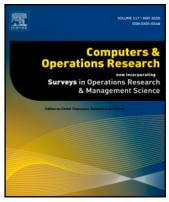
Computers & Operations Research 159 (2023) 106319

Contents lists available at ScienceDirect

Computers and Operations Research

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



A variable neighborhood search for parcel delivery by vehicle with drone cycles

Amro M. El-Adle [a,*,1], Ahmed Ghoniem [b,2], Mohamed Haouari [c,3]

[a] Department of Information Technology & Decision Sciences, Strome College of Business, Old Dominion University, 2026 Constant Hall, Norfolk, 23529, VA, USA
[b] Department of Operations & Information Management, Isenberg School of Management, University of Massachusetts Amherst, 121 Presidents Drive, Amherst, 01002, MA, USA
[c] Department of Mechanical & Industrial Engineering, College of Engineering, Qatar University, P.O. Box-2713, Doha, Qatar

ARTICLE INFO

Keywords:
Distribution
Parcel delivery
Drone
Routing

ABSTRACT

In spite of the growing literature on and relevance of vehicle-drone parcel delivery, the logistical impact of cyclic drone flights, in which a drone launches and lands at the same node (as opposed to distinct sites) while delivering to a customer in between, remains unclear. To assess the pertinence and logistical impact of drone cycles, we propose a variable neighborhood search (VNS) heuristic for the Traveling Salesman Problem with Drone (TSP-D), whereby a vehicle and its companion aerial drone are synchronously routed to deliver customer orders with the objective of minimizing the return time of both carriers to the depot. The key to the success of the proposed VNS is a two-phase intensification scheme. In the first phase, the VNS broadly explores the feasible space by temporarily limiting the scope of drone flights and rendezvous locations. In the second phase, two features are introduced to ensure a deeper exploration of the feasible space: (i) intervening visits to customers are allowed for the vehicle between the drone rendezvous (launch and re-collect) nodes and (ii) drone operations may include no cycles, single cycles, or multiple cycles. The VNS is powered by optimization models that may accommodate the diverse operational settings proposed in the TSP-D literature. Over a set of benchmark instances, the VNS improves upon the best-known results for 113/120 instances having up to 100 nodes with comparable computational effort to existing approaches. The VNS also reveals improvements of up to 1%–8% in delivery times when drone multi-cycles are permitted, over a test-bed of diverse customer topographies and instance sizes.

1. Introduction

The adoption of unmanned aerial systems, or drones, for logistical purposes is accelerating. Industrial players and public agencies have deployed what may be termed "long driveway delivery" (Moring, 2019; Adler, 2020). This system deploys a drone to deliver parcels autonomously to customer locations, whereby the drone is launched and retrieved by a driver of a traditional delivery vehicle. During the drone's flight, the driver delivers parcels simultaneously to distinct customer locations. As technology and regulations further enable drone delivery, decision systems supporting the efficient logistical operations will be essential to fully exploit the connectivity and viability of commercial drone deliveries (Tang and Veelenturf, 2019). Both industry stakeholders and policymakers may benefit from the logistical impact planning made possible by tools that solve industrial-scale instances of the Traveling Salesman Problem with Drones (TSP-D), while investigating the impact of key problem settings.

In the most common TSP-D setting, a vehicle and its companion drone synchronously deliver to customers (Chung et al., 2020). Customers must be visited by one of the two carriers, with the objective of minimizing the makespan of the tour, i.e. the return time of both vehicle and drone to the central depot. Batteries on board the drone provide a limited flight time en route to and from customer deliveries. The drone may be carried aboard the vehicle (in anticipation of a future launch) or await the vehicle at a rendezvous point upon completing a customer delivery. Alternative operational settings have been considered in the fast-growing literature on drone deployment (e.g., allowing multiple drones aboard the same vehicle or allowing the drone to visit multiple customers before returning the vehicle or the depot). The

* Corresponding author.
E-mail addresses: aeladle@odu.edu (A.M. El-Adle), aghoniem@isenberg.umass.edu (A. Ghoniem), mohamed.haouari@qu.edu.qa (M. Haouari).
[1] orcid: 0000-0002-8492-5267
[2] orcid: 0000-0001-8326-1916
[3] orcid: 0000-0003-0767-8220

https://doi.org/10.1016/j.cor.2023.106319
Received 27 September 2022; Received in revised form 15 March 2023; Accepted 15 June 2023
Available online 21 June 2023

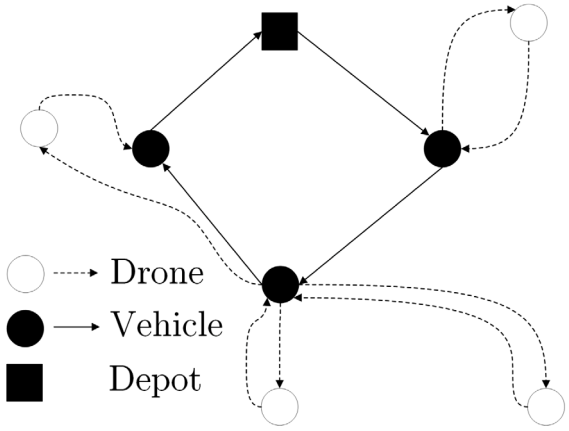


Fig. 1. A TSP-D tour beginning at the central depot and featuring (in clockwise order) a single cycle, a multi-cycle, and a non-cyclic drone flight.

literature spans exact approaches that have solved TSP-D instances with up to 39 nodes (Roberti and Ruthmair, 2021), with heuristic solutions for instances having up to 100 nodes (Agatz et al., 2018; Yurek and Ozmutlu, 2018; Schermer et al., 2019; de Freitas and Penna, 2020) among many others that have been reviewed more extensively elsewhere (Boysen et al., 2021). The focus of this study is a heuristic approach for the TSP-D, with an exploration of the logistical impact of drone cycles, under the aforementioned most common operational assumptions in the literature.

Certain conditions in drone-and-vehicle routing may lead to a drone returning to its launching location after a delivery, thereby forming a *cycle* (also known as a loop in the literature), before visiting the next customer location. Indeed, conditions conducive to a solution featuring a single drone cycle may further allow for *multiple* (i.e. two or more) such cycles at the same node. The dispersion of customer locations and the flight time of the drone may create a setting in which the shortest delivery path requires the vehicle to idle at a single location awaiting the drone as the latter conducts multiple cyclic flights, returning to the vehicle after each one. Fig. 1 illustrates a multi-cycle (with two cyclic flights). Drone cycles add new layers of decision-making to the TSP-D. This work makes the following contributions:

- Methodologically, we propose a VNS with a two-phase intensification scheme alongside a perturbation scheme that ensures diversification in the search and avoids stalling at local optima. Phase I of the intensification scheme ensures rapid improvements in the incumbent solution using a mixed-integer programming (MIP) formulation wherein, contrary to the problem settings, the vehicle is limited from visiting any intervening customers between the drone launch and the re-collect nodes. In Phase II of intensification, the feasible space is explored more thoroughly by activating two operational problem features in a second MIP formulation: (i) vehicle visits to customers are allowed between the drone launch and re-collect nodes and (ii) drone operations are allowed to include (single or multiple) cycles, as chosen by the decision-maker. Jointly, these two intensification phases balance the quality of the solutions and the associated computational effort. In contrast to extant heuristic approaches that route the vehicle alone before introducing drone flights in later stages (e.g. Agatz et al., 2018; Schermer et al., 2019; de Freitas and Penna, 2020), the proposed VNS begins with and maintains an integrated vehicle-drone tour from the outset. In so doing, the proposed VNS demonstrates the tractability and computational viability of this type of approach, particularly for solving instances with drone cycles.
- From a modeling perspective, each of the MIP formulations embedded within the two-phase intensification scheme adapts or extends existing models in the literature. The Phase I MIP adapts models introduced by Agatz et al. (2018), Boysen et al. (2018), and Roberti and Ruthmair (2021). The Phase II MIP, on the other hand, extends a formulation from El-Adle et al. (2021) by introducing drone multi-cycles in feasible solutions, and analytical results on optimality conditions for their occurrence.
- Computationally, the performance of the VNS is demonstrated on TSP-D benchmark instances, in which the heuristic yields best-known solutions to 113/120 instances in a comparison (under identical operational settings) against some benchmark instances solved by Schermer et al. (2019). The VNS also provides a platform to investigate the logistical impact and frequency of both single and multiple drone cycles in solutions to some classical benchmark instances. This analysis is timely, because few articles in the literature discuss the computational and logistical impact of drone cycles (Schermer et al., 2019, 2020; Roberti and Ruthmair, 2021; Boccia et al., 2021; Vásquez et al., 2021).

The remainder of the article is organized as follows: Section 2 reviews related literature; Section 3 introduces the problem statement and accompanying MIP formulation; Section 4 details the phases of the VNS and a second MIP formulation; Section 5 presents benchmark computational results, including a sensitivity analysis for drone cycles; and Section 6 summarizes our findings and suggests future research.

2. Literature review

A sampling of recent work (summarized in Table 1) on the TSP-D highlights the varied nature of operational features incorporated into vehicle-and-drone routing problems. According to Boysen et al. (2021), "... there may be some elaborate problem versions not yet investigated, but seeing that drones are not yet operating in mass markets and many operational details are still unclear, ... scientific research should rather concentrate on the identification of the most promising drone-delivery concept, which may vary for last-mile delivery tasks.". The scope of this article centers on MIP-based heuristic approaches for the (single vehicle and single drone) TSP-D with the objective of minimizing the duration or makespan of the tour for both carriers while allowing drone cycles. The interested reader is referred to Chung et al. (2020) for a more extensive review of vehicle-drone routing problems.

2.1. MIP-based heuristic approaches

Murray and Chu (2015) introduced the Flying Sidekick Traveling Salesman Problem (largely matching the settings of the TSP-D), for which the authors developed an exact MIP formulation and a variety of heuristic strategies based on using vehicle-only tours to derive TSP-D solutions. Bouman et al. (2018) relied on the notion of "operations" for the drone and vehicle, whereby a node visited by both carriers bookends any intervening deliveries, to motivate a 3-pass dynamic programming (DP) approach. Solutions are built sequentially: first, the DP generates a TSP tour; then drone and vehicle operations are interwoven; and finally, those operations are optimally sequenced to span all nodes. The authors used a parameter, k, to limit the number of nodes that may be visited by the vehicle while the drone is in flight. Agatz et al. (2018) used carrier "operations" that combine complementary tours for the vehicle and drone, featuring one drone flight each. An MIP formulation then sequences a combination of operations to serve all customers. The authors also presented several route-first, cluster-second heuristic approaches. Yurek and Ozmutlu (2018) proposed a two-stage approach: first, TSP tours are generated by DP; then an MIP is used to generate complementary drone flights.

Boysen et al. (2018) presented several MIPs in which the vehicle's path is given, with one or more drones being launched to serve

Table 1
A sampling of operational assumptions in recent works.

