endurance is limited by flight distance. In particular, the velocity of the drone is $\alpha \in \{1, 2, 3\}$ times that of the truck (which moves at a velocity of one distance unit per time unit). Moreover, the overhead times are assumed to be negligible, that is, $\overline{t}_l = \overline{t}_r = 0$. Regarding the endurance, let $\mathcal{E} = \arg \ \max_{i,j \in V} \overline{d}_{ij}$ be the edge with maximum length in the graph G. Using this value of $\mathcal{E}$, for each instance, the drone's range of operation is restricted by a parameter $R \in \{20\%, 40\%, 60\%, 100\%, 150\%, 200\%\}$ that is multiplied by $\mathcal{E}$. In order to incorporate this parameter into our models, for *MILP-A* and *MILP-BC*, constraints (10) are replaced as follows:

$$\sum_{\substack{i \in V_L, \\ i \neq w}} \overline{d}_{iw} y_{iw} + \sum_{\substack{j \in V_R, \\ w \neq j}} \overline{d}_{wj} \tilde{y}_{wj} \leq R \cdot \mathcal{E}, \qquad \forall w \in V_N. \tag{49}$$

Moreover, for *MILP-B*, we adjust constraints (28) as follows:

$$\sum_{\substack{i \in V_L, \\ i \neq w}} \overline{d}_{iw} y_{iw} + \sum_{\substack{j \in V_R, \\ w \neq j}} \overline{d}_{wj} y_{wj} \leq R \cdot \mathcal{E} + M r_w, \qquad \forall w \in V_N. \tag{50}$$

Finally, for all formulations, the endurance for round trips defined in (11) is instead restricted as follows:

$$\sum_{\substack{i \in V_L, \\ i \neq w}} (\overline{d}_{iw} + \overline{d}_{wi}) z_{iw} \leq R \cdot \mathcal{E}, \qquad \forall w \in V_N. \tag{51}$$

5.1.2 | Numerical results

The resulting set of 360 instances was solved using formulations *MILP-A*, *MILP-B* (with their respective valid inequalities) as well as with *MILP-BC* and *MILP-BC-V I*, yielding a total of 1440 experiments. Table 5 provides detailed results on the performance of each approach. Based on the instance size n, the relative velocity α, and relative endurance R, this table shows the MIP gap and runtime (averaged over 10 instances) as well as the share of instances solved to optimality by each approach. Moreover, for *MILP-BC* and *MILP-BC-VI*, the average number of cuts of types (38) and (39), which are added during the search, are shown.

Based on the results presented in Table 5, we can make the following observations. The formulation *MILP-A* can be used to effectively solve instances with $n = 9$ customers, requiring no more than 2 seconds on average. In contrast, the alternative formulation *MILP-B* is noticeably worse in terms of average runtime. Beyond this size, that is, for $n = 19$, the performance of *MILP-A* is noticeably better: 16.1% of these instances can be solved optimally with an average remaining MIP gap of 16.1%. Overall, the branch-and-cut algorithm that utilizes *MILP-BC* shows promising performance. While *MILP-A* is on average slightly faster for $n = 9$ customers, the branch-and-cut approach shows a slightly superior performance in tackling instances with $n = 19$ customers. To be more precise, through *MILP-BC* we can solve a slightly larger share (22.2%) to proven optimality with an average remaining MIP gap of 16.2%. In contrast, *MILP-BC-VI* solves fewer instances to optimality (19.2%); however, with an improved MIP gap of 15.7%. A closer look at these two formulations reveals that the cuts of type (38) and (39), which are added during the search, are about half if *MILP-BC-VI* is used. Table 5 also displays the quality of the relative root relaxation $\%_r$ associated with each formulation. We provide this information by using the following formula (52) as a performance metric:

$$\%_r := \frac{\text{root relaxation objective value}}{\text{objective value of the best feasible solution after runtime (found through any formulation)}} \tag{52}$$

Overall, the quality of the root relaxation associated with *MILP-B* is noticeably worse. In fact, on average, the relative root relaxation of *MILP-A* is about 13% better which also helps to explain why *MILP-A* performs significantly better. When we look at the formulations *MILP-BC* and *MILP-BC-VI*, we notice that their relative root relaxations are just slightly better (about 1%) than those associated with *MILP-A*.

Table 5 Average MIP gap, average runtime (in seconds), the share of instances solved to optimality (opt.), and the average relative root relaxation $\%_r$ depending on the number of customers n, the relative velocity α, and the relative endurance R on the Bouman et al. [4] instance set (10 instances per row)

