Computers & Industrial Engineering 129 (2019) 14–30

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Multiple traveling salesman problem with drones: Mathematical model and heuristic approach

Patchara Kitjacharoenchai[a,*], Mario Ventresca[a], Mohammad Moshref-Javadi[b], Seokcheon Lee[a], Jose M.A. Tanchoco[a], Patrick A. Brunese[a]

[a] School of Industrial Engineering, Purdue University, West Lafayette, IN, United States
[b] Center for Transportation and Logistics, Massachusetts Institute of Technology, MA, United States

ARTICLE INFO

Keywords:
Drone delivery
Optimization
Unmanned aerial vehicles
Heuristic
Transportation
Routing

ABSTRACT

E-commerce and retail companies are seeking ways to cut delivery times and costs by exploring opportunities to use drones for making last mile delivery deliveries. This paper addresses the delivery concept of a truck-drone combination along with the idea of allowing autonomous drones to fly from delivery trucks, make deliveries, and fly to any available delivery truck nearby. We present a mixed integer programming (MIP) formulation that captures this scenario with the objective of minimizing the arrival time of both trucks and drones at the depot after completing the deliveries. A new algorithm based on insertion heuristics is also developed to solve large sized problems containing up to a hundred locations. Experiments are conducted to compare the MIP solutions with those obtained from different models with single truck, multiple truck and a single truck and drone system, as well as test the performance of the proposed algorithm. The numerical results demonstrate the potential operational gain when implementing the proposed drone delivery system compared to the conventional truck alone or single truck/drone delivery system.

1. Introduction and literature review

Technological innovations in last mile delivery of products can determine the profitability of companies, both traditional players and new entrants in this growing market (Adams, 2017; Balcik, Beamon, & Smilowitz, 2008; Lee, Chen, & Gillai, 2016). Due to high competition in the market, customers are likely to buy services and products from a company with fast, flexible, and free delivery. Consequently, there has been some recent attention on the use of drones, robots, and driverless cars to deliver items to households (Patil, 2016; Rao, Gopi, & Maione, 2016). Many technology companies such as Google and Amazon, as well as traditional logistics providers such as UPS, USPS, DHL and FedEx have been experimenting with drone delivery, with the goal of cutting costs and providing cheaper, yet faster and more efficient service (Ungerleider, 2016). In 2013, Jeff Bezos, CEO of Amazon, stated that Amazon will be using drones for 30-minute delivery time for its customers (Gross, 2013; Rose, 2013; Smith, 2013). Since then, Amazon has developed a project called “Prime Air” that performed a public drone delivery demonstration at an Amazon-hosted conference in Palm Springs, California in March 2017 (Glaser, 2017). Besides Amazon, Google has initiated a project called “Wing” to deliver supply relief kits such as medicine, food, and water after disasters. The company uses Project Wing’s software to control multiple drones at the same time (Vincent, 2017). DHL also has launched a drone delivery service to ship items, such as medication and other urgently needed goods to people located on an island in Germany’s North Sea where most traditional delivery options are not available (Hern, 2014). Other successful drone delivery events include 7-Eleven and Flirtey autonomous drone delivery, Domino drone delivery, and HorseFly Drone, in which the drone is launched from the delivery truck to make a package delivery at one location while the truck makes another delivery simultaneously (Perez & Kolodny, 2017).

Drones seem to be a great fit for the last mile delivery due to their high travel speed and ability to access areas where no other mode of transportation is available. Most drones developed by Google, Amazon, and DHL fly at a speed of 30–40 miles (48–64 km) per hour with a flight range of 10–30 miles (16–48 km) (Heath, 2015). Since drones are not restricted by road infrastructure, they can deliver packages faster than a truck from the same location. While trucks must travel on streets and avenues using rectilinear distance, drones can travel through areas like mountains, jungles, and rivers with relative ease, and take shorter routes (Bamburry, 2015; Berg, 1991; Goel & Kok, 2012). However,

* Corresponding author.
E-mail address: pkitjach@purdue.edu (P. Kitjacharoenchai).

https://doi.org/10.1016/j.cie.2019.01.020
Received 14 June 2018; Received in revised form 6 January 2019; Accepted 8 January 2019
Available online 10 January 2019
0360-8352/ Published by Elsevier Ltd.

drones' disadvantages are mainly their small shipping capacity and their battery run time; e.g., the Amazon Prime drone can carry approximately 5 lb and their batteries must be swapped or recharged after 30 min (Brar, Rabbat, Raithatha, Runcie, & Yu, 2015). In addition to this limitation, the regulation of commercial drone deliveries is still evolving, and companies cannot experiment with their drone deliveries in the U.S. unless given permission by the FAA (Dorr & Duquette, 2016). Regarding air transportation management, the FAA has recently partnered with NASA to form an Unmanned Aircraft Systems (UAS) traffic management platform, essentially an air traffic control system for drones. In the meantime, e-commerce and logistics companies have been testing their drone delivery programs in Canada, the U.K., and the Netherlands, which have less restrictions than in the U.S. (Bonnington, 2017). To manage safety, drone manufacturers and software providers are quickly developing technologies like geo-fencing and collision avoidance to ensure safety when drones fly autonomously. The increase in drone technology development is pushing governments to create new regulations that balance safety and innovation (Meola, 2016). In the most recent update, the White House announced the launch of a three-year pilot program to create "innovation zones" to evaluate commercial drone operations flying at night, flying over people, and flying outside a pilot's line of sight (McFarland, 2017). This program allows logistics companies to test their drone delivery service without any restriction. With the supports from different public sectors to finalize the regulation and air transportation platform, commercial drone delivery should be ready to be implemented in the near future.

In the field of operations management, we are aware of many related works. In 2015, Murray and Chu (2015) introduced a new type of TSP problem called the "Flying Sidekick Traveling Salesman Problem" (FSTSP). The authors proposed the idea of having a drone attached to the top of a truck that can make a delivery while the truck is making another delivery simultaneously. Once the drone finishes making a delivery, it needs to fly back to the truck at the current delivery location or along its route to the next delivery location. Due to the problem complexity, FSTSP considers only one truck and one drone. This leaves some room for an improvement to include more than one unit of each vehicle type in the model. The FSTSP model is inspired by the Vehicle Routing Problem (VRP) with synchronization constraints (Drexl, 2012). The primary problem in this class is VRPTT which was originally motivated by milk farms delivery using a heterogeneous fleet of vehicles to transport the milk from the depot to the customers. In this problem, some customers are required to be visited by be visited by the small vehicle (lorries) which has to be dispatched from the trailer at the certain location. Due to the multiple interdependencies of this problem, there is currently no successful exact solution algorithm for this problem and only common heuristic solution approaches (local search) can return acceptable solution.