Article	Max drone customers	Farthest drone flight	Flight capacity	Deliveries per flight	Drone speed	Launch landing time	Intervening vehicle visits	Single-/ multi-cycles
Murray and Chu (2015)	50%	Fixed	Time	1	Fixed	Y/Y	Y	N/N
Agatz et al. (2018)	100%	Unlimited	Time	1	Fixed	N/N	Y	Y/Y
Boysen et al. (2018)	N/A	Unlimited	Time	1	Fixed	Y/Y	N	Y/N
Carlsson and Song (2018)	100%	Unlimited	Time	1	Fixed	Y/Y	Y	N/N
Schermer et al. (2019)	100%	Unlimited	Time	1	Fixed	N/N	Y	Y/Y
de Freitas and Penna (2020)	100%	Unlimited	Time	1	Fixed	N/N	Y	N/N
Raj and Murray (2020)	85%	Weight-dependent	Time/Range	1	Variable	Y/Y	Y	N/N
Schermer et al. (2020)	100%	Unlimited	Time	1	Fixed	Y/Y	N	Y/N
Boccia et al. (2021)	80%	Fixed	Time	1	Fixed	Y/Y	Y	Y/N
El-Adle et al. (2021)	100%	Unlimited	Time	1	Fixed	N/N	Y	N/N
Roberti and Ruthmair (2021)	100%	Unlimited	Time	1	Fixed	Y/Y	Y	Y/Y
Vásquez et al. (2021)	100%	Unlimited	Time	1	Fixed	N/N	Y	Y/N
This work	100%	Unlimited	Time	1	Fixed	N/N	Y	Y/Y

nearby customers at each stop. Heuristically, the authors also proposed constraining the drone's launching and landing locations (to disallow drone cycles) and the vehicle's movement during drone flights (i.e. with only one, rather than an unrestricted, number of intervening vehicle visits). Notably, the authors did not combine both drone cycles and multiple intervening vehicle visits during a drone flight in the same problem variant. Solving instances with up to 100 nodes, the authors concluded that limiting the vehicle's intervening visits during a drone flight is most effective when customers are clustered closely. Schermer et al. (2019) introduced both an exact MIP formulation and a heuristic approach for the Vehicle Routing Problem with Drone (VRP-D). After enhancing the formulation with valid inequalities, the authors compare its performance with a matheuristic which partitioned the VRP-D into sub-problems. The first set of sub-problems assigned customers to a particular vehicle-drone tandem; the latter sub-problems, termed the drone assignment and scheduling problem (DASP) by the authors, found an optimal path for a single vehicle and its companion drone(s) through a set of customers. The DASP does not assume a given vehicle route, but instead requires the vehicle and drone(s) to be routed simultaneously. As such, a single-vehicle DASP with a single drone is equivalent to the TSP-D. For large instances having up to 100 nodes, for which solving the DASP was intractable, the authors proposed a heuristic scheme wherein a starting feasible solution may be "partitioned" into several overlapping sections, each of which may be individually optimized. Poikonen et al. (2019) developed a branch-and-bound (B&B) algorithm, with each B&B node associated with a tour sequence. Starting at the root node, the farthest location from a customer already visited in the given sequence is chosen to be visited next. In the next level of the B&B tree, the children nodes represent a full enumeration of sequences that visit the new location. For each child node, the authors deployed an exact partitioning (EP) technique to assign locations to each carrier. The EP of a complete tour forms a valid upper bound (and a heuristic solution if the B&B algorithm is terminated early), while the EP of a partial tour generally yields a lower bound. Dell'Amico et al. (2021) investigated carrier "missions" as part of a B&B algorithm in which nodes in the search tree represent (partial) customer visit sequences for both carriers. At each node, an assignment problem is used to evaluate the feasibility and duration of the route. Heuristically, the authors propose applying the B&B algorithm on sub-sequences within a given solution to provide local improvements. Vásquez et al. (2021) used Benders Decomposition to solve the problem in two stages. With an MIP enhanced by valid inequalities, the first stage sequences a subset of customers to be visited by the vehicle; the second stage introduces drone flights and corresponding waiting times.

Roberti and Ruthmair (2021) developed a compact mixed-integer programming formulation for the TSP-D to accommodate a variety of operational assumptions, including cycles. The formulation is enhanced by a series of valid inequalities. The authors also proposed a DP-based algorithm for the TSP-D which combined with a set partitioning formulation for a branch-and-price (B&P) solution approach. The pricing problem in the B&P algorithm relaxes the TSP-D to allow customers to be served more than once in so-called *ng*-routes. A series of dominance rules are then used to pare the solution space. Furthermore, the authors introduce a three-level strategy used to branch on nodes featuring non-integral solutions to the master set-partitioning problem. The authors deploy a time-limited implementation of the B&P algorithm on benchmark instances as a heuristic solution approach. Finally, Boccia et al. (2021) deployed an exact formulation for the TSP-D with cycles (dubbed FS-TSP* by the authors) relying upon a branch-and-cut (B&C) algorithm in which sub-tour elimination constraints are added dynamically.

2.2. Drone cycles

Several studies have investigated the impact of drone cycles to some extent. Schermer et al. (2020) referred to cycles as round trips, showing they account, on average, for 3%–5% of the total makespan of the tour for instances with up to 20 nodes. Boccia et al. (2021) showed an improvement of 2–7.5% on average in makespan owing to the inclusion of drone cycles on instances having 10–20 nodes. Roberti and Ruthmair (2021) reported on the improvement in objective values and impact on computational effort when cycles are permitted, without commenting on the prevalence or relevance of single versus multi-cycles. Vásquez et al. (2021) permitted drone cycles but did not comment on their impact.

Despite a variety of extant heuristic solution approaches for the TSP-D, there remains open the question of how prevalent and logistically impactful drone cycles (and multi-cycles) may be. To that end, this work proposes a VNS building upon extant heuristics enhanced by intensification strategies.

3. Problem statement & optimization model

The problem statement of the TSP-D with the possibility of drone cycles is introduced in Section 3.1, with an MIP formulation and related notation provided in Section 3.2, and an analytical insight governing the formation of drone cycles in an optimal solution presented in Section 3.3.

3.1. Problem statement 可以模仿这篇论文的写作方式

The TSP-D is the problem of routing a single ground vehicle and its companion drone on a tour that serves each customer in a network, beginning and ending at a single depot. Hereafter, the drone and vehicle are referred to generically as *carriers* unless an explicit reference is required. The objective of the TSP-D is to minimize the return time of both carriers to the depot (i.e. the makespan of the tour). The following subsections characterize carrier movements (independently and together) and their synchronization.

3.1.1. Vehicle-drone travel

While the vehicle's driver serves customers, the drone may be launched autonomously to deliver in parallel. The drone may be launched or collected only at nodes in the network (e.g. the depot and any customer locations). The drone may launch from the depot only once; upon landing at the depot, the drone is no longer permitted to fly. The drone's carrying capacity is limited, such that the drone must return to the vehicle to be loaded with a new parcel after each delivery. Parcels are assumed to be of appropriate volume and weight to be carried by the drone to the customer, while leaving sufficient flight time for the drone to fly back (without the parcel) to the vehicle. This assumption reflects the reality of e-commerce orders, the overwhelming majority of which weigh fewer than five pounds (Otto et al., 2018; Boysen et al., 2018; Macrina et al., 2020).

3.1.2. Drone flight operations and time

In accordance with the TSP-D literature, including Macrina et al. (2020) and Boysen et al. (2021), a drone's flight time is typically limited by its battery charging state. We assume that after each flight, the drone's battery may be swapped and therefore fully replenished. Therefore the drone may be launched and retrieved as often as needed. On each individual flight, the maximum flight time of the drone is limited. Indeed, the drone's maximum flight time is known *a priori*. Notably, the maximum flight time may be set large enough to make drone service feasible for any customer from any location visited by the vehicle. In other words, the drone's flight time may be set to permit a round trip drone flight to any customer from any other customer node in the network.

3.1.3. Carrier synchronization

Either carrier may serve any customer. The two carriers may meet and await one another only at customer nodes, in which case the node is called a *rendezvous node*. After a rendezvous, the drone may be carried aboard the vehicle, or launched again to serve a customer. Additionally, the drone may launch from a node i, serve a distinct node j while the vehicle idles at i, and then return to i, which is termed a *drone cycle*. The following assumptions from the literature (Bouman et al., 2018; Schermer et al., 2019) also apply:

1. The drone travels faster than the vehicle;
2. Flight times for the drone and vehicle travel times are symmetric, and satisfy the triangle inequality.

3.2. Notation & formulation

Before presenting the MIP formulation, the following notation is introduced:

Input Parameters

- $V = \{0, 1, \dots, n\}$: Set of service nodes in the network.
- $V^+ = V \backslash \{0\}$: Set of customer nodes, which excludes the central depot (represented by node 0).
- $M, M^1, \dots, M^6$: A series of "large enough" scalars.
- τ_{ij}: Vehicle travel time from i to j, $\forall\, i, j \in V, i \neq j$.
- f_{ij}: Drone flight time from i to j, $\forall\, i, j \in V, i \neq j$.
- F: Maximum flight time for the drone to make a single delivery then return to the vehicle, due to the drone's battery.
- $C_i = \{j \in V^+ |\ f_{ij} + f_{ji} \leq F\}$: Set of nodes that may be served via drone cycle commencing at node i, visiting j, then returning to i, $\forall\, i \in V^+$.

Decision Variables

- $x_{ij} \in \{0,1\}$: $x_{ij} = 1 \iff$ the vehicle travels from i to j, $\forall\, i, j \in V, i \neq j$.
- $y_{ij} \in \{0,1\}$: $y_{ij} = 1 \iff$ the drone travels (i.e. flies or is carried aboard the vehicle) from i to j, $\forall\, i, j \in V, i \neq j$.
- $r_{ij} \in \{0,1\}$: $r_{ij} = 1 \iff$ the drone commences a cycle from i, serves j, then flies back to i, $\forall\, i \in V, j \in C_i$.
- $z_{ij} \in [0,1]$: $z_{ij} = 1 \iff$ the drone conducts a non-cyclic flight from i to j, $\forall\, i, j \in V, i \neq j$. Note that if the drone is carried aboard the vehicle, $y_{ij} = 1$ but $z_{ij} = 0$. And if the drone cycles from i to j back to i, then $r_{ij} = 1$ but $z_{ij} = z_{ji} = 0$. By construction, the z-variables are guaranteed to be binary-valued in the proposed formulation.
- $\delta_j \in [0,1]$: $\delta_j = 1 \iff$ the drone serves node j, $\forall j \in V$. By construction, these variables are guaranteed to be binary-valued in the proposed formulation.
- $a_j \in [0, M]$: The arrival time of the delivery carrier at node j, $\forall j \in V$.
- $d_j \in [0, M]$: The departure time of the delivery carrier at node j, $\forall j \in V$.
- $e_j \in [0, f_j^{max}]$: The time spent by the vehicle waiting for the drone at node j, $\forall j \in V$, where $f_j^{max} = \max\{\max_{i \in V, k \in V^+ | f_{ik} + f_{kj} \leq F} \{f_{ik} + f_{kj} - \tau_{ij}\}, 0\}$.
- $g_d \in [0, f_0^{max}]$: The drone's arrival time at the depot.
- $g_v \in [0, f_0^{max}]$: The vehicle's arrival time at the depot.

An MIP formulation for the Traveling Salesman Problem with Drone with the possibility of drone single- and multi-cycles is presented as follows as Model **TSPD**: see Box I.

The objective function (1a) minimizes the duration of the carriers' tour, accounting for the vehicle's travel time (the summation of x-variables) and its waiting time (the summation of e-variables). Constraints (1b) and (1d) ensure that the vehicle departs from the depot for exactly one destination, and thereafter travels from exactly one predecessor node to exactly one successor node. Constraints (1e) and (1f) serve the same purpose for the drone, whereas Constraint (1c) sets $\delta_i = 0$ for any node i visited by the vehicle. Constraint (1g) enforces either a cyclic or non-cyclic flight by the drone in the absence of a vehicle visit. More generally, if the drone travels from i to j, Constraint (1h) ensures that at least one of i or j must be visited by the vehicle. Constraint (1i) requires that if the drone travels from i to j where both i and j are served by the vehicle, then the vehicle must also travel from i to j (i.e. the drone is carried aboard the vehicle). Constraint (1j) requires that the drone flies from i to j if it was not transported aboard the vehicle. Constraint (1k) synchronizes the values of the z- and y-variables in case of a drone flight. In case of a drone cycle from i to j to i, Constraint (1l) ensures the vehicle visits i ($\delta_i = 0$) and Constraint (1m) assigns the drone to j ($\delta_j = 1$). Constraint (1p) guarantees that for a given flight, the drone can travel for at most a duration of F before rendezvousing with the vehicle.