Instance			*MILP-A*				*MILP-B*				*MILP-BC*					*MILP-BC-VI*				
n	α	R (%)	Gap (%)	Time	Opt. (%)	$\%_r$	Gap (%)	Time	Opt. (%)	$\%_r$	Gap (%)	Time	Opt. (%)	$\%_r$	Cuts	Gap (%)	Time	Opt. (%)	$\%_r$	Cuts
9	1	20	0.0	0.2	100.0	71.6	0.0	0.2	100.0	69.7	0.0	0.2	100.0	72.8	4.5	0.0	0.2	100.0	72.9	4.5
		40	0.0	0.3	100.0	63.5	0.0	1.0	100.0	37.1	0.0	0.3	100.0	65.0	23.4	0.0	0.4	100.0	66.9	10.3
		60	0.0	1.6	100.0	47.1	0.0	3.7	100.0	33.7	0.0	1.7	100.0	47.7	38.2	0.0	2.7	100.0	48.1	21.6
		100	0.0	5.2	100.0	45.2	0.0	9.3	100.0	35.1	0.0	5.2	100.0	45.4	162.3	0.0	7.9	100.0	45.4	106.4
		150	0.0	6.4	100.0	47.1	0.0	7.7	100.0	36.6	0.0	7.3	100.0	47.4	531.8	0.0	10.8	100.0	47.4	365.4
		200	0.0	6.0	100.0	47.1	0.0	13.6	100.0	36.6	0.0	7.3	100.0	47.4	529.4	0.0	9.0	100.0	47.4	325.3
	2	20	0.0	0.2	100.0	71.5	0.0	0.2	100.0	69.4	0.0	0.1	100.0	72.7	2.3	0.0	0.1	100.0	72.7	1.7
		40	0.0	0.3	100.0	61.3	0.0	0.6	100.0	33.5	0.0	0.3	100.0	64.0	8.3	0.0	0.3	100.0	65.1	5.6
		60	0.0	0.7	100.0	43.2	0.0	2.0	100.0	26.6	0.0	0.9	100.0	44.0	21.6	0.0	1.7	100.0	44.7	14.2
		100	0.0	1.7	100.0	40.6	0.0	3.4	100.0	29.5	0.0	2.3	100.0	40.8	67.6	0.0	3.6	100.0	41.0	36.3
		150	0.0	3.1	100.0	42.3	0.0	3.9	100.0	31.7	0.0	3.8	100.0	42.6	152.4	0.0	4.7	100.0	42.8	69.3
		200	0.0	2.7	100.0	42.5	0.0	4.5	100.0	31.9	0.0	4.2	100.0	42.8	186.9	0.0	4.2	100.0	43.0	60.3
	3	20	0.0	0.2	100.0	71.1	0.0	0.2	100.0	69.0	0.0	0.1	100.0	72.6	3.3	0.0	0.2	100.0	72.6	3.3
		40	0.0	0.3	100.0	59.6	0.0	0.7	100.0	32.1	0.0	0.3	100.0	62.8	9.9	0.0	0.4	100.0	63.5	7.8
		60	0.0	0.7	100.0	40.5	0.0	1.4	100.0	22.6	0.0	0.9	100.0	41.8	20.7	0.0	1.2	100.0	42.6	16.3
		100	0.0	1.1	100.0	35.7	0.0	1.7	100.0	24.7	0.0	1.4	100.0	35.9	51.5	0.0	2.3	100.0	36.1	26.6
		150	0.0	2.1	100.0	38.7	0.0	2.4	100.0	28.6	0.0	2.9	100.0	39.2	85.4	0.0	3.1	100.0	39.3	47.5
		200	0.0	2.1	100.0	39.1	0.0	2.9	100.0	28.9	0.0	3.6	100.0	39.6	146.2	0.0	2.9	100.0	39.8	58.4
Avg. ($n=9$)			0.0	1.9	100.0	50.4	0.0	3.3	100.0	37.6	0.0	2.4	100.0	51.4	113.7	0.0	3.1	100.0	51.7	65.6
19	1	20	1.4	1527.9	60.0	60.7	1.9	1556.7	60.0	38.8	0.0	618.5	100.0	62.2	21.2	0.2	811.0	90.0	62.8	12.3
		40	18.0	3600.0	0.0	44.0	20.3	3600.0	0.0	31.7	12.1	3545.3	10.0	44.4	241.6	12.5	3600.0	0.0	44.3	131.5
		60	26.5	3600.0	0.0	43.7	30.9	3600.0	0.0	33.6	22.2	3600.0	0.0	43.8	618.3	22.7	3600.0	0.0	43.8	343.7
		100	23.9	3600.0	0.0	45.1	32.2	3600.0	0.0	35.0	23.6	3600.0	0.0	45.1	1448.1	23.7	3600.0	0.0	45.2	837.9
		150	22.8	3600.0	0.0	45.8	30.8	3600.0	0.0	35.6	23.3	3600.0	0.0	45.9	1472.5	22.3	3600.0	0.0	45.9	765.7
		200	24.3	3600.0	0.0	45.5	31.6	3600.0	0.0	35.4	23.8	3600.0	0.0	45.6	1505.2	23.4	3600.0	0.0	45.6	1013.3
	2	20	0.2	1079.2	90.0	60.2	1.0	1236.4	80.0	37.8	0.0	740.6	100.0	61.7	12.2	0.1	784.7	90.0	62.0	9.9
		40	9.4	3129.4	20.0	40.5	11.3	3306.3	10.0	24.6	5.7	2605.3	50.0	41.5	121.1	7.1	2920.9	30.0	41.5	51.9
		60	22.9	3600.0	0.0	38.8	27.1	3600.0	0.0	27.9	19.0	3600.0	0.0	38.9	248.3	18.1	3600.0	0.0	39.0	136.3
		100	24.2	3600.0	0.0	40.8	36.0	3600.0	0.0	30.4	22.8	3600.0	0.0	40.9	504.9	24.9	3600.0	0.0	40.9	203.5
		150	23.8	3600.0	0.0	40.9	38.2	3600.0	0.0	30.5	24.3	3600.0	0.0	41.0	705.1	23.3	3600.0	0.0	41.0	252.0
		200	24.4	3600.0	0.0	40.6	35.8	3600.0	0.0	30.3	24.5	3600.0	0.0	40.7	659.7	22.0	3600.0	0.0	40.7	249.4
	3	20	0.5	1107.0	90.0	58.6	1.1	1274.6	70.0	37.5	0.3	751.6	90.0	61.1	10.2	0.2	801.4	90.0	61.3	9.7
		40	6.6	3091.2	30.0	38.3	7.6	3141.3	30.0	20.7	5.2	2588.5	50.0	39.9	100.2	5.0	2934.8	50.0	40.1	58.9
		60	19.6	3600.0	0.0	34.8	24.3	3600.0	0.0	23.9	16.0	3600.0	0.0	34.9	206.4	14.7	3600.0	0.0	35.2	94.3
		100	25.3	3600.0	0.0	36.9	39.1	3600.0	0.0	26.9	24.2	3600.0	0.0	37.0	435.9	22.0	3600.0	0.0	37.0	135.2
		150	24.0	3600.0	0.0	37.0	40.6	3600.0	0.0	27.0	22.9	3600.0	0.0	37.1	533.8	20.6	3600.0	0.0	37.1	194.2
		200	23.9	3600.0	0.0	37.4	39.7	3600.0	0.0	27.2	21.0	3600.0	0.0	37.4	678.7	20.3	3600.0	0.0	37.5	160.1
($n=19$)			17.9	3151.9	16.1	43.9	25.0	3184.2	13.9	30.8	16.2	3002.8	22.2	44.4	529.1	15.7	3058.5	19.4	44.5	258.9

Note: For *MILP-BC* and *MILP-BC-VI*, this table shows also the average number of constraints (38) and (39) added as cuts.