Besides FSTSP, there are well studied drone-truck routing problems in the past literature. Agatz, Bouman, and Schmidt (2018) developed a new TSP-D, in which truck and drone are making deliveries in parallel. The authors presented the new MIP model and proved the theoretical worst-case approximation bound by solving 12 nodes problems exactly. They also developed heuristic approach following a route first, cluster-second principle with local search afterwards. Once building the TSP truck-only route, the authors used two methods to split the truck route to include drone route: a greedy algorithm and an exact partition algorithm. The experiment showed that exact partition heuristic outperforms the greedy heuristic when testing with the larger instances more than 10 nodes. Similar to FSTSP, the TSP-D considers an operation with one truck and one drone with the potential extension to solve the problems with multiple trucks and drones. The same authors recently developed an exact solution for the TSP-D using the dynamic programming approaches based on the well-known Bellman-Held-Karp dynamic programming algorithm (Bouman, Agatz, & Schmidt, 2018). This approach was able to find the optimal solution for problem instances 10 vertices much faster than the solution from the integer programming approach presented by Agatz et al. (2018). Their proposed approach can also successfully solve larger problems than with the mathematical programming approaches. The authors did numerical experiments by restricting the number of locations the truck can visit in each instance while the drone is away. This lowers the computational times while still maintaining the overall solution quality. They addressed the limitation of the model as it only considers simple operations with at most one drone. The future research can incorporate multiple drones where each drone starts and ends at different nodes.

Ponza (2015) examined the FSTSP in detail and applied the simulated annealing technique to search for good solutions. This approach is shown to find good solutions for the tested instances within the reasonably linear time. He also studied the changes in objective values when modifying the drone's endurance, drone's top speed and the different number of iterations. The results from the experiment demonstrated the positive impact on the solution quality as the time it takes to complete the tour decreases once increasing these parameters. Ha, Deville, Pham, and Hà (2018) proposed the min-cost TSP-D which represents the same operation as FSTSP with the objective to minimize the total transportation cost. This work included the model and the two algorithms: TSP-LS based on local search and a Greedy Randomized Adaptive Search Procedure (GRASP) to solve a new TSP-D problem. The numerical experiments showed that the drone's utility rate is higher when minimizing the cost in the objective function rather than delivery time. Ha reiterated Murray and Agatz 's point to focus the future research on multiple trucks and multiple drones type of operation and the development of efficient metaheuristics.

In a similar theme, Ferrandez, Harbison, Weber, Sturges, and Rich (2016) proposed an optimization model of a truck-drone system in tandem delivery networks by using the K-means algorithms to find the most efficient launch locations as well as using a genetic algorithm to assign the truck route between those launch locations. The drones are, however, not restricted by the flight endurance. The authors did some experiments to investigate the impact of various relative velocities with the results showing that the drone's speed should be at least twice as fast as the truck's speed to significantly reduce the route time. The authors also addressed the benefits of using multiple drones (per vehicle) in reducing the energy consumption. Mathew, Smith, and Waslander (2015) proposed a new drone and truck problem called the Heterogeneous Delivery Problem (HDP). The goal is to seek an optimal delivery route in an urban location with the objective to minimize the total cost of deliveries, which consists the cost of truck travels, the cost of drone travels, and the cost of simultaneous truck and drone travels. To solve the HDP, the authors suggested reducing the HDP to Generalized Traveling Salesman Problem (GTSP), which can be solved using a variety of solvers in existing literature. Mathew et al. (2015) also proposed a special case of the HDP called the Multiple Warehouse Delivery Problem (MWDP) consisting of a single vehicle and multiple static warehouses. Two algorithms were introduced to solve MWDP: one using an alternative transformation to the Traveling Salesman Problem (TSP) and another one using an exact polynomial time exact algorithm to obtain an optimal route.

In a recent work, Campbell, Sweeney, and Zhang (2017) proposed continuous approximation (CA) models to obtain the optimal number of truck and drone deliveries per route, the optimal number of drones per truck and the total cost of operation in the hybrid truck-drone delivery problem. The authors implement their models to compare the expected delivery cost for different delivery systems which helps identifying when and where the implementation of truck-drone delivery is likely to be beneficial. The results demonstrated that the truck drone hybrid delivery model has the potential benefits to provide substantial cost savings by allowing demand to be covered with fewer routes and by reducing the route costs. These benefits strongly depend on the operating costs per mile for both trucks and drones, the stop costs, and the spatial density of customers. Similarly, Carlsson and Song (2017) used a CA technique to determine the best set of parameters that results in the

minimum completion of all truck-drone deliveries in the Euclidean plane. The authors discovered that the benefit of using a drone along with truck is proportional to the square root of the relative velocity between truck and drone. The potential benefit of truck drone operation also depends on the instance features.

Dorling, Heinrichs, Messier, and Magierowski (2017) in another perspective, proposed the Drone Delivery Problem (DDP). Two VRP based models were presented in this study: the Minimum-Time-DDP (MT-DDP) and the Minimum-Cost-DDP (MC-DDP). The total delivery cost is subject to a delivery time limit in the MT-DDP and the overall delivery time is subject to a budget constraint in the MC-DDP. In both models, the author provided the mathematical model for a cost function that considers energy consumption and drone reuse. Both problems can be solved by a MILP solver and a Simulated Annealing (SA) algorithm, which the SA can reach near-optimal solutions for small-sized problems.

Wang, Poikonen, and Golden (2016) provided worst-case analyses for the vehicle routing problem with drones (VRPD) problem by studying how the two parameters—the number of drones per truck and the speed of the drones—can affect the maximum savings from using drones. The authors introduced several upper bounds on the amount of time saved by using drones compared to using truck alone in the route. The upper bounds can be obtained by investigating how the relative velocity of the drones compared to the trucks and the number of drones per truck affect the solution quality. Poikonen, Wang, and Golden (2017) further continued their work of VRPD (Wang et al., 2016) by expanding their previous results to fit with the wide variety of circumstance including the limited battery life, different distance metrics, and operational expenditures of deploying drones and trucks. They established the bounds that relate to the Vehicle Routing Problem (VRP) and the min-max Close Enough Traveling Salesman Problem (CETSP). Lastly, Daknama and Kraus (2017) proposed the Vehicle Routing with Drones (VRD) problem that shares most of the features of the VRPD (Wang et al., 2016). The difference is that Daknama and Kraus allow a drone to be launched and retrieved by different trucks, which is one of the unique feature in our work as well. The authors also developed a heuristic based on two nested local search procedures to solve the VRD problem. Two important findings from this work: using drones along with trucks significantly reduce the delivery time and the increase in the number of drones generally leads to the better solution.