Depending on which carrier serves a node, Constraints (1q) through (1u) determine arrival and departure times for the carriers in the spirit of MTZ-type subtour elimination constraints. Tightened M values are employed in order to strengthen the continuous relaxation of this formulation, as detailed in Appendix A. Two conditions determine the vehicle's idle time: a drone cycle, which is accounted for in Constraints (1r) and (1s); and in case of arriving at a node before the drone has arrived. In the latter case, Constraint (1v) sets the vehicle waiting time equal to the difference of the actual departure time and the earliest

$$\textbf{TSPD}: \text{Minimize} \quad \sum_{i\in V}\sum_{j\in V}\tau_{ij}x_{ij} + \sum_{k\in V} e_k \tag{1a}$$

$$\text{s.t.} \quad \sum_{j\in V^*} x_{0j} = 1, \tag{1b}$$

$$\sum_{j\in V} x_{ij} = 1 - \delta_i, \qquad \forall i \in V, \tag{1c}$$

$$\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = 0, \qquad \forall i \in V, \tag{1d}$$

$$\sum_{j\in V^*} y_{0j} = 1, \tag{1e}$$

$$\sum_{j\in V} y_{ij} - \sum_{j\in V} y_{ji} = 0, \qquad \forall i \in V, \tag{1f}$$

$$\delta_j \le \sum_{i\in V} y_{ij} + \sum_{i\in V: j\in C_i} r_{ij} \le 1, \qquad \forall j \in V^*, \tag{1g}$$

$$\delta_i + \delta_j \ge y_{ij}, \qquad \forall i,j \in V | i \ne j, \tag{1h}$$

$$y_{ij} + \delta_i + \delta_j \le x_{ij} + 2, \qquad \forall i,j \in V | i \ne j, \tag{1i}$$

$$y_{ij} - x_{ij} \le z_{ij} \le 1 - x_{ij}, \qquad \forall i,j \in V | i \ne j, \tag{1j}$$

$$z_{ij} \le y_{ij}, \qquad \forall i,j \in V | i \ne j, \tag{1k}$$

$$r_{ij} \le 1 - \delta_i, \qquad \forall i \in V, j \in C_i, \tag{1l}$$

$$r_{ij} \le \delta_j, \qquad \forall i \in V, j \in C_i, \tag{1m}$$

$$\sum_{j\in C_i} r_{ij} \le 1, \qquad \forall i \in V, \tag{1n}$$

$$|C_i| \sum_{k\in V} y_{ki} \ge \sum_{j\in C_i} r_{ij}, \qquad \forall i \in V^* : |C_i| > 0, \tag{1o}$$

$$\sum_{i\in V} f_{ij}z_{ij} + \sum_{k\in V} f_{jk}z_{jk} \le F + M_j^1(1-\delta_j), \qquad \forall j \in V^*, \tag{1p}$$

$$a_j \ge d_i + \tau_{ij} - M_{ij}^2(1-x_{ij}), \qquad \forall i,j \in V | i \ne j, \tag{1q}$$

$$d_j \ge a_j + \sum_{k\in C_j} (f_{jk} + f_{kj})r_{jk}, \qquad \forall j \in V^*, \tag{1r}$$

$$d_0 \ge \sum_{k\in C_0} (f_{0k} + f_{k0})r_{0k}, \tag{1s}$$

$$a_j \ge d_i + f_{ij} - M_{ij}^3(1-z_{ij}), \qquad \forall i,j \in V | i \ne j, \tag{1t}$$

$$d_j \ge d_i + f_{ij} - M_{ij}^4(1-z_{ij}), \qquad \forall i \in V, j \in V^*, \tag{1u}$$

$$e_j \ge d_j - (d_i + \tau_{ij}) - M_{ij}^5(1-x_{ij}), \qquad \forall i \in V, j \in V^* | i \ne j \tag{1v}$$

$$e_j \le d_j - (d_i + \tau_{ij}) + M_{ij}^5(1-x_{ij}), \qquad \forall i \in V, j \in V^* | i \ne j \tag{1w}$$

$$g_v \ge d_i + \tau_{i0} - M(1-x_{i0}), \qquad \forall i \in V^*, \tag{1x}$$

$$g_d \ge d_i + f_{i0} - M(1-z_{i0}), \qquad \forall i \in V^*, \tag{1y}$$

$$e_0 \ge g_d - g_v, \tag{1z}$$

$$a_0 \le \sum_{i\in V}\sum_{j\in V} \tau_{ij}x_{ij} + \sum_{k\in V} e_k, \tag{1aa}$$

$$a, d, \delta, z, e, g_d, g_v \ge 0, \quad \delta \le 1,$$

$$a, d \le M, \quad e, g_d, g_v \le f^{max}, \quad x, y, r \text{ binary}. \tag{1ab}$$

Box I.

departure time had the vehicle not needed to wait at that node. To ensure that waiting times are assigned to the relevant nodes, Constraint (1w) enforces an upper bound for each such waiting time. In the special case of waiting time at the depot, Constraints (1x) and (1y) determine the arrival time of the vehicle and drone at the depot, respectively. If the drone's arrival time exceeds that of the vehicle, then Constraint (1z) determines the non-negative value of the vehicle's waiting time. Constraint (1aa) bounds the arrival time at the depot from above by the makespan of the carriers' tour. Constraint (1ab) enforces nonnegativity and binary restrictions on variables.

3.3. Conditions for drone cycles

To help simplify the solution space associated with drone cycles in the TSP-D, Lemma 1 identifies conditions that must hold in an optimal solution featuring drone cycles:

存在无人机周期的最佳解，即如果无人机要从i点飞向k点，必定有无人机（无cycle的）在i点降落并飞离i点

Lemma 1. *Given unlimited maximal flight time, if there is a drone cycle from node i to node k, then there must be one non-cyclic drone flight landing at i and one non-cyclic drone flight departing from i.*

Proof. See Appendix B in the Supplementary Material.

之所以这样推论是和目标函数分不开的~——>最小化完工时间

Remark. In contrast, under limited maximal flight time, an optimal solution may include a drone cycle from node i to node k absent a non-cyclic drone flight landing at i. This occurs whenever a node k may not be served through a drone flight that departs from a predecessor of i and lands at i. Similarly, an optimal solution may include a drone cycle from i to k absent a non-cyclic drone flight departing from i. This latter case occurs whenever k may not be served via a drone flight departing i and landing at a successor of i.

Based on Lemma 1, the following valid inequalities may be appended to Model TSPD.

$$\sum_{k \in C_i} r_{ik} \leq \Big(\sum_{j \in V} z_{ij}\Big)|C_i|, \qquad \forall i \in V : |C_i| > 0, \tag{2a}$$

$$\sum_{k \in C_i} r_{ik} \leq \Big(\sum_{j \in V} z_{ji}\Big)|C_i|, \qquad \forall i \in V : |C_i| > 0. \tag{2b}$$

These valid inequalities ensure a node k may be served via drone cycle from node i if and only if i has at least one outgoing and one incoming non-cyclic drone flight (Constraint (2a) and Constraint (2b), respectively). The constraints are only valid with maximum flight time that permits drone service for any customer.

4. VNS overview

This section introduces the proposed VNS, referred to hereafter as the Directed Improvement Procedure (DIP). In what follows, Section 4.1 details and illustrates the proposed neighborhood structure, Section 4.2 provides an overview of the procedure, Sections 4.3 and 4.4 detail each of the two intensification phases, and Section 4.5 concludes with the diversification schemes to escape local optima via perturbation.

4.1. Neighborhood selection

DIP iteratively designates *sub-networks*, which are sub-sequences of customers in a given solution that comprise the neighborhood to be searched, while carrier movements in the remainder of the network in the given solution stay fixed. DIP also designates a search *direction*: nodes comprising a sub-network are selected in a clockwise or counterclockwise sequence mirroring the movement of the carriers. Also critical to the construction of sub-networks are *rendezvous nodes*, where both carriers are present (e.g. when the drone is carried aboard the vehicle, or when the drone launches/lands). In what follows, let R1 and R2 be the first and last rendezvous nodes considered in a sub-network, respectively. Proceeding clockwise (or counterclockwise, with accompanying notation), the sub-network begins at R1 and selects customers in increasing (decreasing) order of the carriers' arrival time until reaching R2. The predecessor of R1 (R2) and the successor of R2 (R1) are also included in the sub-network. If R1 has multiple predecessors (successors), such that the vehicle and drone each visited a distinct nodes before a rendezvous at R1, then the preceding (succeeding) node visited by each carrier is added to the sub-network.

Figs. 2 and 3 illustrate sub-networks on a small instance in the clockwise and counterclockwise directions, respectively. As compared with Fig. 2, the sub-network illustrated in Fig. 3 begins at the same rendezvous node and traces the carriers' path in the counterclockwise direction (in decreasing order of arrival times) to capture nodes that might not have otherwise been included in the same sub-network. Both figures show sub-networks in which one or more predecessors and successors are included.

Since sub-networks are bookended by rendezvous nodes, two parameters govern the sub-network's length: a soft lower bound of $L_{\min}$ and a hard upper bound of $L_{\max}$. Starting at R1 and proceeding in the designated direction, nodes are appended to the sub-network in order of their arrival times until the sub-network reaches a length of at least $L_{\min}$. If the last node appended was a rendezvous node, then that node is designated R2, its successor(s) are added to the sub-network, and the selection process ends. If, however, the last node appended was not a rendezvous node, then more nodes are added. Once R2 is designated and its successors added to the sub-network, the length of the sub-network is examined. If a sub-network has length greater than $L_{\max}$, then R2 and its successors are removed. This is repeated until the sub-network length is less than or equal to $L_{\min}$. For example, the sub-network illustrated in Fig. 5 has $L_{\min} = 6$. If $L_{\max} = 6$ for the same sub-network, then R2 would be shifted counterclockwise from Node 4 to Node 2 to reduce the sub-network's length back to that pictured in Fig. 4. As such, $L_{\min}$ and $L_{\max}$ serve as a "soft" floor and "hard" ceiling for sub-network length, respectively, to vary the *size* of neighborhoods considered by DIP.

网络长度要超过Lmin,但是必须小于Lmax

4.2. DIP outline

In general, DIP follows these steps repeatedly until convergence:

1. **Initialization:** DIP may commence with any feasible solution to the TSP-D (e.g. Agatz et al., 2018, Boysen et al., 2018, or El-Adle et al., 2021), including a vehicle-only solution.
2. **Solution Scanning:** Designate a search direction (starting with clockwise as default), and repeat Steps 3–6 until every node in the network has been captured in at least one sub-network. Then switch directions, and repeat.
3. **Sub-network Selection:** Designate a sub-network.
   - The sub-network begins with the last node explored in the preceding sub-network. If no prior sub-network have been explored, the sub-network commences at the depot, regardless of the direction being explored.
4. **Re-optimization:** Depending upon the intensification phase, invoke the pertinent MIP model (detailed in the next subsections) to re-optimize the carriers' routes within sub-networks.