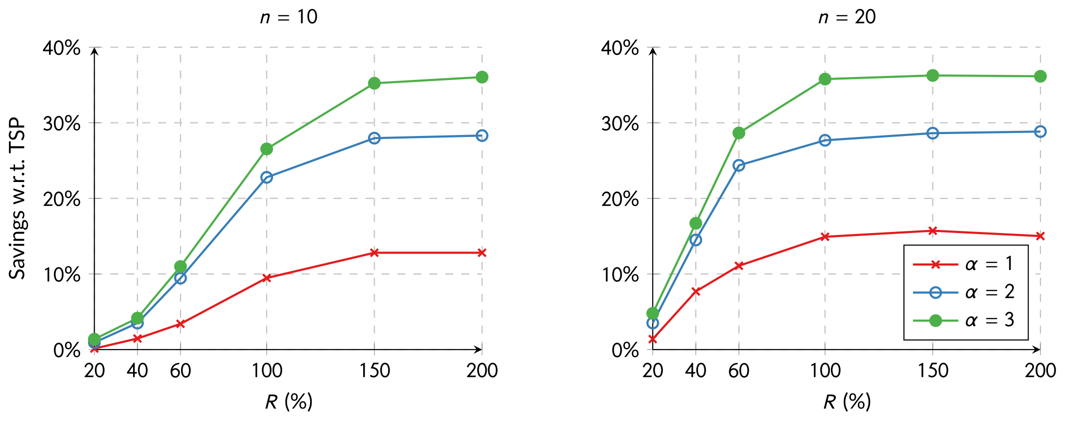

Figure 5 The savings w.r.t. the TSP solution depending on the drone's relative velocity α and its relative endurance R for the results obtained through *MILP-BC-VI* [Color figure can be viewed at wileyonlinelibrary.com]

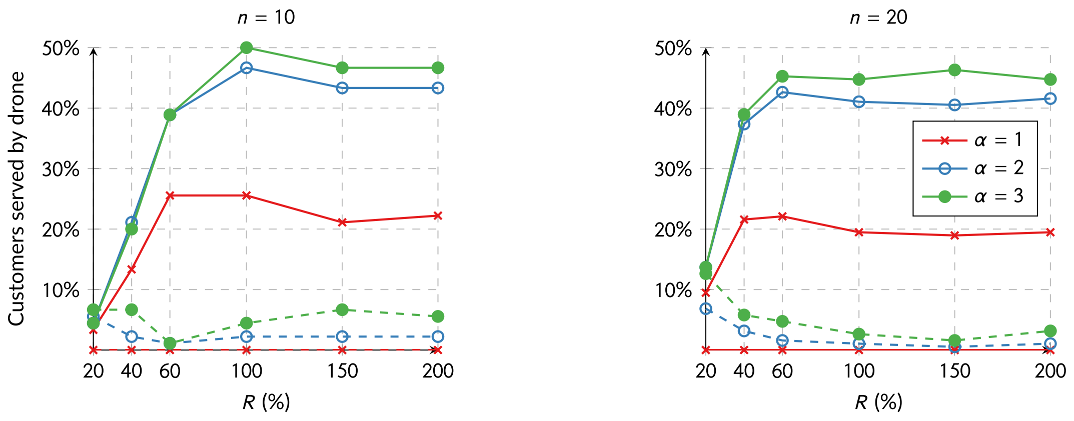

Figure 6 The share of customers served by the drone depending on its relative velocity α and its relative endurance R for the results obtained through *MILP-BC-VI*. Solid lines represent multileg sorties and dashed lines indicate round trips [Color figure can be viewed at wileyonlinelibrary.com]

In Figure 5, we provide an insight into the savings (w.r.t. the TSP solution) that are calculated as follows:

$$\Delta := 100\% - \frac{\text{Objective value of the feasible solution after runtime}}{\text{Optimal objective value of the TSP solution}} \qquad (53)$$

Moreover, Figure 6 highlights the share of customers served by the drone. These figures are plotted for the results obtained through *MILP-BC-VI*. Regarding the savings shown in Figure 5, we can make the following observations. As might be expected, the savings generally increase as the drone's velocity and endurance are increased. However, for a given value of α, no further gains can be obtained once a certain threshold of the endurance parameter R is reached. In any case, savings of about 35% w.r.t. the TSP solution are possible. This magnitude of savings is consistent with previous observations (see, e.g., [1,27]). In Figure 6, overall, the number of customers served by the drone is significant and grows as the relative velocity α of the drone is increased. What is interesting about these figures is that the shares of operations do not grow monotonic with α and R. In particular, we can observe that as these values change, round trips tend to be substituted with multileg sorties.

Finally, let us investigate the makespan components according to the objective function (34). For this purpose, we use the results obtained through the formulation *MILP-BC-VI*. Table 6 provides a breakdown of the makespan (recall that $\bar{t}_l = \bar{t}_r = 0$) and is best viewed in conjunction with Figure 6.

The time spent traveling on arcs has the largest contribution with more than 97% on average. Moreover, the time spent waiting by the truck for the drone to join from a multileg is less than 1.5% on average (similar observations were made in [11]). However, there is a strong dependence on the relative velocity α as a faster drone (and a drone with an increased relative endurance R) typically also causes the (nonsynchronized) waiting time to increase.

5.2 | Instances proposed by Poikonen et al.

If we assume that a drone can make an autonomous delivery to a customer, it might need to land at the customer's location. After landing, if we ask that the drone can delay its departure for a sufficient amount of time, only the time in-flight or the distance covered (given that the velocity is constant) are decision-relevant (for a detailed discussion, refer to [27]). This assumption is present in several papers (see, e.g., [1,20]) and was investigated in Section 5.1. However, in some situations, for example, for

Table 6 The proportion of the time spent moving, waiting for the drone to arrive from a multileg (nonsynchronized waiting time), and the time spent waiting while the drone is doing round trips

Instance		$n = 9$			$n = 19$		
α	R (%)	Move (%)	Wait (%)	Round trip (%)	Move (%)	Wait (%)	Round trip (%)
1	20	99.7	0.3	0.0	99.6	0.4	0.0
	40	99.8	0.2	0.0	99.2	0.8	0.0
	60	97.8	2.2	0.0	98.7	1.3	0.0
	100	98.5	1.5	0.0	98.5	1.5	0.0
	150	99.1	0.9	0.0	99.5	0.5	0.0
	200	98.9	1.1	0.0	99.4	0.6	0.0
2	20	99.0	0.0	1.0	97.4	0.1	2.5
	40	98.9	0.1	0.9	96.8	0.5	2.7
	60	97.0	1.8	1.2	96.4	2.3	1.3
	100	95.8	1.8	2.4	98.0	1.6	0.4
	150	96.5	2.5	0.9	99.5	0.5	0.0
	200	96.6	2.5	0.9	98.0	1.6	0.4
3	20	99.1	0.0	0.9	96.5	0.0	3.5
	40	97.9	0.0	2.1	96.3	0.7	3.0
	60	98.5	0.7	0.8	95.1	2.0	2.9
	100	96.7	1.2	2.1	97.6	1.5	0.9
	150	90.6	4.4	5.1	98.1	1.7	0.3
	200	91.0	5.1	3.9	96.6	2.3	1.1
Average		97.3	1.4	1.2	97.8	1.1	1.1

Note: The presented results were obtained through *MILP-BC-V I* and have been averaged over 10 instances per entry.

the sake of operational or safety reasons, it might be required that the drone must depart immediately; therefore, the drone only performs a *flyover* at the customer's location. In such a scenario, if the drone arrives at the retrieval location before the truck and is not permitted to land it will need to *hover*. While hovering, battery charge is consumed at approximately the same rate as during straight flight (refer to [9]). Such a case is investigated by Poikonen et al. [22] and in this section, we experiment using the instances provided in [22]. We begin by describing the experimental setup in Section 5.1.1. Afterwards, we present the numerical results in Section 5.1.2.