Main Contribution

In this paper, we propose a new MIP model and a heuristic algorithm to solve a new problem, the Multiple Traveling Salesman with Drones (mTSPD) problem. The main contributions of this paper are the following:

1. We introduce a new variant of the Traveling Salesman Problem with Drones in which multiple trucks and drones are deployed to make deliveries. We call the problem “Multiple Traveling Salesman with Drones” (mTSPD). The model is based on the Multiple Traveling Salesman Problem (Bektas, 2006) with the mathematical formulation adapted from the FSTSP model (Murray & Chu, 2015).
2. We develop a new heuristic called “Adaptive Insertion Heuristic” (ADI) to solve mTSPD. The heuristic builds truck-drone tours from the original constructed mTSP solution, which consists only m truck tours. We propose three mTSP heuristics to be used for ADI and compare the performance among them.
3. We use the ADI heuristic to solve some generated small-scale (less than 10 nodes) test instances and compare the solution performance with mTSPD other existing TSP/FSTSP models MIP formulation solved by CPLEX. The results give us an insight on the savings achieved by implementing the multiple trucks and multiple drones delivery system
4. We implement the ADI heuristic on several TSP/mTSP benchmark problems and compare its performance with the modified FSTSP heuristic, which is based on the heuristic developed to solve the FSTSP (Murray & Chu, 2015). The solutions of both heuristics are compared with the best known optimal TSP/mTSP solutions in each benchmark. The analysis of the results will show how much the logistics operator would benefit from implementing drones along with trucks to make the last mile deliveries.

The remainder of the paper is structured as follows. Section 2 provides a description of the mTSPD problem and terminology with explicit assumptions. Section 3 presents the formal definition of the mTSPD model and its mathematical MIP formulation. Section 4 presents the new algorithm to solve the proposed model along with the pseudocode in detail. In Section 5, we provide the results of the numerical experiments on the conducted test instances and benchmark problems with the analysis of the performance of the algorithm. Section 6 concludes the paper and provides discussions for future research.

2. Problem description

The Multiple Traveling Salesman Problem with Drones (mTSPD) is an extension of the Multiple Traveling Salesman Problem (mTSP) with the implementation of drones in the operations. The mTSPD model formulation is derived from the FSTSP model in Murray and Chu (2015) with some additional constraints from the mTSP. In the mTSP, the m salesmen must visit n nodes, forming totally m tours, one per salesperson. There are two main constraints in the mTSP problem. The first constraint requires that all salesmen must depart and return to the starting node (depot) at the end of the trip, no matter which tour they choose. The second constraint states that every salesman must travel to a specific set of customers between the first and the last node given that each node can only be visited once by the assigned salesman except the starting node (depot) (Bektas, 2006). The typical objective of the mTSP is to find the total shortest tour that each salesman must travel from the depot to visit the assigned set of cities and back to the depot. In the mTSPD, the objective is to minimize the time that the last delivery is completed. The objective of the mTSPD problem follows a variant of mTSP, called min-max TSP, which aims to minimize the maximum tour length (time) of each salesman with the purpose to divide the cost (time) of tours among each salesman equally (Bertazzi, Golden, & Wang, 2015; Kivelevitch, Cohen, & Kumar, 2013). This min-max mTSP is used in the aid supplies delivery or express delivery when the delivery service time of each tour is prioritized over the total delivery cost of the entire operation (Campbell, Vandenbussche, & Hermann, 2008; Huang, Smilowitz, & Balcik, 2011). As drone delivery service aims to provide a faster service using the advantage of drone speed, we would like to use a min-max objective for the mTSPD problem. The solution of mTSPD requires exactly m tours, which is equal to the number of salesmen.

In this problem, a truck is treated as a salesman because we do not consider the capacity of trucks and demand quantity of customers. We assume that each truck has sufficiently large capacity to carry both packages and drones through the entire operation. At the depot, the departure times of all trucks are zero. All trucks must initially depart from the depot, serve all customers, and return to the depot. The model will determine the set of customers specifically visited by trucks and the set of customers visited by drones. It is required that each customer must be visited exactly once by either a truck or drone.

The mTSPD model represents the actual operation similar to the truck-drone delivery test conducted by UPS in February 2017 (Hughes, 2017; Zito & Radocaj, 2016), which showed the case when a drone launched from atop one of the UPS delivery trucks autonomously delivers a package and then returns to the vehicle. This happens simultaneously while a driver continues to drive along a route, delivering packages. In addition, we assume the drone can be retrieved by any truck that is nearby and not necessarily the same truck that it is