   将子网外的内容设置为固定，调用MIP模型求解子网内部的解
   - To focus on the sub-network exclusively, fix all routing/assignment variables associated with nodes outside the sub-network to their values in the incumbent solution.
   - If the incumbent solution is improved, deactivate any perturbation/diversification schemes and reset the non-improving iterations (*NII*) to 0.
   - If the incumbent solution is not improved, increment *NII* by 1. 解没有优化那么NII+1
5. **Diversification/Perturbation Schemes:** If a pre-specified *NII* value is reached, activate schemes (detailed in Section 4.5) to vary sub-network length and computational effort per iteration. Then return to Step 2. If the schemes are already active, proceed to Step 7.
6. **Sub-network Splicing:** Given the re-optimized solution, select the final rendezvous node from the previous iteration as the starting node for the next sub-network. Fig. 6 illustrates an example where the terminal rendezvous node (R2 on the left-hand side) in one sub-network serves as the starting rendezvous node for the next. 方向修改

随机从一个可行解开始

搜索子网并改变，再逆时针搜索子网并改变，相关操作是3-6

如果之前没有子网就从仓库随机方向开始。有的话就从上一个子网的最后一个点开始

如果NII到达一定数量，就增加子网长度和计算辆然后从2重新开始，不然就从7

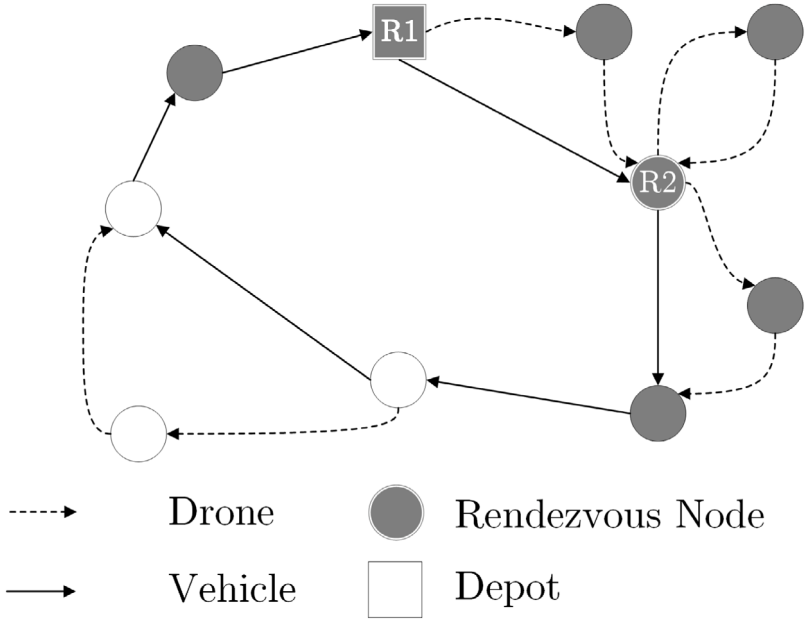


Fig. 2. The shaded nodes illustrate a sub-network comprising two rendezvous nodes (R1 and R2), the predecessor(s) and successor(s) of those nodes, and all nodes visited in between.

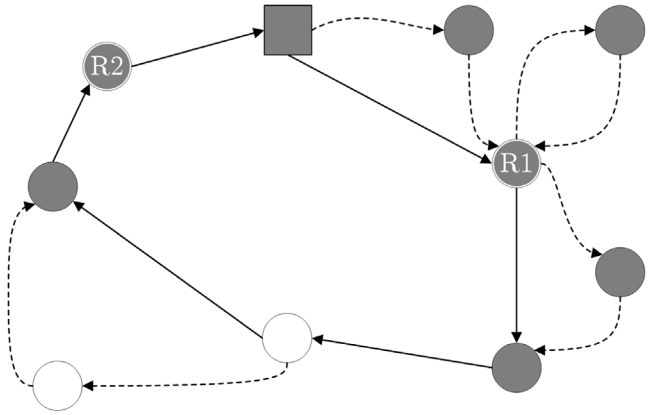


Fig. 3. An illustrative sub-network beginning at R1 and tracing the carriers' path back to R2 in the counterclockwise direction. Only the shaded nodes are part of the sub-network.

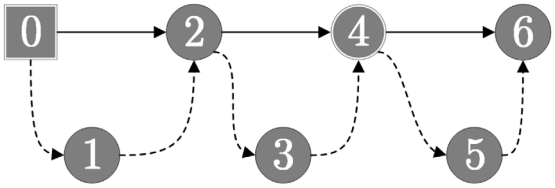


Fig. 4. This sub-network encapsulates seven nodes. If $L_{\min} = 5$, then this sub-network is too long.

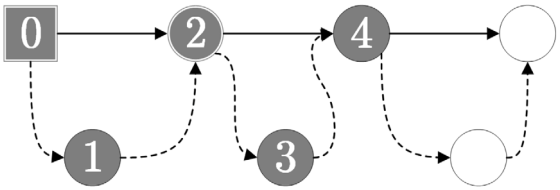


Fig. 5. Given $L_{\min} = 5$, then the sub-network from Fig. 4 would be reduced to include five nodes as shown.

- In the clockwise direction, the final rendezvous node has the latest departure time in the sub-network.
- In the counterclockwise direction, the final rendezvous node has the earliest arrival time.
- This step further diversifies the *composition* of neighborhoods selected by DIP.

7. **Phase Termination:** Conclude Phase I and proceed to Phase II, or terminate DIP if in Phase II.

4.3. Intensification phase I: Rapid improvement

Phase I of DIP is designed to generate quick improvements. To achieve that while retaining tractability, the formulation deployed therein incorporates a number of simplifications from the original problem statement. This section begins with an introduction of the notation and problem formulation for Phase I before detailing those simplifications.

4.3.1. Notation & formulation

The following notation is used to introduce the Phase I MIP formulation for the TSP-D:

Input Parameters

- $\hat{V} = \{0, 1, \ldots, n, n+1\}$: Set of all service nodes in the network, where customers are represented by nodes $1, \ldots, n$.
- $\hat{V}^0 = \hat{V} \backslash \{n+1\}$: Set of customer nodes and the dummy node representing the departure depot (0).
- $\hat{V}^{n+1} = \hat{V} \backslash \{0\}$: Set of customer nodes and the dummy node representing the arrival depot $(n+1)$.
- $\hat{V}^+ = \hat{V} \backslash \{0, n+1\}$: Set of customer nodes, excluding the central depot (represented by dummy nodes 0 and $n+1$).
- τ_{ij}: Vehicle travel time from i to j, $\forall\, i, j \in \hat{V}, i \neq j$.
- f_{ij}: Drone flight time from i to j, $\forall\, i, j \in \hat{V}, i \neq j$.
- F: Maximum flight duration for the drone to make a single delivery then return to the vehicle, due to the drone's battery.
- $\hat{M}_{ij}$: A "large enough" scalar, $\forall\, i, j \in \hat{V}, i \neq j$.
- $C_{\max}$: An upper bound on the optimal objective value (based on a feasible solution to the instance at hand, if available). Nominally $C_{\max} = \infty$, with details in Section 5.1 for the value used in this study.

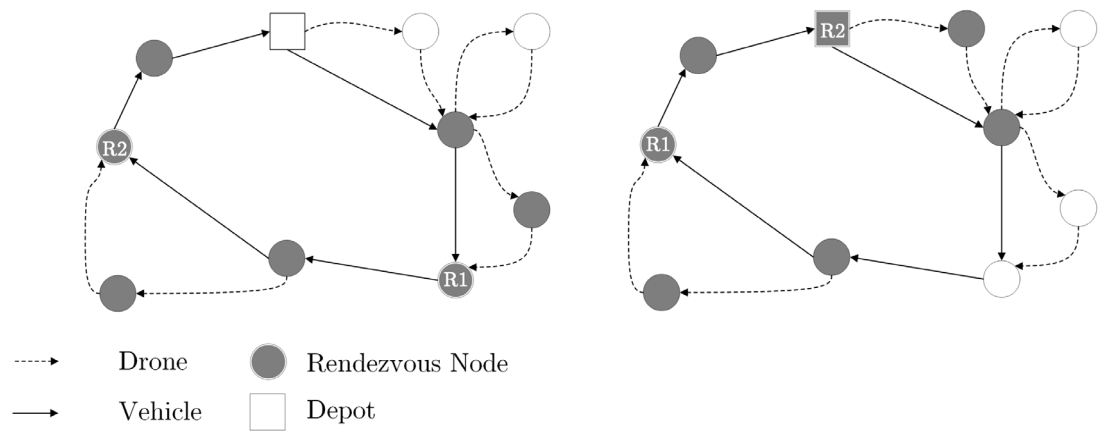


Fig. 6. Illustration of one sub-network's ending rendezvous node (R2, left) being spliced into the starting rendezvous node (R1, right) for the next sub-network.

- Φ: A share of the nearest nodes to be considered in the sparsified network comprising variables.
- $\tau^i_{\max}$: The value of τ_{ij} such that j is the $\lceil \Phi n \rceil$ nearest node to i, $\forall\, i, j \in \hat{V}$, $i \neq j$.
- $\phi_i = \{j \in \hat{V} | \tau_{ij} \leq \tau^i_{\max}\}$: A sparsified set of nodes near $i \in \hat{V}$.
- $E_k = \{(i,j) \in \hat{V} \mid f_{ik} + f_{kj} \leq F, \forall\, i \in \hat{V}^0, j \in \phi_i\}$: All pairs of rendezvous nodes (i,j) permitting drone delivery to customer k, with distinct i, j, k and $k \in \hat{V}^+$.
- $D = \{j \in \hat{V}^+ \mid |E_j| > 0\}$: Set of customers with rendezvous nodes that make them eligible for drone delivery.
- $\hat{C}_i = \{j \in \hat{V}^+ | f_{ij} + f_{ji} \leq F\}$: Set of nodes which may be served via drone cycle commencing at node i, $\forall i \in \hat{V}^+$.
- $\hat{f}_{ikj} = \max(0, f_{ik} + f_{kj} - \tau_{ij})$: The vehicle's waiting time for a drone flight originating from i, delivering to k, and landing at j, $\forall\, i \in \hat{V}^0, k \in D, j \in \hat{V}^{n+1}$ with distinct i, j, k.

Decision Variables

- $\hat{x}_{ij} \in \{0,1\}$: $\hat{x}_{ij} = 1 \iff$ the vehicle travels from i to j, $\forall\, i \in \hat{V}^0, j \in \phi_i$.
- $\hat{y}_{ikj} \in \{0,1\}$: $\hat{y}_{ikj} = 1 \iff$ the drone launches from i, delivers to k, and lands at j, $\forall\, k \in D$, $(i,j) \in E_k$.
- $\hat{z}_{ij} \in \{0,1\}$: $\hat{z}_{ij} = 1 \iff$ the drone commences a cycle from i, serves j, then flies back to i, $\forall\, i \in \hat{V}^+, j \in \hat{C}_i$.
- $\hat{\delta}_j \in [0,1]$: $\hat{\delta}_j = 1 \iff$ the drone serves node j, $\forall j \in \hat{V}^+$. By construction, these variables are guaranteed to be binary-valued in the proposed formulation.
- $b_j \in [0, C_{\max}]$: The departure time of the vehicle at node j, $\forall j \in \hat{V}$.

The Phase I formulation for the Traveling Salesman Problem with Drones is presented as follows, and denoted **Model TSPD0**: see Box II.

The objective function (3a) minimizes the makespan of the tour for all carriers across three routing decisions: the summation of $\hat{x}$-variables minimizes the travel time of the vehicle; and the summation of $\hat{z}$-variables and $\hat{y}$-variables account for the vehicle's idle time during drone cycles and non-cyclic drone deliveries, respectively. Since all routing variables relating to $i \in \hat{V}$ are defined over sets ϕ_i, setting $\Phi < 1$ sparsifies the network. Constraints (3b) and (3c) ensure the vehicle's path begins and ends at the depot (represented as nodes 0 and $n+1$, respectively), whereas Constraint (3d) ensures a Hamiltonian path through the intervening customer nodes. Constraint (3e) restricts the vehicle from intervening deliveries during a drone flight. Constraint (3f) guarantees the vehicle is present at nodes from whence cycles commence and Constraint (3g) allows at most one non-cyclic drone flight per rendezvous node. Constraint (3h) ensures each customer must be served, while Constraints (3i)–(3k) require each customer to be served either via a vehicle delivery, a drone flight, or a drone cycle.

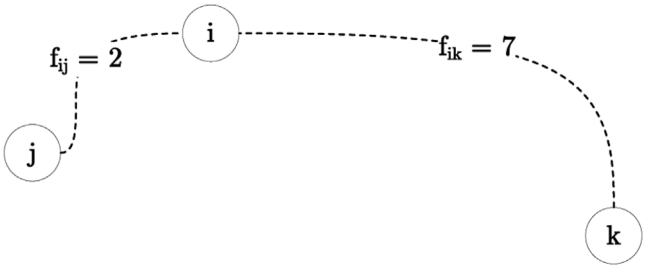


Fig. 7. Illustrative example for drone delivery possibilities with maximum flight time $F = 10$.