5.2.1 | Experimental setup

Poikonen et al. [22] propose a large set of instances where coordinates are distributed uniformly on a 50 by 50 grid. From this set, we test on 25 instances for each size of $n \in \{9, 14, 19\}$ customers. In these instances, as opposed to Section 5.1, it is generally assumed that the truck follows the Manhattan metric instead. The drone is still modelled with a relative velocity of $\alpha \in \{1, 2, 3\}$ w.r.t. the truck and the overhead times are furthermore assumed to be negligible, that is, $\bar{t}_l = \bar{t}_r = 0$. However, in contrast to Section 5.1, the drone has a battery life of $\mathcal{E} \in \{10, 20, 30\}$ time units and must be recovered by the truck within a time window of length $\mathcal{E}$ after each launch.

Due to the relatively poor performance of *MILP-B* in Section 5.1 and the computational overhead, for the remainder of this paper, we only consider *MILP-A*, *MILP-BC*, and *MILP-BC-VI* for solving the resulting set of 675 problems; hence, a total number of 2025 experiments were performed. In order to match our models to the changed endurance assumption, some adjustments are necessary. For this purpose, in *MILP-A*, we should replace the endurance constraints (10) with the following ones:

$$(a_j - s_i) - M(2 - y_{iw} - \widetilde{y}_{wj}) \leq \mathcal{E} \quad : \quad \forall i \in V_L,\ w \in V_N,\ j \in V_R,\ i \neq w,\ i \neq j,\ w \neq j. \tag{54}$$

These constraints state that if a drone performs a sortie (i, w, j) then the total flight and hovering time for that sortie may not exceed $\mathcal{E}$. Preliminary experiments have shown that, to enable a fairer comparison, it is in fact appropriate to keep constraints (10) in *MILP-A* as valid inequalities, that is, $\mathcal{E}$ is a valid upper bound on the time in motion (excluding hovering). As constraints (10) are not affected by the M-coefficient, this has a positive impact on the effectiveness of Gurobi in solving *MILP-A*. In

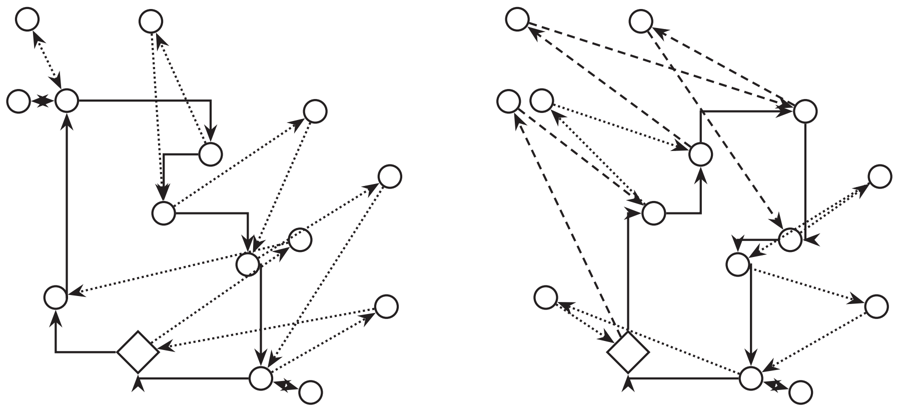

Figure 7 Optimal solutions for a problem instance with $n = 15$ vertices where the relative velocity is $\alpha = 3$ and the endurance $\mathcal{E}$ is 20 and 30 units in the left and right solutions, respectively. The sorties are indicated by dotted and dashed lines: the former can be performed with an endurance $\mathcal{E} \in [0, 20]$ and latter require an endurance $\mathcal{E} \in]20, 30]$

addition, for *MILP-BC*, we propose the following endurance constraints:

$$\sum_{\substack{i \in V_L,\\ i \neq w,\\ i \neq j}} \bar{t}_{iw} y_{iw} + \bar{t}_{wj} \widetilde{y}_{wj} + w_j^d \leq \mathcal{E} \quad : \quad \forall w \in V_N,\ j \in V_R,\ w \neq j. \tag{55}$$

These constraints state that the flyover time and hovering (nonsynchronized waiting) time at the retrieval location may not exceed the maximum endurance. Note that, round trips do by their nature not include a drone hovering time. Hence, the endurance constraints for round trips displayed in (9) remain unaffected for either formulation.

5.2.2 | Numerical results

The detailed numerical results are presented in Tables 7 and 8. To be more precise, Table 7 shows the MIP gap, runtime, the share of optimal solutions, and the relative root relaxation as average values. In addition, for formulations *MILP-BC* and *MILP-BC-V I*, the average number of cuts of type (38) and (39) is shown. Moreover, Table 8 provides an insight in the composition of the makespan and the share of customers served by the truck and drone, respectively.

Compared to the observations made in Section 5.1, as Table 7 reveals, the change in the assumptions has a noticeable influence on the solver. In more detail, small instances with $n = 9$ customers are still solved in less than 2 seconds on average by either formulation. For $n = 14$, almost all instances can be solved to proven optimality within 1 hour. In particular, *MILP-BC-V I* is more effective than *MILP-BC* and solves the same amount of instances (a share of 98.2%) to optimality as *MILP-A* in about half the time (237.5 seconds vs 383.1 seconds) on average. In contrast, *MILP-BC* solves a fraction of instances less to optimality (a share of 95.6%) with an average runtime that is similar to *MILP-A* (393.7 seconds). However, the difference in performance is especially visible for $n = 19$. Here, *MILP-BC-V I* dominates the other approaches and we can solve about 63.1% of all instances to proven optimality within 1 hour (in 1836.1 seconds with a remaining gap of just 5% on average). This is about 21% (respectively, 13%) more than can be solved by *MILP-A* (respectively, *MILP-BC*) in much shorter time. In general, the relative root relaxations are of a similar order of magnitude for all formulations with slight differences in favor of *MILP-BC-V I*. In particular, the relative velocity α and endurance R have a noticeable influence on the quality of the root relaxation. When we compare the cuts inserted through Algorithm 1, we notice that *MILP-BC-VI* generates noticeably less cuts than *MILP-BC*.