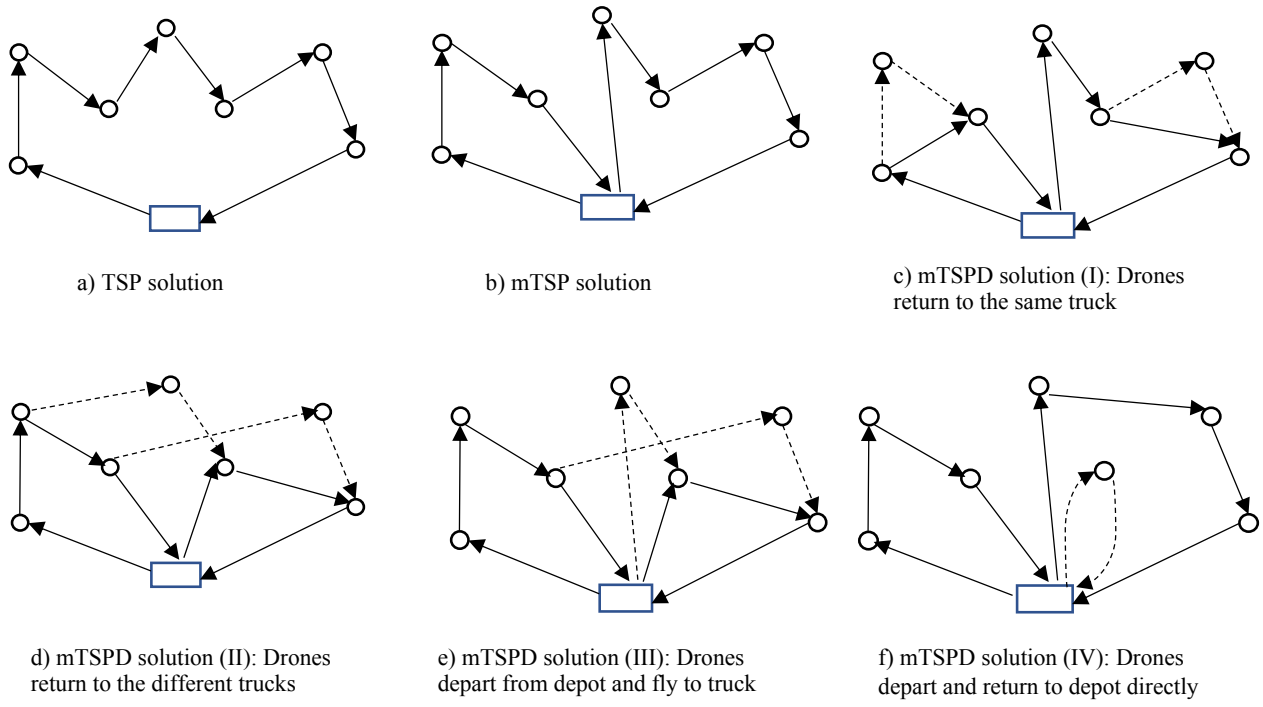


Fig. 1. Illustration of the feasible solutions from TSP, mTSP and mTSPD.

launched from. Fig. 1 represents an illustrative problem and different types of solutions from mTSPD comparing to the mTSP and TSP on the same problem. The solid line in all sub Fig. 1(a)–(f) refers to the truck tour and the dot line refers to the drone tour.

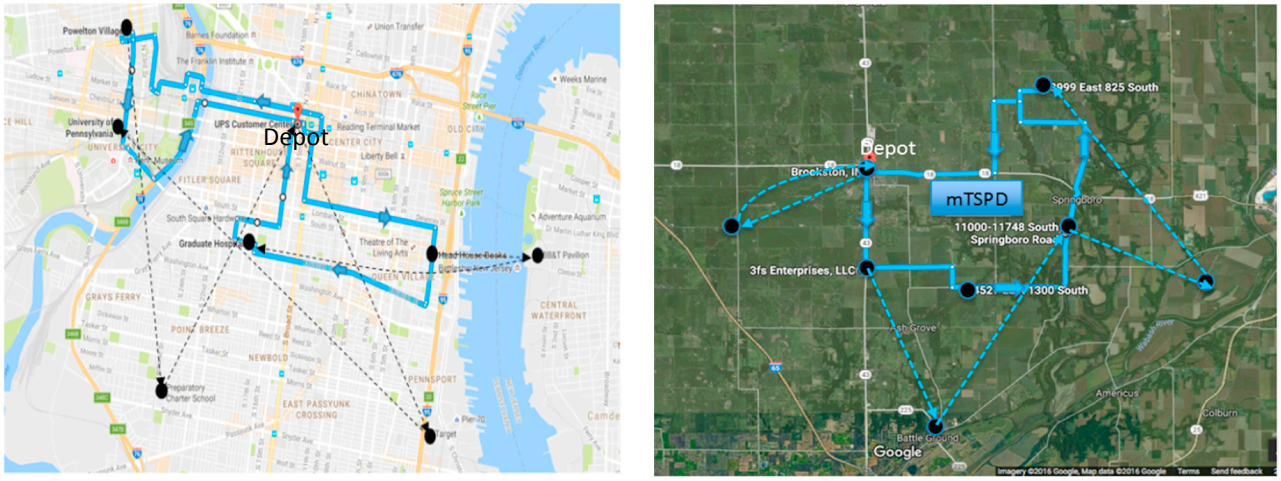
Turning from the described operations to the routing problem, several assumptions must be made:

(1) We disregard the capacity of each truck and assume each truck has a sufficiently large capacity to carry both packages and drones through the entire operation. This is to reduce the complexity of the problem and to ensure that a truck can operate without exceeding its capacity.
(2) Drones are assumed to be homogeneous with the same configuration and can carry one package with a small payload up to 5 lb (2.3 kg) at a time (Glaser, 2017). Once the drone finishes the delivery to a customer, it must immediately fly to any customer node that has not been visited yet, or it can fly back to the depot directly. Due to these characteristics, drones can be launched from a truck and return to another truck in the fleet. There are no specific assignments of drones to trucks once they leave the depot.
(3) In our model, we disregard the set-up and recovery times when the drone is launched or retrieved at a particular node. This set-up and recovery time can be negligible since their values are small compared to the truck and drone travel time and will not affect how the solution route is determined in the model. We also assume that the drone can complete its delivery and return to the truck before it runs out of battery, or it can land on a recharging stations to extend its flying range (Hong, Kuby, & Murray, 2017; Gentry, Hsieh, & Nguyen, 2016).
(4) Drones can only merge with a truck at a customer node and are not allowed to merge with a truck in any intermediate location. From a practical view, it would be difficult for a drone to land on a moving truck, as both of their speeds have to match (Buchmueller, Green, Kalyan, & Kimchi, 2016). Merging a drone with a movable truck requires the drone to reduce its speed and the truck to increase its speed, which violates the constant speed assumption of the drone and the truck and would decrease a drone's performance from flying at full speed. Also, trucks and drones must wait for each other whenever one arrives at the customer node before the other. This assumption is to ensure that the drone is serviced prior to departure and right after merging with the truck. Once the drone arrives at the truck, it is serviced for the functional check and battery charge at the truck.
(5) Multiple drones can fly simultaneously; however, only one drone can be launched or retrieved at each particular node at the same time. In other words, the truck cannot launch or retrieve more than one drone in any customer node. Allowing multiple launches and retrievals requires that the delivery truck must reserve a certain amount of space for keeping drones. Currently, our model does not include the capacity constraint of the drone's space which refers to the truck's capacity to carry some finite number of drones. Without limiting the number of retrievals, one of the feasible solutions is to launch as many drones as possible from a single node. For operational simplicity and to reduce model complexity, it would be safe to limit the number of launches and retrievals to just one.
(6) We assume that the FAA regulations about the visual line-of-sight (VLOS) can be relaxed by allowing a first-person-view piloting. In this system, the drone pilot controls the drone, as he can see what the drone sees as it flies (Browne, 2017).

Fig. 2 represents the result of two generated case studies we have conducted using the mTSPD model to generate the solution routes. Fig. 2(a) shows the paths of both trucks and drones. These case studies illustrate the advantages of drones compared with the trucks in delivery operations. The trucks can make deliveries to multiple customers on the tour while the drone can deliver the package to a specific customer at a faster speed. It also shows the substantial advantage of the drone, as it does not get stuck in traffic and can travel across the river quite easily. Fig. 2(b), on the right, similarly shows that the drone can fly across areas where the road has no access to make a delivery. In both cases, the drone gains the advantage of flying in a Euclidean travel space while the truck has to follow the road distance.