Although every node may be served by the vehicle, the constraints limit the appropriate set of drone routing decisions: Constraint (3i) considers nodes that are eligible for both drone flights and drone cycles; Constraint (3j) considers nodes that may be served by drone cycle but do not have rendezvous nodes available for non-cyclic drone service; and Constraint (3k) considers nodes that must be served by the vehicle because they are not eligible for cyclic nor non-cyclic drone flights.

Model TSPD0 makes a distinction between a customer i who may be eligible for non-cyclic drone service (i.e. $|E_i| > 0$) and the same customer being eligible for cyclic drone service from node j (i.e. $i \in \hat{C}_j$). The distinction is necessary because of examples like the one in Fig. 7, where customer i is eligible for both non-cyclic drone service and one drone cycle from j. In that figure, supposing the drone's maximum flight duration $F = 10$, then customer i may be served via drone cycle (from j), or from a non-cyclic drone flight from j to i to k. But customer k is not eligible for cyclic nor non-cyclic drone service. Therefore being eligible for one type of drone service neither automatically qualifies nor disqualifies a customer from eligibility for the other type. Since a customer may be eligible for non-cyclic drone service, or cyclic drone service, both, or neither, Model TSPD0 accounts for all of these possibilities.

At each node, Constraint (3l) sets the departure time of the vehicle, with a secondary purpose of setting subtour elimination constraints of the Miller-Tucker-Zemlin (MTZ) type. Setting $\hat{M}_{ij}$ as follows improves the continuous relaxation of the formulation:

$$\hat{M}_{ij} = C_{\max} - \tau_{i,n+1} + \tau_{ij} - \tau_{0j}, \qquad \forall i \in \hat{V}^0, \forall j \in \phi_i.$$

Constraint (3m) enforces an upper bound on the carriers' arrival time at the depot (which is represented by the departure time at the artificial node $n+1$), since the latest possible departure time from any node must be less than the length of the entire tour.

Based on Lemma 1, the following valid inequality may be appended to Model TSPD0 whenever the drone's maximum flight time is unlimited to ensure that a node k may only be served via drone cycle from

$$\textbf{TSPD0}: \text{Minimize} \quad \sum_{i\in\hat{V}^+}\sum_{j\in\phi_i}\tau_{ij}\hat{x}_{ij} + \sum_{i\in\hat{V}^+}\sum_{k\in\hat{C}_i}(f_{ik}+f_{ki})\hat{z}_{ik} + \sum_{k\in D}\sum_{(i,j)\in E_k}\hat{f}_{ikj}\hat{y}_{ikj} \tag{3a}$$

$$\text{s.t.} \quad \sum_{j\in\phi_0}\hat{x}_{0j} = 1, \tag{3b}$$

$$\sum_{j\in\phi_{n+1}}\hat{x}_{j,n+1} = 1, \tag{3c}$$

$$\sum_{j\in\phi_i}\hat{x}_{ij} - \sum_{j\in\hat{V}^0:i\in\phi_j}\hat{x}_{ji} = 0, \qquad \forall i\in\hat{V}^+, \tag{3d}$$

$$\hat{y}_{ikj} \le \hat{x}_{ij}, \qquad \forall k\in D,(i,j)\in E_k, \tag{3e}$$

$$\sum_{k\in\hat{C}_i}\hat{z}_{ik} \le 1-\hat{\delta}_i, \qquad \forall i\in\hat{V}^+ : |\hat{C}_i|\ge 1, \tag{3f}$$

$$\sum_{k\in D:\{(i,j)\}\in E_k}\hat{y}_{ikj} \le 1, \qquad \forall i\in\hat{V}^0, j\in\phi_i \tag{3g}$$

$$\sum_{j\in\phi_k}\hat{x}_{kj} = 1-\hat{\delta}_k, \qquad \forall k\in\hat{V}^+, \tag{3h}$$

$$\sum_{(i,j)\in E_k}\hat{y}_{ikj} + \sum_{i\in\hat{V}^+}\hat{x}_{ik} + \sum_{i\in\hat{C}_k}\hat{z}_{ik} = 1, \qquad \forall k\in D : |\hat{C}_k|\ge 1, \tag{3i}$$

$$\sum_{i\in\hat{V}^0}\hat{x}_{ik} + \sum_{i\in\hat{C}_k}\hat{z}_{ik} = 1, \qquad \forall k\in\hat{V}^+\backslash D : |\hat{C}_k|\ge 1, \tag{3j}$$

$$\sum_{i\in\hat{V}^0}\hat{x}_{ik} = 1, \qquad \forall k\in\hat{V}^+\backslash D : |\hat{C}_k| = 0, \tag{3k}$$

$$b_j \ge b_i + \tau_{ij} + \sum_{k\in D:(i,j)\in E_k}\hat{f}_{ikj}\hat{y}_{ikj} + \sum_{k\in C_i}(f_{ik}+f_{ki})\hat{z}_{ik} - \hat{M}_{ij}(1-\hat{x}_{ij}), \qquad \forall i\in\hat{V}^0, \forall j\in\phi_i, \tag{3l}$$

$$b_{n+1} \le \sum_{i\in\hat{V}^0}\sum_{j\in\phi_i}\tau_{ij}\hat{x}_{ij} + \sum_{i\in\hat{V}^0}\sum_{k\in\hat{C}_i}(f_{ik}+f_{ki})\hat{z}_{ik} + \sum_{k\in D}\sum_{(i,j)\in E_k}\hat{f}_{ikj}\hat{y}_{ikj}, \tag{3m}$$

$$b,\hat{\delta}\ge 0, \quad \hat{x},\hat{y},\hat{z} \text{ binary}. \tag{3n}$$

Box II.

node i if and only if i has at least two incident non-cyclic drone flights:

$$2\Big(\sum_{k\in\hat{C}_i}\hat{z}_{ik}\Big) \le \sum_{k\in D}\sum_{(i,j)\in E_k}\hat{y}_{ikj} + \sum_{k\in D}\sum_{(j,i)\in E_k}\hat{y}_{jki}, \qquad \forall j\in\hat{V}^+ \tag{4a}$$

Model TSPD0 aims for good quality feasible solutions in manageable times in the early phase of the heuristic. Using Model TSPD0 in Phase I, the routing/assignment variables (i.e. $\hat{x},\hat{y},\hat{z},\hat{\delta}$) are fixed or left free during the re-optimization step depending on whether they relate to a node inside or outside the sub-network being re-optimized, respectively. This partial fixing of key decision variables mitigates the computational effort associated with invoking Model TSPD0. In contrast, the timing variables (i.e. b) are left free, since re-optimizing a sub-network may affect departure times of nodes outside of the sub-network.

4.3.2. Temporary phase I simplifications

To mitigate the computational burden while searching for high-quality solutions, the Phase I formulation simplifies (temporarily until Phase II) the original problem statement as follows:

1. **Network Sparsity:** for each node i in a network with n nodes, let ϕ_i be a set comprising the nearest $\lceil \Phi n \rceil$ nodes to i, where Φ represents a specified ratio of the network's edges to be retained. Setting $\Phi < 1$ and fixing to 0 the values of variables that route the carriers from i to nodes outside of ϕ_i reduces the solution space of the formulation.
2. **Intervening Vehicle Visits:** During a drone flight, the original problem statement allows the vehicle to serve any number of customers. Both Bouman et al. (2018) and Boysen et al. (2018) limited intervening vehicle deliveries during drone flights to mitigate computational effort while still yielding high-quality solutions. The relevant constraints in the Phase I formulation ensure that a drone flight launching from node i, serving node k, and landing at node j is only feasible if the vehicle travels from node i to j directly, without intervening deliveries.
3. **Disallowing Drone Multi-Cycles:** Where feasible, multiple drone cycles may be launched from the same node in the original problem statement. In the Phase I formulation, however, at most one drone cycle (i.e. a single cycle) may launch per rendezvous node.

By focusing on routing decisions with shorter inter-node distance, the Phase I formulation allows the two carriers to rendezvous more frequently, and temporarily suspends the requirement that one wait for the other. Combining the restrictions discussed above not only simplifies the vehicle routing decisions, but also by extension reduces the number of feasible drone flights. Solutions featuring these temporarily simplified assumptions are always feasible, albeit possibly sub-optimal.

4.4. Intensification phase II: Deeper exploration

Phase II re-explores the Phase I solution, with the goal of rectifying elements of the problem that were previously simplified. To exhaustively consider all vehicle and drone routing decisions possible under the problem settings from Section 3.1, the Phase II formulation considers the full scope of routing decisions (i.e. an unsparsified network), allows intervening deliveries for the vehicle during drone flights, and permits (single) drone cycles, multi-cycles, or no cycles depending on

the constraints deployed. As compared with its predecessor in Phase I, Model TSPD as deployed in Phase II affords a full exploration of the solution space based on the original problem statement.

When using Model TSPD in Phase II, the routing/assignment variables (i.e. r, x, y, z, δ) may be fixed or left free for optimization depending on the composition of the sub-network at hand. On the other hand, the arrival/departure time variables (i.e. a, d, e, g_d, g_v) for the carriers must not be fixed during re-optimization to allow for improvements to the incumbent solution. Finally, as alluded to earlier, Model TSPD may accommodate a single drone cycle, multiple drone cycles, or disallow cycles at each eligible node. As presented, Constraint (1n) permits a single drone cycle. To allow multi-cycles, Constraint (1n) may be dropped. To disallow cycles completely, the r-variables in the formulation may be fixed to 0. In that case, it is also necessary to fix the $\hat{z}$-variables from Model TSPD0 to 0, to ensure that the drone cycles are disallowed in both phases of DIP. The impact of drone cycles is further discussed in Section 5.5.

4.5. Diversification/perturbation schemes & termination criteria

In spite of the diversification strategies discussed so far, DIP may eventually become entrapped at a local optimum. To mitigate that possibility, whenever DIP reaches a certain number of successive non-improving iterations (*NII*), it deploys a mix of schemes designed to further vary the search:

- **Expanded Search:** Given $L_{\max}$, the hitherto maximum sub-network length, let $L^e_{\min}$ be a larger number of nodes to be explored in sub-networks, such that $L^e_{\min} > L_{\max}$ in each phase. Under the expanded search, $L^e_{\min}$ becomes the floor for sub-network length. Moreover, the "hard" ceiling of $L_{\max}$ for sub-network length is disabled, so that the minimum sub-network length is at least $L^e_{\min}$.
- **Early Iterations:** During both phases, the earliest iterations of DIP are most likely to yield improvements, since the solution space has yet to be explored fully. Therefore DIP permits extended computational effort for re-optimization until the solution has been scanned once in both the clockwise and counter-clockwise directions.
- **Phase Transition:** In expanding the solution space, the transition between Phases I and II serves as a perturbation scheme.

Although these schemes aid in exploring the solution space, they are also computationally demanding, and are therefore best deployed parsimoniously. As described in Section 4.2, whenever DIP yields an improved solution, *NII* is reset and any diversification schemes are de-activated immediately.

Finally, DIP terminates after each phase of the heuristic has reached a pre-specified number of non-improving iterations. That is, Phase I of DIP follows the procedure outlined in Section 4.2. After the Phase I search has terminated, the solution from Phase I is translated into the formulation deployed in Phase II, at which point the procedure outlined in Section 4.2 is again followed. When Phase II reaches a pre-specified number of non-improving iterations, DIP terminates.

5. Computational study

This section outlines parameter settings for DIP used in this study, introduces a schema for generating instances featuring drone cycles, as well as an analysis relating to drone cycles. All computational results were obtained on a machine with an Intel i7-7700K processor and 32 GB of RAM. DIP was implemented using AMPL, with both formulations solved using GUROBI version 9.0.2.