Noteworthy, for several cases in Table 7 where the relative velocity $\alpha \in \{2, 3\}$, instances with $\mathcal{E} = 20$ appear to be more difficult to solve (based on increased runtimes or gaps) than instances with $\mathcal{E} = 30$. Intuitively, one might argue that a reduced endurance should also make it easier to solve the TSP-D, as the resulting solution space is more restricted. However, a change in endurance often results in structural changes to the nature of optimal solutions. Indeed, when the endurance is large, the number of round trips is more limited, many long-range sorties are performed, and the drone is rarely taken along by the truck. As an example, Figure 7 shows two sample solutions.

Figure 8 provides insights into the savings Δ (refer to formula (53)). At first glance, the strong dependence on both the endurance R and relative velocity α is visible. Overall, for these instances where the truck follows the Manhattan instead of the Euclidean metric, we see that the savings are slightly larger than those observed in Figure 5.

Finally, Table 8 reveals that the time in motion has the largest contribution to the makespan. Moreover, waiting times are generally small and increase as R and α increase—that also causes an increase in the share of customers served through multilegs. Similar observations apply to round trips, even though, their share is generally larger compared to Table 6.

Table 7 Average MIP gap, average runtime (in seconds), the share of instances solved to optimality (opt.), and the average relative root relaxation $\%_r$ depending on the number of customers n, the relative velocity of the drone α, and the endurance R on the Poikonen et al. [22] instance set (25 instances per row)

Instance			*MILP-A*				*MILP-BC*					*MILP-BC-V I*				
n	α	$\mathcal{E}$	Gap (%)	Time	Opt. (%)	$\%_r$	Gap (%)	Time	Opt. (%)	Cuts	$\%_r$	Gap	Time	Opt.(%)	Cuts	$\%_r$
9	1	10	0.0	0.1	100	72.7	0.0	0.1	100	7.0	74.5	0.0	0.2	100	3.3	75.1
		20	0.0	0.3	100	63.3	0.0	0.4	100	20.0	65.4	0.0	0.4	100	16.6	66.1
		30	0.0	0.8	100	53.6	0.0	0.8	100	46.3	54.9	0.0	1.0	100	31.8	55.7
	2	10	0.0	0.3	100	62.4	0.0	0.3	100	20.4	65.1	0.0	0.4	100	12.1	65.7
		20	0.0	1.3	100	43.4	0.0	1.6	100	82.4	43.9	0.0	2.2	100	37.4	44.4
		30	0.0	3.8	100	41.7	0.0	3.8	100	147.7	42.0	0.0	4.1	100	79.2	42.3
	3	10	0.0	0.7	100	49.4	0.0	0.9	100	37.4	51.5	0.0	1.1	100	16.5	52.4
		20	0.0	3.1	100	36.0	0.0	4.0	100	172.0	36.3	0.0	4.3	100	65.6	36.7
		30	0.0	3.8	100	41.3	0.0	5.7	100	213.0	41.9	0.0	4.0	100	94.4	42.3
Avg. ($n = 9$)			0.0	1.6	100	51.5	0.0	2.0	100	82.9	52.8	0.0	2.0	100	39.7	53.4
14	1	10	0.0	1.8	100	73.0	0.0	1.3	100	4.3	74.4	0.0	1.4	100	4.8	74.5
		20	0.0	22.6	100	59.6	0.0	10.6	100	38.7	61.7	0.0	12.2	100	21.8	62.5
		30	0.6	445.1	96.0	48.1	0.3	359.7	96.0	162.0	48.5	0.4	343.4	96.0	65.9	48.8
	2	10	0.0	23.5	100	57.0	0.0	15.9	100	23.4	59.7	0.0	16.8	100	13.0	60.4
		20	0.0	404.3	100	41.8	0.5	412.1	96.0	291.0	41.9	1.0	552.5	92.0	85.3	42.1
		30	0.5	890.1	92.0	45.9	0.2	664.1	96.0	513.7	46.0	0.0	370.3	100	165.4	46.1
	3	10	0.3	227.7	96.0	37.1	0.0	136.7	100	166.3	38.5	0.6	211.1	96.0	45.0	39.4
		20	0.0	873.8	100	39.0	1.4	1510.4	76.0	714.4	39.1	0.0	492.4	100	167.0	39.4
		30	0.0	559.2	100	43.8	0.0	432.7	96.0	612.6	43.9	0.0	137.9	100	130.8	44.2
Avg. ($n = 14$)			0.1	383.1	98.2	49.5	0.3	393.7	95.6	280.7	50.4	0.2	237.5	98.2	77.7	50.8
19	1	10	0.0	28.0	100	71.9	0.0	15.0	100	13.2	73.2	0.0	15.7	100	10.0	73.9
		20	2.0	1071.3	84.0	51.6	1.0	648.1	88.0	150.0	52.8	0.8	539.9	92.0	89.1	53.6
		30	10.1	3259.9	20.0	49.7	6.4	2837.3	36.0	414.5	49.9	5.4	2716.9	44.0	184.7	50.0
	2	10	1.0	910.0	88.0	49.4	0.5	542.8	96.0	104.4	51.5	0.6	480.1	96.0	50.2	52.3
		20	12.7	3350.9	16.0	43.3	8.8	3120.0	36.0	527.0	43.4	7.7	2991.7	32.0	168.8	43.5
		30	14.8	3479.7	4.0	47.0	12.1	3309.4	16.0	525.1	47.1	7.6	2688.0	52.0	160.9	47.2
	3	10	5.6	2405.1	52.0	36.2	3.6	1997.9	68.0	399.2	36.9	3.9	2078.3	64.0	133.8	37.1
		20	19.7	3600.1	0.0	40.2	15.8	3600.0	0.0	634.4	40.3	13.1	2917.0	32.0	194.2	40.4
		30	14.4	3407.5	12.0	43.2	10.7	3365.3	12.0	917.6	43.3	6.3	2097.6	56.0	211.2	43.4
Avg. ($n = 19$)			8.9	2390.3	41.8	48.1	6.5	2159.6	50.2	409.5	48.7	5.0	1836.1	63.1	133.6	49.0

Note: For *MILP-BC* and *MILP-BC-VI*, this table shows also the average number of constraints (38) and (39) added as cuts.