3. Mathematical formulation

The mTSPD is defined on a directed graph $G = (V, E)$, where V is the



a) The solution routes in downtown Philadelphia

b) The solution routes in Brookston, Indiana

Fig. 2. Examples of mTSPD solution routes from case studies which show the truck and drone combined routes. The blue solid lines indicate the truck's path and that dashed lines indicate the flight path of the drone.

set of n nodes representing customers with one depot and E is the set of arcs. Let $\tau^T_{i,j}$ be a truck travel time associated with E, $(i, j) \in E$ and $\tau^D_{i,j}$ be a drone travel time associated with E, $(i, j) \in E$. Differentiating the travel times for the truck and drone accounts for each vehicle's unique travel speed. The mTSPD is said to be symmetric if $\tau^T_{i,j} = \tau^T_{j,i}$ and $\tau^D_{i,j} = \tau^D_{j,i}$ and asymmetric otherwise. Let m be the number of trucks that depart and return to the depot. Next, denote the set of customer nodes by $C = \{1, 2, 3, 4, 5, 6, \cdots, n\}$. Although only one physical depot location exists, we assign it to two unique node numbers at $0(s)$, the starting depot, and $0(r)$, the ending depot. Set $C_0 = C \cup \{0(s)\}$ as the set of customer nodes including the starting depot and set $C_+ = C \cup \{0(r)\}$ as the set of customer nodes including the ending depot.

Following the formulation in Murray and Chu (2015), we define $F = \{(i, j, k)\}$ as all possible three-node sorties of the drone path. An element $(i, j, k) \in F$ if the following conditions hold: (1) The launch node i must not be the ending depot node (i.e., $i \in C_0$). (2) The delivery node j must be in the customer set and must not be the same as the launch point (i.e., $j \in C$ such that $i \neq j$). (3) The merging point k can be either a customer node or the ending depot and cannot equal i or j ($k \in C_+$ such that $k \neq i$ and $k \neq j$).

We define the following decision variables: Let $x_{i,j}$ equal to 1 if arc $(i, j) \in E$ is used on a truck path and 0 otherwise. This refers to the situation when the truck travels from node $i \in C_0$ to $j \in C_+$ where $i \neq j$. Let $y_{i,j,k}$ equal to 1 if arc (i, j) and $(j, k) \in E$ is used on the path and 0 otherwise. This refers to the situation when a drone is launched from node $i \in C_0$ to node $j \in C$ (visiting customer node) and merges with a truck or the ending depot at node $k \in C_+$ such that $(i, j, k) \in F$. We denote Tl_j as the truck arrival time at node $j \in C_+$ and Dl_j as the drone arrival time at node $j \in C_+$. Tl_j and Dl_j are the arrival times of the truck and drones at node j respectively. Lastly, the auxiliary decision variable u_i is used in the TSP subtour elimination constraints (Desrochers & Laporte, 1991). All the mentioned notations can be summarized as follows.

Indices	
i, j, k	Represent customers, and depot
Sets	
C	Set of customers, $\{1, 2, 3, 4, 5, 6, \cdots, n\}$
C_0	Set of customer nodes including the starting depot, $C \cup \{0(s)\}$
C_+	Set of customer nodes including the ending depot, $C \cup \{0(r)\}$
F	Set of all possible three-node sorties of the drone path
Parameters	
$\tau^T_{i,j}$	Truck travel time between nodes i and j
$\tau^D_{i,j}$	Drone travel time between nodes i and j
m	Number of trucks in the entire fleet
n	Number of total customers to be served
Variables	
$x_{i,j}$	1 if a truck traverses arc (i, j) from customer i to customer j; otherwise, 0
$y_{i,j,k}$	1 if a drone traverses arc (i, j) and (j, k) from customer i to customer j and from customer j to customer k; otherwise, 0
Tl_j	Truck arrival time at node j
Dl_j	Drone arrival time at node j

The proposed MIP formulation of mTSPD is presented as follows.

Objective

$$\text{minimize} Dl_{0(r)} \tag{1}$$

Subject to

$$\sum_{\substack{i \in C_0 \\ i \neq j}} x_{i,j} + \sum_{\substack{i \in C_0 \\ i \neq j}} \sum_{(i,j,k) \in F} y_{i,j,k} = 1 \quad \forall j \in C \tag{2}$$

$$\sum_{j \in C_+} x_{0(s),j} = m \tag{3}$$

$$\sum_{i \in C_0} x_{i,0(r)} = m \tag{4}$$

$$u_i - u_j + n x_{i,j} + (n-2) x_{j,i} \leq n - 1 \quad \forall i, j \in C, \quad i \neq j \tag{5}$$

$$1 + (n-2) x_{i,0(s)} + \sum_{\substack{j \in C \\ i \neq j}} x_{j,i} \leq u_i \leq n - (n-2) x_{0(s),i} - \sum_{\substack{j \in C \\ i \neq j}} x_{i,j} \quad \forall i \in C \tag{6}$$

$$\sum_{\substack{i \in C_0 \\ i \neq j}} x_{i,j} = \sum_{\substack{k \in C_+ \\ k \neq j}} x_{j,k} \quad \forall j \in C \tag{7}$$

$$\sum_{\substack{j \in C \\ i \neq j}} \sum_{(i,j,k) \in F} y_{i,j,k} \leq 1 \quad \forall i \in C_0 \tag{8}$$

$$\sum_{\substack{i \in C_0 \\ i \neq k}} \sum_{(i,j,k) \in F} y_{i,j,k} \leq 1 \quad \forall k \in C_+ \tag{9}$$

$$2 y_{i,j,k} \leq \sum_{\substack{h \in C_0 \\ h \neq i}} x_{h,i} + \sum_{\substack{l \in C \\ l \neq k}} x_{l,k} \quad \forall i, j \in C, \quad \forall k \in C_+ \tag{10}$$

$$y_{0(s),j,k} \leq \sum_{\substack{h \in C_0 \\ h \neq k}} x_{h,k} \quad \forall j \in C, \quad \forall k \in C_+ \tag{11}$$

$$\sum_{\substack{i \in C_0 \\ i \neq j}} \sum_{(i,j,k) \in F} y_{i,j,k} \leq 1 - \sum_{\substack{a \in C \\ a \neq j}} \sum_{(j,a,b) \in F} y_{j,a,b} \quad \forall j \in C \tag{12}$$