5.1. Search parameter settings

This section describes our computational experience setting parameters that govern DIP's search, as discussed in Section 4. Firstly, DIP initializes with a greedy deterministic tour involving both the vehicle and drone, as proposed by El-Adle et al. (2021). To ascertain the parameter settings for DIP, the formulation from each phase was used to solve (exactly, rather than heuristically) a series of small instances (in increasing size) having fewer than 20 nodes. Those runs were time-limited, and designed to explore the settings with which the solver could most efficaciously and efficiently yield improving feasible solutions. From those runs a collection of promising parameter settings, including settings for the solver as well as settings for DIP (such as sub-network length), were ascertained. By testing over the entirety of the test-bed, the best of these settings, determined on the basis of solution quality and speed, are recommended as follows.

In our computational experience, DIP performed best with $\Phi = \frac{2}{3}$ for the network sparsity setting (in Phase I). Moreover, we set $NII = 2$, which remained consistent in both phases of DIP. The length of sub-networks may and in fact did vary across phases of DIP. Longer sub-networks generally hold greater potential for an improved solution at increased computational expense. Shorter sub-networks may yield only modest improvements to the solution, thereby requiring a greater number of algorithmic iterations until a termination criterion is met. Due to the relaxed settings deployed during Phase I, DIP is capable of re-optimizing relatively large sub-networks in that phase. Thus we set $L_{\min} = 16, L_{\max} = 25$, and $L^e_{\min} = 20$. Shorter sub-networks were generally more effective in Phase II, when we set $L_{\min} = 8, L_{\max} = 12$, and $L^e_{\min} = 16$. Note that $L_{\max}$ is a non-binding upper limit on the length of the sub-network, so that instances smaller than $L_{\max}$ may be solved by the algorithm. But $L^e_{\min}$ is a binding lower limit on the length of the sub-networks, and thus must be greater than or equal to the size of the instance. The ideal sub-network length for each phase allows significant improvements within manageable computational efforts per iteration.

These parameter values were tuned empirically, through a series of runs that balanced the computational effort of DIP with its ability to yield high-quality results. For simplicity, the parameter values were unified for the test-bed of instances in this study. The parameter tuning tools available with commercial solvers would aid in the optimization of these values for alternative instances and computational settings.

5.2. Problem instances

Benchmark instances for the TSP-D have been published by Agatz et al. (2018), where the authors provide details of the generation schemes. This study explores two topographies therein: nodes distributed randomly (the Uniform topography); and nodes clustered in a pre-determined pattern (the Single Center topography). Agatz et al. (2018) also published a third topography (Double Center) which was, in our experience, far less amenable to drone cycles. Those results have been omitted from this article, but are available online.[4] The benchmark instances have 20, 50, and 100 nodes (each size having 10 distinct instances) with the following assumptions published by the authors of the instances:

- All nodes are connected with symmetrical Euclidean travel distances;
- The drone's speed is defined as a multiple of the vehicle's speed. Unless otherwise noted, let $\alpha = 2$ be that ratio, such that the drone moves twice as fast as the vehicle;

[4] https://bit.ly/3fkHt1M

Table 2
A Comparison of exact and heuristic approaches for the TSP-D with cycles.

Solution approach	$\Delta_{\mu}^{180}(\%)$	$\Delta_{\mu}^{end}(\%)$	CPU (s)	Opt.	MIP gap (%)
Model TSPD (Phase II)	0.0	−0.2	1,433.4	7/10	5.2
Roberti and Ruthmair (2021)	27.8	10.8	3,600.0	0/8[a]	∞[b]
Model TSPD0 (Phase I)	1.0	1.0	101.0	0/10	–
DIP	0.0	0.0	123.0	5/10	–

[a]The formulation did not produce a feasible solution for two instances after one CPU hour.
[b]The lower bounds across all instances was zero after one CPU hour.

- Following Agatz et al. (2018) and Schermer et al. (2019), the drone's flight time, F, changes for each instance, where $F = \mathcal{E}_r e_{\max}$; $e_{\max}$ represents the longest vehicle arc in the network, and $\mathcal{E}_r \in \{0.2, 0.4, 0.6, 1\}$. Setting $\mathcal{E}_r = 1$ extends the flight time such that any node may be served via drone cycle or a non-cyclic drone flight from any other node in the network (so long as the drone moves at least twice as fast as the vehicle).

Instance Generation Schema

In addition to the benchmark (Agatz et al., 2018) instances, this study introduces a schema that may be used to generate instances designed specifically to be conducive to drone (single and multi-) cycles. Customers are located in square clusters, which closely mimic tranches of houses in the grids typical in urban environments. Details of the **Grid** instance generation scheme are available in Appendix C, and solutions to a set of small instances (having up to 20 nodes) including objective values and the sequence of customers visited by both the vehicle and drone are in a public repository.[4]

5.3. Performance on small benchmark instances with randomly distributed customers

To contextualize the impact of drone cycles, this section presents results for instances having 20 nodes, with the smallest maximal flying time of $\mathcal{E}_r = 0.2$. In what follows, the performance of four approaches are compared. The top two rows of Table 2 show exact approaches: the model proposed for Phase II (Model TSPD); as well as the compact formulation published by Roberti and Ruthmair (2021). The bottom two rows show heuristic approaches: the simplified formulation proposed for Phase I (Model TSPD0); and the proposed algorithm, DIP. Each approach was limited to one hour of CPU time. $\Delta_{\mu}^{end}(\%)$ shows the average deviation of the proposed approach from the result yielded by DIP at the termination of both. For example, Table 2 shows that Model TSPD produced objective values that were 0.2% better than those yielded by DIP at the termination of both approaches. $\Delta_{\mu}^{180}(\%)$ reports the same deviation after 180 s of CPU time. **CPU (s)** reports the average computational effort of each approach, **Opt.** reports the number of instances with optimal solutions, and **MIP Gap (%)** reports the percentage deviation between the best-known upper and lower bounds for the exact approaches.

Model TSPD compares favorably with the compact formulation proposed by Roberti and Ruthmair (2021) in terms of solution quality and computational effort. Table 2 also shows the advantage of the simplifications proposed in Section 4.3.2 for Model TSPD0, with computational effort of just over 100 s (as compared with more than 1400 s for Model TSPD) and deviations of less than 1.2% on average from the proposed exact solution approach. Although DIP requires slightly greater computational effort than Model TSPD0, the former achieved 5/10 optimal solutions, while the latter achieved none. DIP's key advantage is to deploy the Phase I formulation effectively, then to further intensify the search for a solution in Phase II. Moreover, as the next section shows, the advantage of using DIP grows with the size of instances and the maximal flight time of the drone.

Table 3
Summary of DIP's performance on benchmark instances.

n	$\mathcal{E}_r$	Imp.	Δ_{μ} (%)	Δ_{σ} (%)	CPU (s)
20	0.2	10/10	−12.6	2.9	123.2
	0.4	10/10	−18.6	5.3	154.8
	0.6	10/10	−11.0	5.6	167.9
	1.0	10/10	−5.7	3.2	176.9
Summary		40/40	−11.9	6.3	155.7
50	0.2	10/10	−21.4	3.7	383.5
	0.4	10/10	−7.9	1.7	477.2
	0.6	10/10	−7.3	1.3	466.4
	1.0	10/10	−6.8	2.4	565.6
Summary		40/40	−10.8	6.6	473.2
100	0.2	10/10	−11.4	6.4	512.8
	0.4	8/10	−5.4	6.9	529.4
	0.6	7/10	−4.9	6.9	578.6
	1.0	8/10	−5.1	6.9	490.4
Summary		33/40	−6.7	7.1	527.8

5.4. Performance on larger benchmark instances with randomly distributed customers

As part of a study investigating the VRP-D (summarized in Section 2.1), Schermer et al. (2019) investigated settings with a single vehicle and a single drone, which correspond to a TSP-D with cycles. The authors of that study graciously agreed to provide detailed results for the benchmark Uniform instances (up to 100 nodes) published by Agatz et al. (2018). The following notation is used to assess the performance of DIP on the same instances:

- $\Delta\%$: The percentage deviation of Z^*_{DIP}, the solution produced by DIP, over Z^*_{S}, the best solution produced by the approach of Schermer et al. (2019) (i.e. $100(\frac{Z^*_{\text{DIP}} - Z^*_{\text{S}}}{Z^*_{\text{S}}})$).
- Δ_{μ}: The mean of $\Delta\%$ over 10 instances of the same size.
- Δ_{σ}: The standard deviation of $\Delta\%$ over 10 instances of the same size.
- **Imp.:** The count of instances for which Z^*_{DIP} improved upon Z^*_{S}.
- **CPU (s):** The mean computational effort in CPU seconds over 10 instances of the same size.

The results in Table 3 indicate DIP yielded consistent improvements over the approach of Schermer et al. (2019) across instance size and values of $\mathcal{E}_r$. Overall, DIP improved upon the results of Schermer et al. (2019) for 113/120 instances, by an average of 11.9%, 10.8%, and 6.7% for instances having 20, 50, and 100 nodes, respectively. DIP yielded improvements on 80/80 instances tested having 20 or 50 nodes.

Across instance size, larger values of $\mathcal{E}_r$ generally yielded smaller improvements in solution quality as shown in Fig. 8, which plots the differential percentage ($\Delta\%$) for each instance in the test-bed. For example, with $n = 50$, the average value of Δ was 20.9% under $\mathcal{E}_r = 0.2$, while the remaining values of $\mathcal{E}_r$ yielded average differentials no higher than 8%. Larger $\mathcal{E}_r$ values imply that many more customers are within the drone's maximum flight time, which may adversely affect the efficacy of DIP. The valid inequalities in each formulation based on Lemma 1 had a favorable impact on DIP's performance for instances with unlimited flight time (i.e. $\mathcal{E}_r = 1$): absent the inequalities, DIP improved upon 26/30 instances, as compared with 28/30 instances improved upon with the inequalities active under comparable computational effort.

Although Table 3 shows Δ_{σ} values up to nearly 7%, Fig. 8 highlights the consistency and efficacy of DIP. Note the standard deviation in the Summary rows in Table 3 are the standard deviations of all 40

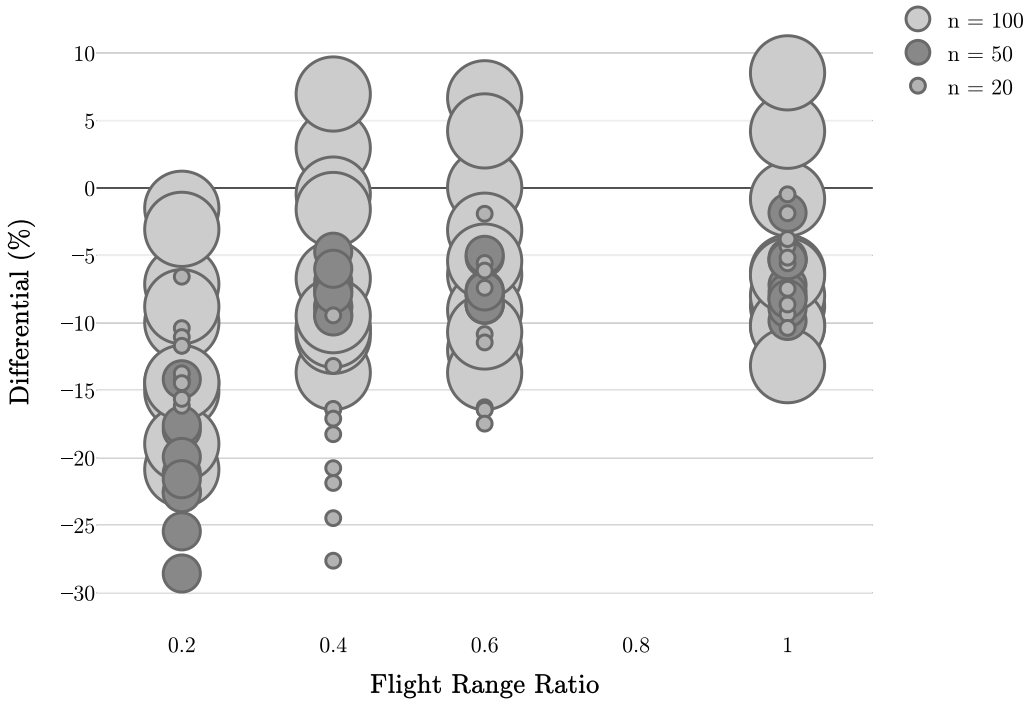


Fig. 8. DIP Improvements by Flight Ratio, Instance Size.

instances for a given n value. Excepting 7/120 instances with positive Δ values, DIP yielded solutions that improved upon those of Schermer et al. (2019) with Δ values below – 10%, and occasionally below – 20%. Just as important, DIP is deterministic: repeated runs of DIP yield identical solutions whereas the approach of Schermer et al. (2019) depends on randomized procedures that yield variable results. Note that Δ compares a unique DIP solution against the best of *five* different solutions reported by Schermer et al. (2019). Comparing against the average (115/120) or worst-case (118/120) solutions from all the solutions reported by Schermer et al. (2019) shows greater improvements by DIP.