Table 8 The proportion of the time spent moving, waiting for the drone to arrive from a multileg, and the time spent waiting while the drone is doing round trips

Instance			Makespan compositon			Customers served by		
n	*α*	*R*	Move (%)	Wait (%)	Round trip (%)	Truck (%)	Multileg (%)	Round trip (%)
9	1	10	99.7	0.2	0.1	96.9	2.2	0.9
		20	97.0	2.2	0.8	82.2	16.0	1.8
		30	95.2	3.4	1.3	72.9	25.8	1.3
	2	10	96.5	0.8	2.7	80.9	8.9	10.2
		20	94.4	2.7	2.9	63.6	29.8	6.7
		30	92.1	4.4	3.4	54.7	39.6	5.8
	3	10	92.9	0.6	6.5	68.0	11.1	20.9
		20	85.5	3.3	11.2	49.8	32.4	17.8
		30	84.7	3.9	11.4	44.0	39.6	16.4
Avg. ($n = 9$)			93.1	2.4	4.5	68.1	22.8	9.1
14	1	10	99.4	0.1	0.5	95.7	3.4	0.9
		20	96.0	2.3	1.7	77.1	20.9	2.0
		30	94.1	4.4	1.5	65.7	33.4	0.9
	2	10	92.5	0.7	6.8	76.3	10.0	13.7
		20	92.4	3.5	4.0	58.9	36.9	4.3
		30	94.2	3.9	1.9	52.6	44.3	3.1
	3	10	87.4	0.9	11.8	61.4	14.3	24.3
		20	89.9	4.4	5.7	50.0	41.1	8.9
		30	90.4	3.1	6.5	45.1	45.1	9.7
Avg. ($n = 14$)			92.9	2.6	4.5	64.8	27.7	7.5
19	1	10	98.8	0.5	0.7	93.7	4.8	1.5
		20	95.9	3.0	1.1	73.3	25.9	0.8
		30	94.4	4.0	1.6	65.7	33.5	0.8
	2	10	92.7	1.5	5.8	71.6	17.9	10.5
		20	94.0	3.8	2.2	56.2	40.0	3.8
		30	96.0	2.4	1.7	54.1	43.2	2.7
	3	10	90.1	1.8	8.1	61.5	23.6	14.9
		20	93.1	3.0	3.8	50.9	41.1	8.0
		30	93.8	3.1	3.0	49.1	45.1	5.9
Avg. ($n = 19$)			94.3	2.6	3.1	64.0	30.5	5.5

Note: Moreover, the share of customers served by the truck and drone (through multilegs and round trips, respectively) is shown. The presented results were obtained through *MILP-BC-V I* and have been averaged over 25 instances per entry.

5.3 | Instances proposed by Murray and Chu

In this section, we complete our numerical study by testing on the instances proposed by Murray and Chu [20]. We describe the experimental setup in Section 5.3.1. Afterwards, the numerical results will be presented in Section 5.3.2.

5.3.1 | Experimental setup

Murray and Chu [20] propose small instances with $n = 10$ customers for the FSTSP and some larger instances with $n = 20$ customers for the *parallel drone scheduling problem* (refer to [20]) that we adapt. In these instances, customers are uniformly and independently distributed across an 8-mile square region. Moreover, the depot location corresponds either to the center of gravity, the average of the customer's x-coordinate with y-coordinate of zero (edge), or at the southwest corner of the region (origin). For these instances, it is assumed that the truck moves at 25 miles per hour on the Manhattan metric. When $n = 10$, a drone flying with 15, 25, or 35 miles per hour on the Euclidean metric is considered. In contrast, for $n = 20$, the drone is fixed at 25 miles per hour [20]. Furthermore, the endurance is either 20 or 40 minutes. Finally, for these instances, it is also assumed that the overhead times are $\bar{t}_l = \bar{t}_r = 1$ minute.

We follow the assumption of Murray and Chu [20] and do not permit round trips, that is, we set $z_{iw} = 0, \forall i \in V_L, w \in V_N, i \neq w$. Additionally, in these instances, only a limited subset $V_D \subset V_N$ of customers (that contains 80% to 90% of all customers)

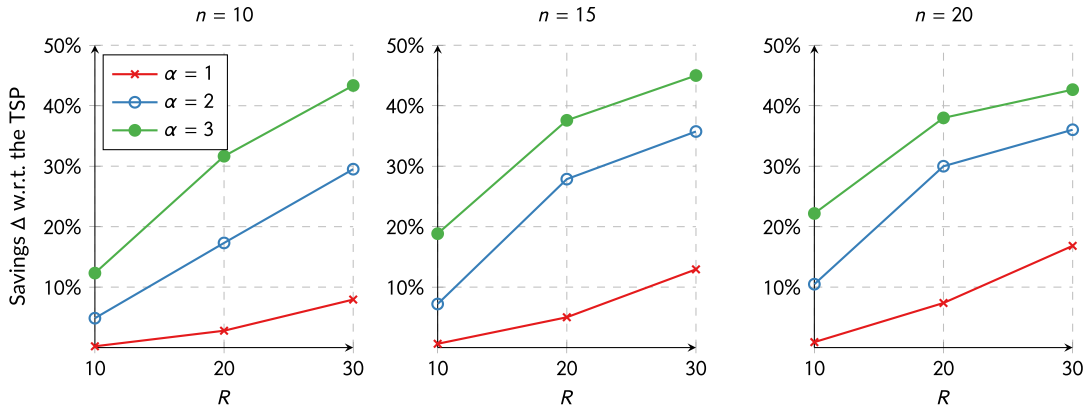

Figure 8 The average savings Δ w.r.t. the TSP solution depending on the drone's relative velocity α and its endurance $\mathcal{E}$ for the results obtained through *MILP-BC-VI* on the instances proposed in [22] [Color figure can be viewed at wileyonlinelibrary.com]

Table 9 Average MIP gap, average runtime (in seconds), and the average relative root relaxation $\%_r$ depending on the drone velocity v_d and endurance $\mathcal{E}$ on the Murray and Chu [20] instance set (12 instances per entry)

Instance		***MILP-A***			***MILP-BC-VI***				**Makespan**			
v_d	$\mathcal{E}$	**Gap (%)**	**Time**	$\%_r$	**Gap (%)**	**Time**	$\%_r$	**Cuts**	**Move (%)**	**Wait (%)**	**Overhead (%)**	**Multilegs (%)**
15	20	0	11.3	54.7	0	8.2	56.4	47.1	97.9	0.0	2.1	6.7
	40	0	53.9	53.1	0	33.8	53.8	575.6	96.3	0.0	3.7	10.8
25	20	0	29.9	53.2	0	16.6	53.6	172.3	91.7	0.8	7.5	20.0
	40	0	17.4	54.5	0	16.9	54.9	304.6	89.8	1.3	8.9	23.3
35	20	0	12.5	54.8	0	10.0	55.3	197.2	84.2	3.2	12.6	30.8
	40	0	16.3	55.1	0	10.6	55.6	241.8	84.5	2.8	12.7	30.8
	10)	0	23.6	54.2	0	16.0	54.9	256.4	90.7	1.4	7.9	20.4