$$\sum_{\substack{i \in C_0 \\ i \neq j}} \sum_{(i,j,k) \in F} y_{i,j,k} \leq 1 - \sum_{\substack{a \in C_0 \\ a \neq j}} \sum_{(a,b,j) \in F} y_{a,b,j} \quad \forall j \in C \tag{13}$$

$$Dl_{(i)} \geq Tl_{(i)} - M\left(1 - \sum_{i \neq j}^{j \in C} \sum_{(i,j,k) \in F}^{k \in C_+} y_{i,j,k}\right) \quad \forall\, i \in C \tag{14}$$

$$Dl_{(i)} \leq Tl_{(i)} + M\left(1 - \sum_{i \neq j}^{j \in C} \sum_{(i,j,k) \in F}^{k \in C_+} y_{i,j,k}\right) \quad \forall\, i \in C \tag{15}$$

$$Dl_{(k)} \geq Tl_{(k)} - M\left(1 - \sum_{j \neq k}^{j \in C_0} \sum_{(i,j,k) \in F}^{j \in C} y_{i,j,k}\right) \quad \forall\, k \in C_+ \tag{16}$$

$$Dl_{(k)} \leq Tl_{(k)} + M\left(1 - \sum_{j \neq k}^{j \in C_0} \sum_{(i,j,k) \in F}^{j \in C} y_{i,j,k}\right) \quad \forall\, k \in C_+ \tag{17}$$

$$Tl_{(k)} \geq Tl_{(h)} + \tau^{T}_{(h,k)} - M(1 - x_{h,k}) \quad \forall\, h \in C_0, \quad \forall\, k \in C_+ \tag{18}$$

$$Dl_{(k)} \geq Dl_{(i)} + \tau^{D}_{(i,j)} + \tau^{D}_{(j,k)} - M(1 - y_{(i,j,k)}) \quad \forall\, i \in C_0, \quad \forall\, j \in C, \quad \forall\, k \in C_+ \tag{19}$$

$$Dl_{0(s)} = 0 \tag{20}$$

$$Tl_{0(s)} = 0 \tag{21}$$

$$x_{i,j} \in \{0, 1\} \quad \forall\, i, \mathrm{j} \in C \cup C_0 \cup C_+ \tag{22}$$

$$y_{i,j,k} \in \{0, 1\} \quad \forall\, i, \mathrm{j}, \mathrm{k} \in C \cup C_0 \cup C_+ \tag{23}$$

$$Dl_i \geq 0 \quad \forall\, i \in C \cup C_0 \cup C_+ \tag{24}$$

$$Tl_i \geq 0 \quad \forall\, i \in C \cup C_0 \cup C_+ \tag{25}$$

The objective function (1) minimizes the arrival time of drones and trucks at the depot. The objective function is equivalent to $\min\{\max\{Tl_{0(r)}, Dl_{0(r)}\}\}$ since both drones and trucks have to wait for each other and the arrival time of trucks and drones at the depot will be adjusted to be the same by constraint (14)–(17). Constraint (2) ensures that each customer will receive the package either by a drone or truck. Constraint (3) and constraint (4) ensure that the trucks depart from and arrive to the depot. Constraints (5) and (6) are sets of the Desrochers and Laporte (DL) subtour elimination constraint which ensures that there is no subtour in all tours of the trucks (Desrochers & Laporte, 1991). Constraint (7) guarantees that whenever the truck arrives at a node, it must depart from the node as well. Constraint (8) represents that at most one drone can depart at each stop of the truck. Similarly, constraint (9) defines that at most one drone can arrive to a truck if it visits a node. Without these constraints, there is no restriction on how many drones can be launched from and land on a certain node, which would violate assumption 5 in Section 3. Constraint (10) states that a truck must visit node i and node k if the drone is launched from node i and is retrieved at node k. Similarly, constraint (11) ensures that when the drone flies from the depot to node j and k respectively, a truck must correspondingly depart from the depot and eventually arrive at node k. Constraint (12) describes the cases that if the drone flies from node i to node j to node k, there are no other drones that make such a delivery simultaneously from node j to node a to node b. Constraint (13) enforces that if a drone departs from node i to visit node j and merges with the truck at node k, no other drones can arrive at the delivery node j. Constraint (12) and (13) will ensure the flow conservation for the delivery node. Constraint (14) and (15) state that the departure time of drone and truck must be the same. Also, once the drone and truck are in the same node, they must wait for each other before each of them can leave the node. For an example, let assume that both truck and drone are at node i. If in the next move, the drone travels from node i to j to k, then constraints (14) and (15) will be binding, resulting the same departure time for both truck and drone. Similarly, constraints (16) and (17) ensure that the arrival time of both truck and drone will be the same when they merge at the same node. These sets of constraints are based on the assumption that if either the drone or truck arrives earlier than the other, the earlier one has to wait until the later one arrives (both constraints are binding, resulting in the same arrival time of both truck and drone). Constraints (18) keeps track of the arrival time of the truck at every node. It adds the truck travel time to the previous customer node when the truck travels from one customer node to another customer node. Similarly, constraint (19) keeps track of the arrival time of the drone at the node to which the drone returns after making a delivery. This constraint considers the drone's travel time from the previous departure node to the delivery node to the arrival node. Constraints (20) and (21) set the initial departure time of drones and trucks at the depot to be zero. Constraints (22)–(25) specify the types and ranges of the variables. Note that the M value must be large enough. Thus, we can use the minimum total travel time of a single truck to visit all customers in one single tour, i.e., solve a regular TSP.

We have tested the MIP formulation of the mTSPD using the CPLEX solver on small and medium size (5–50 nodes) problems, which we generated based on the method in Section 5.1. Although we are able to obtain the optimal solutions within one hour on 10-node problems, the computational time becomes significantly high (more than 8 h) on larger-scale problems (more than 10 customer nodes). As a result, we have developed an Adaptive Insertion algorithm (ADI) to solve the mTSPD problem. We propose a modified version of the FSTSP heuristic, so-called "Adapted FSTSP heuristic" so that it can be used to solve the mTSPD problem. In the experiment section, we will compare the performance of our proposed heuristic with the FSTSP heuristic proposed by Murray and Chu (2015). The numerical results and related statistics will be provided in Section 5.

4. The proposed algorithm

The Adaptive Insertion algorithm, "ADI" is developed to efficiently solve the mTSPD. This heuristic is based on the greedy node(s)-insertion strategy (Gendreau, Hertz, & Laporte, 1992; Lu & Dessouky, 2006). Prior to developing the heuristic, we used the MIP model to solve the mTSPD, and we were able to solve and return the optimal solutions of the problems with up to 10 customer nodes within an hour. However, larger problem sizes could not be optimally solved even within 8 h. Therefore, heuristic solution approaches are required to find optimal or near optimal solutions once the problem size becomes larger than 10 nodes.