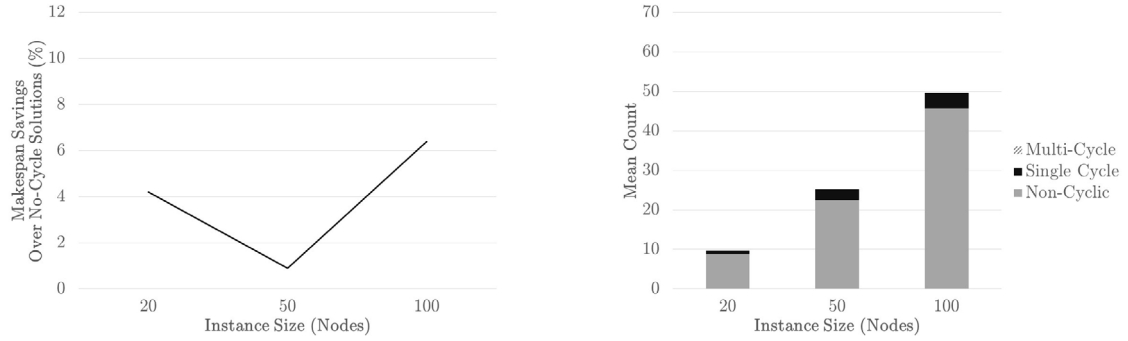
Overall, the computational effort exerted by DIP remained below 10 min on average across all instances. The growth of the computational effort across values of n and $\mathcal{E}_r$ hints at the advantages of DIP. Between $n = 20$ and $n = 50$, DIP's computational effort more than doubled on average; but between $n = 50$ and $n = 100$, the computational effort grew from roughly 475 s to about 530 s. This marginal change was driven by DIP's termination criteria: specifically, the number of iterations in each phase of the algorithm. On average, Phase I consumed 19 iterations for $n = 50$, and 39 iterations for $n = 100$. Nevertheless, Phase I required about 290 s on average for both $n = 50$ and $n = 100$. Moreover, in both cases DIP achieved 70% of the overall improvement over the Schermer et al. (2019) solutions by the end of Phase I. This suggests that the termination criteria used in this study allowed for an effective exploration of the solution space across the test bed. Finally, the importance of Phase II and its termination criteria is clear with $\mathcal{E}_r = 1.0$. Under these more challenging settings, Phase I often required greater computational effort. DIP managed a fuller exploration of the solution space in Phase II to yield high-quality solutions without adversely affecting computational effort.

5.5. Drone cycle sensitivity analysis

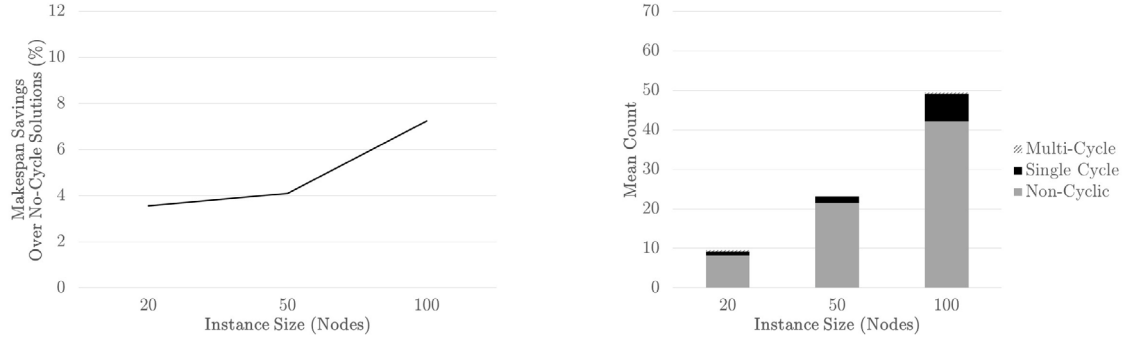
With a robust platform in DIP to solve TSP-D instances, this section highlights the impact of drone single- and multi-cycles across different customer topographies. For the remainder of this section, the benchmark Uniform and Single Center benchmark instances from Schermer et al. (2019) are referred to as **Random** and **Clustered** customer topographies, respectively, to better capture their customer layouts. The generation scheme of each topography is detailed in Section 5.4.

For each topography, the left-hand side of Fig. 9 charts the mean objective value over a set of instances on a relative scale: it compares the savings in the objective values obtained by DIP when cycles are permitted against the objective values obtained by DIP on the same instance when cycles are disallowed. Thus the displayed values are the mean percent savings across topographies and instance sizes when drone multi-cycles are permitted. On the right-hand side of Fig. 9, the mean number of drone flights (broken down by single- versus multi-cycle versus non-cyclic flights) is shown. Note that for all instances, DIP deployed the same search parameter settings outlined in Section 5.1. And to facilitate drone cycles, the instances were tested with unlimited maximum flight time (i.e. $\mathcal{E}_r = 1$), and $\alpha = 4$, such that the drone moved four times as quickly as the vehicle. This analysis suggests:

- **Drone Cycle Robustness:** Across all customer topographies and instance sizes tested, drone cycles consistently yielded improvements in the makespan of the TSP-D ranging from 1%–8%. These savings are directly attributable to the frequency of drone cycles, which are summarized for each topography. Empirically, only single cycles featured in the **Random** and **Clustered** topographies. In limited testing of instances (having 20 nodes) generated using the **Grid** schema, our results also show that it is possible to create conditions conducive to multi-cycles. Nevertheless, our results indicate multi-cycles are rare in the benchmark instances we have tested. Finally, the multi-cycles in this analysis featured at most two cyclic drone visits. Multi-cycles with more customers are of course feasible. But across all instances explored in this study, multi-cycles proved rare, and were limited to two cyclic drone flights at most.
- **Cycle Frequency versus Makespan Savings:** Across topographies, the magnitude of savings associated with cycles varied along with the frequency of the cycles. For example, the **Grid** topography features the greatest number of cycles overall - up to five times as many cycles for $N = 20$ when compared with



(a) Random Customer Topography Analysis



(b) Clustered Customer Topography Analysis

Fig. 9. A summary of the impact of drone cycles across customer topographies.

the Random topography, for example. But the magnitude of improvements to the makespan of the instances did not match that scale: for $N = 20$, there was an approximate improvement of 4% (**Random**) versus 3% (**Grid**) across topographies. Even as faster drone speeds facilitate cycles, those faster speeds also diminish the savings that may be attributed to the cycles.

- **Negligible Impact of Vehicle Idling:** Although a drone cycle may improve the makespan of a TSP-D tour, cycles require the vehicle to idle while the drone is in flight. Across all topographies and instance sizes, these results showed a negligible increase in the vehicle's idling time. Specifically, the vehicle's idling time may be expressed as a percentage of the makespan of the tour. As an example, without drone cycles, instances having 100 nodes under the Random topography had an idling to makespan ratio of 1.6%. With cycles, that ratio was 2.2%. This ratio may indicate that the idling time of the vehicle increases as expected with cycles. But just as importantly, the ratio shows that the vehicle's idle time still constitutes a small share of the total makespan. Across all topographies and instance sizes, the idling to makespan ratio without cycles was 1.2% versus 3.8% with cycles allowed.

6. Conclusion

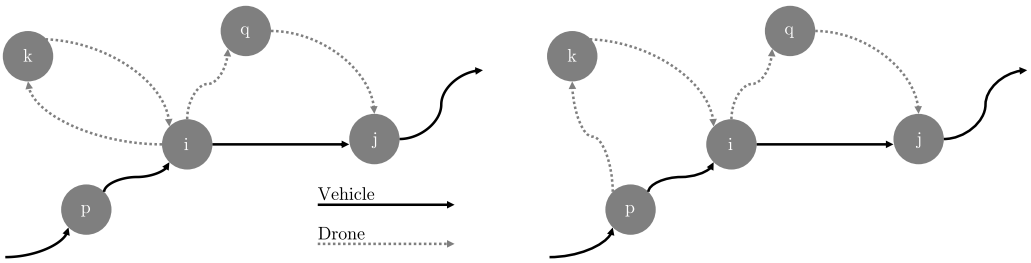
This study introduces a variable neighborhood search heuristic denoted DIP, underpinned by two distinct MIP formulations designed to generate high-quality solutions to the TSP-D. Phase I of DIP generates rapid improvements in a starting solution by temporarily disallowing vehicle visits to intermediate customer between the nodes where the drone is launched and recollected. Phase II restores that assumption to yield a deeper, yet computationally more onerous, exploration of the feasible space. To avoid entrapment at local optima, the proposed methodology is further enhanced through the use of diversification schemes that vary the size of neighborhoods and the re-optimization effort expended therein. On a set of benchmark TSP-D instances, DIP yields, to our knowledge, the best-known solutions to 113/120 instances having up to 100 nodes, with comparable computational performance to existing approaches.

In contrast with classical routing problems that allow at most one visit to each customer, the TSP-D is complicated by cyclic drone flights, in which the drone may launch, deliver to a customer, then land at the same node from which it launched. DIP permits decision-makers to modulate this feature: to disallow cycles; to allow single cycles only; or to allow both single and multi-cycles. On a variety of customer topographies, DIP also revealed the consistent presence of drone cycles, and their potential to improve the makespan of TSP-D tours by up to 8% versus disallowing them in the solution. Our analysis also shed light on the empirical preponderance of single cycles as compared with multi-cycles, as well as a negligible impact on vehicle idling time as a result of drone cycles. We hope this insight advances the TSP-D literature since the relevance of (single or multiple) drone cycles was not clearly settled in this fast-growing stream of research.

A key assumption in the classical TSP-D literature is that customer locations serve as launch and landing nodes for the drone. In future studies, it may be interesting to explore networks comprised of customer nodes as well as distinct non-residential rendezvous locations used exclusively for the launch and retrieval of delivery drones. It also may be insightful to investigate the customer dispersion patterns and operational conditions which maximize the logistical benefits of cyclic drone flights.

CRediT authorship contribution statement

Amro M. El-Adle: Conceptualization, Methodology, Software, Validation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Ahmed Ghoniem:** Conceptualization, Methodology, Validation, Formal analysis, Resources, Writing – original draft, Writing – review & editing, Visualization. **Mohamed Haouari:** Conceptualization, Methodology, Validation, Formal analysis, Writing – original draft, Writing – review & editing, Visualization.



(a) Scenario 1 (b) Scenario 2

Fig. B.10. A comparison of feasible drone flights involving cycles.

Data availability

The manuscript includes a link to a repository with the data from the article.