Note: For *MILP-BC-VI*, the average number of constraints (38) and (39) added as cuts is shown. Moreover, this table presents the makespan composition and the share of customers served by the drone through multilegs (based on the results of *MILP-BC-VI*).

is eligible to be served by the drone. Hence, we must add the following restrictions to the models:

$$y_{iw} = 0 \quad : \quad \forall i \in V_L,\ \ w \in V_N \setminus V_D,\ \ i \neq w, \tag{57}$$

$$\tilde{y}_{wj} = 0 \quad : \quad \forall w \in V_N \setminus V_D,\ \ j \in V_R,\ \ w \neq j. \tag{58}$$

In total, Murray and Chu [20] provide 72 instances with n = 10 customers and 240 instances with n = 20 customers. To limit the number of experiments, we only solve these instances through *MILP-A* and *MILP-BC-VI*, as they have shown the best performance.

5.3.2 | Numerical results

Tables 9 and 10 present the results for the instances with $n = 10$ and $n = 20$ customers, respectively. In these tables, we present the MIP gaps, runtime, and relative root relaxation (as well as the cuts for *MILP-BC-VI*) as average values. Moreover, we also indicate the components that contribute to the makespan as well as the share of customers served through multilegs.

In Table 9, we observe that both formulations can solve instances with $n = 10$ customers to optimality in short computation time. As in the previous sections, here, *MILP-BC-VI* also outperforms *MILP-A* and can solve all small instances in 16 seconds on average. However, on these instances, the time to compute the optimal solution is almost one order of magnitude larger compared to the small instances ($n = 9$) considered in Sections 5.1 and 5.2.

In Table 10, we present the results for $n = 20$ customers. As the drone speed is fixed to 25 miles per hour on these instances, the table highlights performance w.r.t. the depot location. Generally, we observe that *MILP-BC-VI* also performs better than *MILP-A* on these instances: we observe improved MIP gaps, runtimes, and a larger share of instances solved to optimality. In particular, we observe a strong impact of the depot location and endurance on the difficulty of the problem. Generally, going from $\mathcal{E} = 20$ to $\mathcal{E} = 40$ makes the problem more difficult. Moreover, instances where the depot is located along the center of an edge appear to be the most difficult.

Finally, it is worth highlighting that the share of customers served by the drone (through multilegs) is generally smaller in both tables (i.e., Tables 9 and 10) than what we observed in previous sections. The reason might be related to the presence of

Table 10 Average MIP gap, average runtime (in seconds), the share of optimal solutions (opt.), and the average relative root relaxation $\%_r$ depending on the depot location and endurance $\mathcal{E}$ on the Murray and Chu [20] instance set (40 instances per entry)

Instance		*MILP-A*				*MILP-BC-VI*					Makespan			
Depot	$\mathcal{E}$	Gap (%)	Time	Opt. (%)	$\%_r$	Gap (%)	Time	Opt. (%)	$\%_r$	Cuts	Move (%)	Wait (%)	Overhead (%)	Multilegs (%)
Center	20	12.4	3115.3	22.5	49.9	6.7	2730.6	30.0	50.6	256.2	93.5	1.2	5.3	21.0
	40	18.3	3600.0	0.0	49.9	13.4	3585.1	2.5	50.2	705.7	92.1	1.7	6.1	22.8
Edge	20	18.4	3600.0	0.0	47.2	12.9	3536.6	5.0	47.6	347.2	92.1	1.9	5.9	17.1
	40	17.7	3600.0	0.0	48.9	14.7	3600.0	0.0	49.3	674.0	93.5	0.4	6.1	16.9
Origin	20	7.9	3158.8	22.5	51.8	3.5	2528.2	45.0	52.4	204.6	94.0	1.6	4.5	18.0
	40	17.6	3600.0	0.0	50.8	12.7	3520.2	5.0	51.1	1852.9	93.1	2.0	4.9	18.1
	20)	15.4	3445.7	7.5	0.5	10.6	3250.1	14.6	50.2	673.4	93.1	1.5	5.5	19.0

Note: For *MILP-BC-VI*, the average number of constraints (38) and (39) added as cuts is shown. Moreover, this table presents the makespan composition and the share of customers served by the drone through multilegs (based on the results of *MILP-BC-VI*).

overhead times. Even though their value is relatively small, summed up over all operations, they can account for up to 12.7% of the total makespan (refer to Table 9).

6 | CONCLUSION

In this paper, we investigated the TSP-D. More precisely, in Section 2, we reviewed existing (M)ILPs and other exact methods that are used for solving variants of the problem. Given the current state of research, only small-sized instances can be solved to proven optimality—even for the special case of serving every customer exactly once that is the most common one studied in the literature. Motivated by this fact, we proposed two new MILP formulations in Section 3. As an alternative, attempting to leverage the respective strengths of existing MILP and ILP approaches in the literature and motivated by our valid inequalities, in Section 4, we proposed a branch-and-cut algorithm for solving the TSP-D. Afterwards, in Section 5, we conducted an extensive computational study in order to analyze the quality of the different formulations in finding optimal TSP-D solutions on various benchmark instances from the literature. As our detailed numerical study reveals, the performance of our formulations in solving the different benchmark instances varies. However, we have confirmed that the branch-and-cut approach *MILP-BC-VI* showed strong performance throughout the different instance sets. Indeed, using this approach, we can solve small instances with 9 or 10 customers in just a few seconds and several larger instances with 19 or 20 customers within an hour. Thus, this branch-and-cut algorithm challenges the existing state-of-the-art in solving the TSP-D.

From a modelling perspective, naturally, we believe that our formulation might also be used to study variants that consider, for example, multiple trucks or drones (refer to [23,30]). Moreover, the additional variables in Section 4, that model the waiting time of the truck and drone, could be helpful in cases where these times might be constrained or associated with some cost (as it is the case in, e.g., [15]). We also believe that this formulation might be adjusted in a way to achieve the possibility of allowing each vertex to be visited more than once by the truck (refer to [1]).