There are two phases in our proposed our heuristic. In the first phase, an initial mTSP solution which consists only truck tours is generated through construction heuristic algorithm. In the second phase, we build a mTSPD solution from a mTSP solution using different types of removal and insertion operators. The mTSPD solution consists of the improved truck tours together with the drone tours constructed in the second phase. The algorithm is detailed in the following section.

4.1. Initial mTSP solution

Since a mTSPD solution is based on the constructed tours of the mTSP solution in the first phase, different initial mTSP solutions can affect solution quality of the mTSPD in the second phase. Therefore, we use three heuristics to generate the mTSP routes as follows:

Genetic Algorithm (GA)

The GA is based on the principles of natural selection and genetics to generate the better solution over the solution from previous generation. It starts from a group of initial solutions. A fitness function is used to evaluate the performance of the solutions. Then, the two solutions called parent solutions are selected based on certain probability to perform a crossover/mutation operation to produce two new solutions of the next generation. If the new solutions have better fitness values, they will replace the old solutions. The selection, crossover, and mutation operations are repeated for the rest of the population until the population size of the new generation is the same as the size of the previous generation. This completes one iteration (generation). The procedure continues until the number of certain generations is reached

or the solution quality cannot be improved (Carter & Ragsdale, 2006; Li, Sun, Zhou, & Dai, 2013).

To implement GA in our problem, we have to modify the fitness function so that it can be used with the min-max mTSP. We use one One chromosome representation technique, order crossover and two mutation operators proposed by Sedighpour, Yousefikhoshbakht, and Darani (2011) in the GA. We also conduct some experiments with different sets of parameters, including selection probability, cross-over probability, mutation probability and the number of generations to ensure obtaining the high quality solutions by the algorithm. A pseudocode of the proposed GA algorithm for the mTSP with the control parameters is presented below.

Control parameters
NumGen: Number of Generations
PopSize: PopulationSize (Number of chromosomes)
S: Current solution of the mTSP
f: The fitness value (objective value) of S
S^*: *The best solution of the mTSP*
f^*: *The best fitness value of* S^*
p_C: *Crossover probability*
p_m: Mutation probability
p_{select}: Selection probability
Proposed GA for mTSP
Load Input parameters
Create initial populations of the mTSP //Each chromosome represents a route solution//
Find the best *S*. Record the best solution as S^* with the fitness value of f^*
For the total *NumGen*
  For all *PopSize*
    Determine *f* for each solution route (chromosome)
  End For
  Keep the best 2 solutions without going to crossover
  Select parents to perform crossover with p_{select} (Roulette Wheel Selection)
  Perform Order crossover with p_C for all pairs of chromosomes
  Perform Two mutation operators with p_m for all chromosomes
  Find the best *S* among the current populations.
  If f (*Best S*) $\leq f^*$
    $f^* = f$ (*Best S*)
  End If
End For

Combined K-means/Nearest Neighbor

This heuristic combines K-means clustering algorithm with famous nearest neighbor algorithm. K-means technique is quite popular among various clustering techniques because of its ability and efficiency among clustering data. Among the set of customer nodes (C), we partition them into k clusters. Please note that the notation k in this K-means algorithm is different from the notation k in Section 3 which is previously defined as an index for the customer node (i, j, k). Each cluster represents a tour of one tuck. Initially, we randomly assign k cluster centroids into the map. Based on the distance from the centroid, each customer node is assigned to the nearest centroid. Each cluster centroid is updated based on the nodes assigned to the cluster. The process will be repeated until the centroids remain the same or no point changes clusters (Jain, 2010). Let $X = \{x_i\}$, $i = 1, \cdots, n$ be the set of n customer nodes to be clustered into a set K cluster with the cluster mean, $C = \{c_k, k = 1, \cdots, K\}$. The goal of k-means is to minimize the sum of the squared error over all k clusters,

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2 \tag{26}$$

The K-means algorithm to generate K clusters is composed of the following steps:

Step 1: Randomly generate K points into the map represented by the customer nodes that are being clustered. These K points represent initial cluster centroids.
Step 2: Assign each node to the cluster that has the closest centroid.
Step 3: Once all nodes have been assigned to the clusters, recalculate the locations of the K centroids.
Step 4: Repeat steps 2 and 3 until the centroids no longer move. This procedure assigns the customer nodes into clusters which corresponds to minimize Eq. (26).

After partitioning all customer nodes into k clusters, we then use Nearest Neighbor (NN) to solve for the TSP tour in each cluster (Arora, Agarwal, & Tanwar, 2016). It is required that the starting node of each TSP tour must be a depot node and the ending node of each TSP tour must be a depot node. In NN, the truck starts at the depot, repeatedly visits the nearest customer node until all customers in the cluster are visited and returns back to the depot.

The Nearest Neighbor algorithm for the TSP solution is composed of the following steps:

Step 1: Set the depot as the current visited vertex. The rest of the customer nodes are listed in unvisited vertex set.
Step 2: Select the node V with the lowest travel cost and set it as a current vertex.
Step 3: Mark V as visited and update unvisited vertex set.
Step 4: If all the vertices in domain are visited, then terminate. The final visited vertex must be a depot.
Step 5: Repeat step 2–4.

When combing K-means and Nearest Neighbor (NN) algorithms together, we can generate the mTSP solution within small computational effort. The K-means algorithm will build m clusters while NN will generate the tour for each cluster.

Random cluster/tour

In this heuristic, random k clusters are generated using uniform distribution. Each customer node is uniquely assigned to one cluster. Each cluster must have at least one customer node and there is no restricted number of how many customer nodes can be assigned in a cluster. In each cluster, a truck starts at the depot, randomly visits all customer nodes in the cluster and returns back to the depot. The algorithm is composed of the following steps:

Step 1: Randomly generate K points into the map based on the number of trucks (m).
Step 2: Assign each customer node to the cluster. Each node can only be assigned to one cluster. Each cluster must be assigned at least one node.
Step 3: For each cluster, generate the random tour starting from the depot, visit each node exactly once and return to the original depot.