Appendix A. Tightened M values for model TSPD

The scalars $M, M^1, M^2, \ldots, M^6$, introduced in Section 3.2 for Model TSPD, may be validly set as follows (see El-Adle et al., 2021 for detailed derivations):

$$M_j^1 = \max(0, \max_{\substack{i\in V\\ f_{ij}\le F}} f_{ij}, \max_{\substack{k\in V\\ f_{jk}\le F}} f_{jk}),$$

$$M_{ij}^2 = \begin{cases} \tau_{0j} - f_{0j}, & i = 0, \\ \tau_{i0} - f_{i0}, & j = 0, \\ (C_{\max} - f_{i0} + t_{ij}) - f_{0j}, & \text{otherwise}, \end{cases}$$

$$M_{ij}^3 = \begin{cases} 0, & i = 0 \text{ or } j = 0, \\ (C_{\max} - f_{j0} + f_{ij}) - f_{0j}, & \text{otherwise}, \end{cases}$$

$$M_{ij}^4 = \begin{cases} 0, & i = 0, \\ (C_{\max} - f_{i0} + f_{ij}) - f_{0j}, & \text{otherwise}, \end{cases}$$

$$M_{ij}^5 = \begin{cases} (C_{\max} - f_{j0}) - \tau_{0j}, & i = 0, \\ (C_{\max} - f_{j0}) & \text{otherwise}, \end{cases}$$

$$M_i^6 = C_{\max} - \tau_{i0}, \qquad \forall i \in V^*,$$

$$M = C_{\max} - \min_{\substack{i,j\in V^*:\\ f_{ij}+f_{j0}\le F}} \{f_{j0}, \tau_{j0}\}.$$

Appendix B. Lemma 1 Proof

Lemma 1: *Given unlimited flight time, if there is a drone cycle from node i to node k, then there must be one non-cyclic drone flight landing at i and one non-cyclic drone flight departing from i.*

Proof. We proceed to prove Lemma 1 by contradiction. As per the settings in the computational study in Section 5.2, assume symmetrical travel times between nodes, and that the drone's flight time may be expressed as a ratio of the vehicle's travel time (i.e. $\alpha f_{ij} = \tau_{ij}$):

- Assume there exists an optimal solution, illustrated as Scenario 1 in Fig. B.10, featuring a drone cycle from node i to node k absent a non-cyclic drone flight landing at node i. We derive from Scenario 1 a new solution, Scenario 2, that is similar except that node k is now served by a non-cyclic drone flight that lands at i after launching from p. By inspection, Scenario 2 is feasible because the drone has unlimited flight time.
  Assuming Scenario 1 is optimal, then the vehicle's departure time from node i must be earlier or equal to the departure time from the same node in Scenario 2. Comparing, we have

  $$\tau_{pi} + 2f_{ik} \le \max(\tau_{pi}, f_{pk} + f_{ki}).$$

  If $\tau_{pi} \ge f_{pk} + f_{ki}$, then by inspection Scenario 1 yields a longer completion time than Scenario 2. Alternatively, suppose $\tau_{pi} < f_{pk} + f_{ki}$, in which case Scenario 1 is optimal if

  $$\tau_{pi} + 2f_{ik} \le f_{pk} + f_{ki}.$$

  Since the drone flights are symmetric,

  $$\tau_{pi} + f_{ik} < f_{pk}.$$

  Furthermore, since $\alpha f_{ij} = \tau_{ij}$, 全部转化成无人机的飞行时间再用三角不等式

  $$\tau_{pi} + \alpha^{-1}\tau_{ik} \le \alpha^{-1}\tau_{pk}.$$

  By virtue of the triangle inequality,

  $$\tau_{pi} + \alpha^{-1}\tau_{ik} \le \alpha^{-1}(\tau_{pi} + \tau_{ik}).$$

  But this is impossible, since by assumption $\alpha^{-1} < 1$.
- Using a similar argument, the symmetric case in which a solution featuring a drone cycle from node i to node k absent a non-cyclic drone flight departing i must be sub-optimal.

By contradiction, we have shown that when a drone cycle launches from node i, there must be both at least one non-cyclic drone flight launching from i, and at least one non-cyclic drone flight landing at i. □

Appendix C. Grid instance generation scheme

The **Grid** topography schema introduced in this study is designed specifically to generate instances featuring drone single and multi-cycles. The building block of each instance is a tranche of nodes designed to mimic an urban city block. The nodes within the tranche are distributed into three peripheries, as illustrated in Fig. C.11. The inner periphery (the nodes colored in gray) are nodes to be visited by drone multi-cycles; the second periphery features the corresponding rendezvous nodes to be visited by the vehicle (colored in black). Each **Grid** instance also features a third periphery of nodes, designed for single-cycle drone flights (colored in dark gray).

The instances analyzed in this study featured 20 nodes. These instances are comprised of a single tranche having 20 nodes, with the depot at the origin (as illustrated in Fig. C.11, with the depot represented by a square node). To generate larger instances having 50 and 100 nodes, several tranches may be stitched together, with each tranche featuring 25 nodes. The tranches may be arranged in various orientations, with the depot chosen from a particular tranche at random.

Finally, note that the inter-periphery distance (i.e. the distance between the nodes in each periphery) and the flight time of the drone are calibrated according to the drone speed in order to allow multi-cycles. The sensitivity analysis in Section 5.5 uses $\alpha = 4$, and we set $F = 1$. The precise inter-periphery distances are chosen at random for each instance.

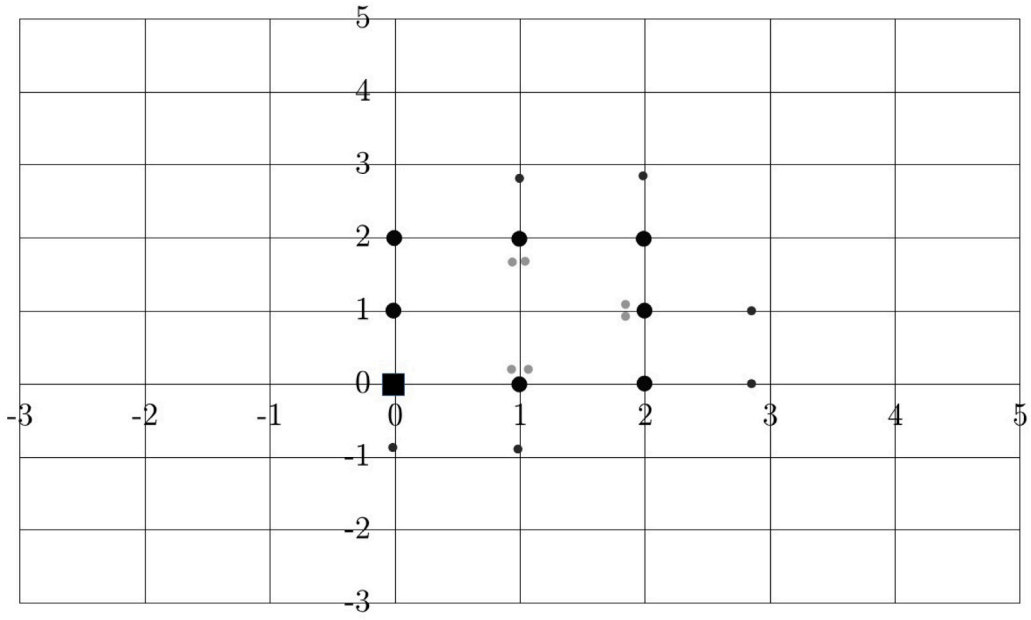


Fig. C.11. An illustration of a sample grid instance with a single tranche.

References

Adler, A., 2020. Workhorse perfecting HorseFly truck-based drone delivery - Freight-Waves. Freight Waves URL: https://www.freightwaves.com/news/workhorse-perfecting-truck-based-autonomous-drone-delivery.

Agatz, N., Bouman, P., Schmidt, M., 2018. Optimization approaches for the traveling salesman problem with drone. Transp. Sci. 52, 965–981. http://dx.doi.org/10.1287/trsc.2017.0791.

Boccia, M., Masone, A., Sforza, A., Sterle, C., 2021. An exact approach for a variant of the FS-TSP. 52, Elsevier B.V., pp. 51–58. http://dx.doi.org/10.1016/j.trpro.2021.01.008,

Bouman, P., Agatz, N., Schmidt, M., 2018. Dynamic programming approaches for the traveling salesman problem with drone. Networks 72, 528–542. http://dx.doi.org/10.1002/net.21864.

Boysen, N., Briskorn, D., Fedtke, S., Schwerdfeger, S., 2018. Drone delivery from trucks: Drone scheduling for given truck routes. Networks 72, 506–527. http://dx.doi.org/10.1002/net.21847.

Boysen, N., Fedtke, S., Schwerdfeger, S., 2021. Last-mile delivery concepts: a survey from an operational research perspective. OR Spectrum 43, 1–58. http://dx.doi.org/10.1007/s00291-020-00607-8.

Carlsson, J.G., Song, S., 2018. Coordinated logistics with a truck and a drone. Manage. Sci. 64, 4052–4069. http://dx.doi.org/10.1287/mnsc.2017.2824.

Chung, S.H., Sah, B., Lee, J., 2020. Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. Comput. Oper. Res. 123, 105004. http://dx.doi.org/10.1016/j.cor.2020.105004.

Dell'Amico, M., Montemanni, R., Novellani, S., 2021. Drone-assisted deliveries: new formulations for the flying sidekick traveling salesman problem. Optim. Lett. 15, 1617–1648. http://dx.doi.org/10.1007/s11590-019-01492-z.

El-Adle, A.M., Ghoniem, A., Haouari, M., 2021. Parcel delivery by vehicle and drone. J. Oper. Res. Soc. 72, 398–416. http://dx.doi.org/10.1080/01605682.2019.1671156.

de Freitas, J.C., Penna, P.H.V., 2020. A variable neighborhood search for flying sidekick traveling salesman problem. Int. Trans. Oper. Res. 27, 267–290. http://dx.doi.org/10.1111/itor.12671.

Macrina, G., Pugliese, L.D.P., Guerriero, F., Laporte, G., 2020. Drone-aided routing: A literature review. Transp. Res. C 120, 102762. http://dx.doi.org/10.1016/j.trc.2020.102762.

Moring, C., 2019. USPS joins the drone delivery domain with RFI for services [Press release]. URL: https://www.crowell.com/newsevents/alertsnewsletters/all/usps-joins-the-drone-delivery-domain-with-rfi-for-services.

Murray, C.C., Chu, A.G., 2015. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transp. Res. C 54, 86–109. http://dx.doi.org/10.1016/j.trc.2015.03.005.

Otto, A., Agatz, N., Campbell, J., Golden, B., Pesch, E., 2018. Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. Networks 72, 411–458. http://dx.doi.org/10.1002/net.21818.

Poikonen, S., Golden, B., Wasil, E.A., 2019. A branch-and-bound approach to the traveling salesman problem with a drone. INFORMS J. Comput. 31, 335–346. http://dx.doi.org/10.1287/ijoc.2018.0826.

Raj, R., Murray, C., 2020. The multiple flying sidekicks traveling salesman problem with variable drone speeds. Transp. Res. C 120, 102813. http://dx.doi.org/10.1016/j.trc.2020.102813.

Roberti, R., Ruthmair, M., 2021. Exact methods for the traveling salesman problem with drone. Transp. Sci. 55, 315–335. http://dx.doi.org/10.1287/TRSC.2020.1017.

Schermer, D., Moeini, M., Wendt, O., 2019. A matheuristic for the vehicle routing problem with drones and its variants. Transp. Res. C 106, 166–204. http://dx.doi.org/10.1016/j.trc.2019.06.016.

Schermer, D., Moeini, M., Wendt, O., 2020. A branch-and-cut approach and alternative formulations for the traveling salesman problem with drone. Networks 76, 164–186. http://dx.doi.org/10.1002/net.21958.

Tang, C.S., Veelenturf, L.P., 2019. The strategic role of logistics in the industry 4.0 era. Transp. Res. E Logist. Transp. Rev. 129, 1–11. http://dx.doi.org/10.1016/j.tre.2019.06.004.

Vásquez, S.A., Angulo, G., Klapp, M.A., 2021. An exact solution method for the TSP with Drone based on decomposition. Comput. Oper. Res. 127, http://dx.doi.org/10.1016/j.cor.2020.105127.

Yurek, E.E., Ozmutlu, H.C., 2018. A decomposition-based iterative optimization algorithm for traveling salesman problem with drone. Transp. Res. C 91, 249–262. http://dx.doi.org/10.1016/j.trc.2018.04.009.