From a computational perspective, using formulation *MILP-BC-VI* and a customized branch-and-cut scheme lead to the best performance overall. However, we also showed that performance varied over the different benchmark sets. Therefore, it might be worth to further investigate how the different assumptions affect the difficulty of the problem. Finally, in order to obtain a further boost, it might be worthwhile to investigate if the cutting scheme proposed in this work can be enhanced by merging it with other type of cuts proposed in, for example, [7,8]. That way, one might integrate a separation procedure into the branch-and-cut framework in order to cut off infeasible subtours as required instead of using explicit MTZ constraints.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments and suggestions that helped us in improving the quality of this paper.

ORCID

Daniel Schermer https://orcid.org/0000-0001-8171-9735
Mahdi Moeini https://orcid.org/0000-0001-6940-7691
Oliver Wendt https://orcid.org/0000-0002-7102-3141

REFERENCES

[1] N. Agatz, P. Bouman, and M. Schmidt, *Optimization approaches for the traveling salesman problem with drone*, Transp. Sci. **52** (2018), 965–981.
[2] J.P. Aurambout, K. Gkoumas, and B. Ciuffo, *Last mile delivery by drones: An estimation of viable market potential and access to citizens across European cities*, Eur. Transp. Res. Rev. **11**(30) (2019), 1–21. https://doi.org/10.1186/s12544-019-0368-2
[3] P. Bouman, N. Agatz, and M. Schmidt, *Dynamic programming approaches for the traveling salesman problem with drone*, Networks **72** (2018), 528–542.
[4] P. Bouman, N. Agatz, and M. Schmidt, Instances for the TSP with Drone (and some solutions) (Version v1.2), Zenodo, 2018. http://doi.org/10.5281/zenodo.1204676.
[5] J.C. de Freitas and P.H.V. Penna, *A randomized variable neighborhood descent heuristic to solve the flying sidekick traveling salesman problem*, Electron. Notes Discrete Math. **66** (2018), 95–102.
[6] J.C. de Freitas and P.H.V. Penna, *A variable neighborhood search for flying sidekick traveling salesman problem*, Int. Trans. Oper. Res. **27** (2020), 267–290.
[7] M. Dell'Amico, R. Montemanni, and S. Novellani, *Drone-assisted deliveries: New formulations for the flying sidekick traveling salesman problem*, Optim. Lett. (2019), 1–32.
[8] M. Dell'Amico, R. Montemanni, and S. Novellani, *Models and algorithms for the flying sidekick traveling salesman problem*. arXiv:191002559, 2019b.
[9] K. Dorling, J. Heinrichs, G.G. Messier, and S. Magierowski, *Vehicle routing problems for drone delivery*, IEEE Trans. Syst. Man Cybern. Syst. **47** (2017), 70–85.
[10] M. Drexl, *Synchronization in vehicle routing—A survey of VRPs with multiple synchronization constraints*, Transp. Sci. **46** (2012), 297–316.
[11] E. Es Yurek and H.C. Ozmutlu, *A decomposition-based iterative optimization algorithm for traveling salesman problem with drone*, Transp. Res. C: Emerg. Technol. **91** (2018), 249–262.
[12] M. Fischetti, I. Ljubić, and M. Sinnl, *Benders decomposition without separability: A computational study for capacitated facility location problems*, Eur. J. Oper. Res. **253** (2016), 557–569.
[13] M. Fischetti, I. Ljubić, and M. Sinnl, *Redesigning Benders decomposition for large-scale facility location*, Manag. Sci. **63** (2017), 2146–2162.
[14] Gurobi Optimization, LLC, Gurobi Optimizer Reference Manual, 2020.
[15] Q.M. Ha, Y. Deville, Q.D. Pham, and M.H. Hà, *On the min-cost traveling salesman problem with drone*, Transp. Res. C: Emerg. Technol. **86** (2018), 597–621.
[16] Q.M. Ha, Y. Deville, Q.D. Pham, and M.H. Hà, *A hybrid genetic algorithm for the traveling salesman problem with drone*, J. Heuristics **26**(2) 2020, 219–247.
[17] S. Kim and I. Moon, *Traveling salesman problem with a drone station*, IEEE Trans. Syst. Man Cybern. Syst. **49** (2018), 42–52.
[18] M. Marinelli, L. Caggiani, M. Ottomanelli, and M. Dell'Orco, *En route truck–drone parcel delivery for optimal vehicle routing strategies*, IET Intell. Transp. Syst. **12** (2018), 253–261.
[19] C.E. Miller, A.W. Tucker, and R.A. Zemlin, *Integer programming formulation of traveling salesman problems*, J. ACM **7** (1960), 326–329.
[20] C.C. Murray and A.G. Chu, *The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery*, Transp. Res. C: Emerg. Technol. **54** (2015), 86–109.
[21] A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch, *Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey*, Networks **72** (2018), 411–458.
[22] S. Poikonen, B. Golden, and E.A. Wasil, *A branch-and-bound approach to the traveling salesman problem with a drone*, INFORMS J. Comput. **31** (2019), 335–346.
[23] S. Poikonen, X. Wang, and B. Golden, *The vehicle routing problem with drones: Extended models and connections*, Networks **70** (2017), 34–43.
[24] A. Ponza, Optimization of drone-assisted parcel delivery, Master thesis, Università Degli Studi di Padova, 2016, pp. 1–80.
[25] D. Rojas Viloria, E.L. Solano-Charris, A. Muñoz-Villamizar, and J.R. Montoya-Torres, *Unmanned aerial vehicles/drones in vehicle routing problems: A literature review*, Int. Trans. Oper. Res. (2020), itor.12783. 1–32. https://doi.org/10.1111/itor.12783
[26] D. Sacramento, D. Pisinger, and S. Ropke, *An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones*, Transp. Res. C: Emerg. Technol. **102** (2019), 289–315.
[27] D. Schermer, M. Moeini, and O. Wendt, *A matheuristic for the vehicle routing problem with drones and its variants*, Transp. Res. C: Emerg. Technol. **106** (2019), 166–204.
[28] D. Schermer, M. Moeini, and O. Wendt, *A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations*, Comput. Oper. Res. **109** (2019), 134–158.
[29] D. Schermer, M. Moeini, and O. Wendt, *The drone-assisted traveling salesman problem with robot stations*, Proceedings of the 53rd Hawaii International Conference on System Sciences, USA: University of Hawaii; 2020, pp. 1308–1317.
[30] X. Wang, S. Poikonen, and B. Golden, *The vehicle routing problem with drones: Several worst-case results*, Optim. Lett. **11** (2017), 679–697.

How to cite this article: Schermer D, Moeini M, Wendt O. A branch-and-cut approach and alternative formulations for the traveling salesman problem with drone. *Networks.* 2020;76:164–186. https://doi.org/10.1002/net.21958