4.2. Construction of mTSPD solution

Once obtaining the initial mTSP solution, we can construct the mTSPD solution by gradually replacing some of the truck customer nodes with drone customer nodes. The ADI algorithm gradually builds up a solution by switching the position of the nodes, reevaluating the objective value, and terminating once the termination condition is met. The heuristic seeks for the best way to remove the truck node from the tour and insert the node back to the tour as a drone customer node. We can perform the truck/drone tour construction by executing two main types of operators: the removal operator and the insertion operator. The removal operator takes one node out of the existing tours. This particular node will be sequentially added back to one of the tours via three types of insertion operators. The algorithm aims to maximize the *Saving* or the difference between "Time Decrease from removing node" (ΔT^D) and "Time Increase from adding node" (ΔT^I) in each iteration until the

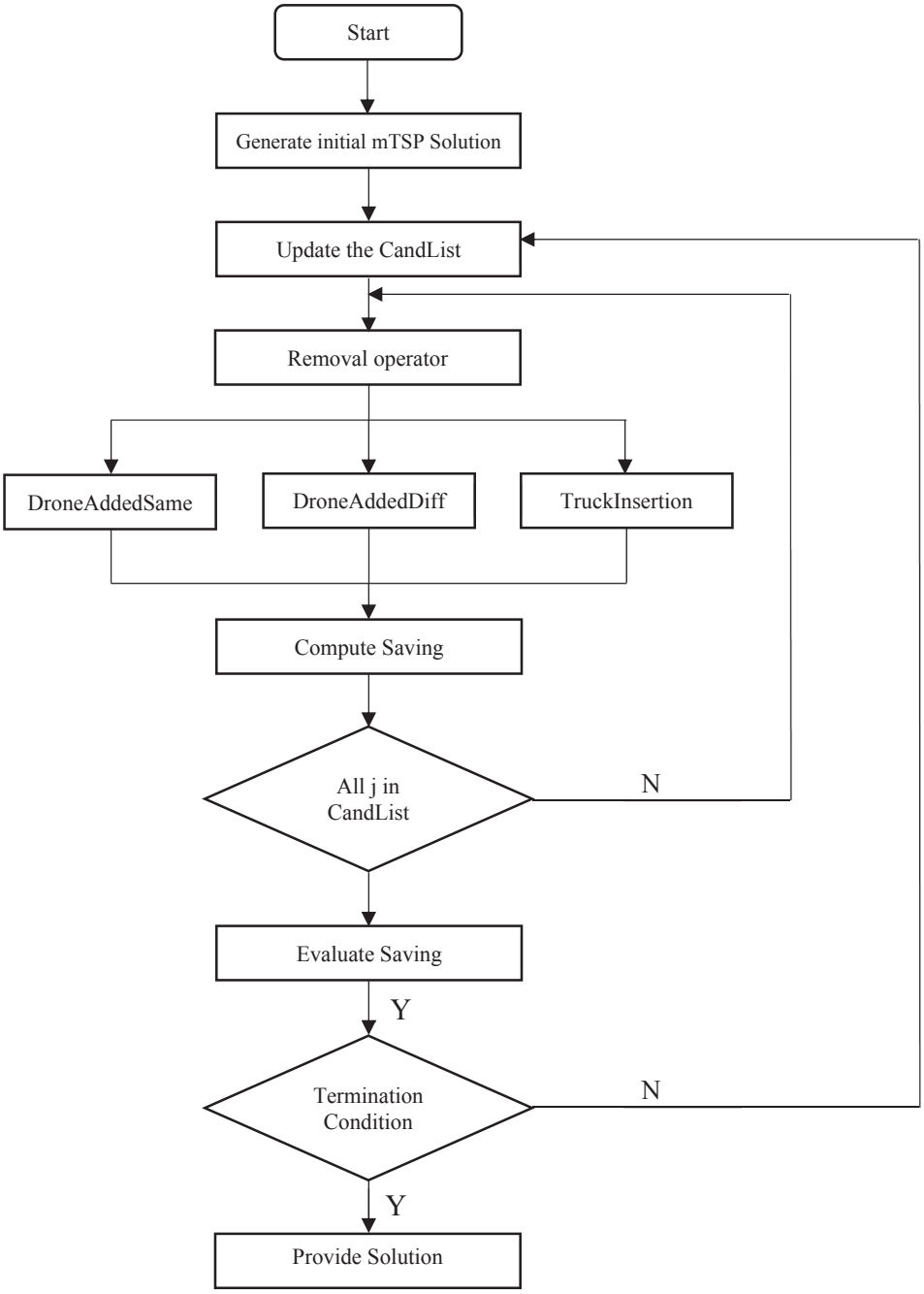


Fig. 3. Framework of main mTSPD-ADI.

termination condition is met. We also define "*#NCS*" to keep track of the number of customers who are served by any drone.

Fig. 3 demonstrates the framework of the described algorithm. As previously mentioned, the algorithm begins by generating the mTSP initial solution through the described heuristics in Section 4.1. Denote the set *Tour = {1, 2, 3, …m}* and assign an index *select* to track all the tours from *select = 1, 2, 3, …m*. Next, we define the set of *CandList = {1, 2, 3, …,n}*. The members of the CandList are all the customers (C) with the size *n* on the map. The removal and insertion procedures are applied on each element of the *CandList*. Once all members in *CandList* complete these two consequential procedures, all the feasible solutions will be evaluated through the fitness function max $\{(\Delta T_j^D - \Delta T_j^I)\}$; $\forall j \in$ *CandList* (line 28 of Algorithm ADI). Then, node *j* with the best fitness value is selected and removed from the *CandList* set (line 29 of Algorithm ADI). We also need to update the solution as well as other global variables including: *Saving, #NCS, all Truck Route (T_R), Drone Path (D_P),* and the arrival time associated with each customer node (t_j^{select}) (line 30–31 of Algorithm ADI). For any truck path (*i, j, k*), we denote $t_k^{select*}$ as the arrival time at node *k* prior to removing node *j* from tour *select* and $t_k^{select**}$ as the arrival time at node *k* after removing node *j* from tour *select*. We also keep track of the set of nodes where the drones are launched from using the notation *Lch_N* and the set of nodes where the drones land to using the notation *La_N*. The algorithm will repeat iteratively and terminate once $\Delta T_j^S = \Delta T_j^D - \Delta T_j^I \leq 0$ or *CandList* $\in \emptyset$. Please note that if *#NCS* $\geq$ *Max(#NCS)* (line 15–16 of Algorithm ADI), we can only insert node *j* between nodes that are currently served by the truck. This constraint is used to prevent the algorithm from generating an excessive number of drone delivery paths with many infeasible solutions.

Algorithm: ADI

1. Initialize
2. Parameters: m, n, τ^T, τ^D
3. Generate the initial mTSP solution with m truck tours
4. Compute time (t_i^{select}) associated with each node i in tour *select*
5. $Max\ \#NCS = \sum_i^m \lfloor \frac{(\#\ customersintouri - 2)}{2} \rfloor$
6. Create T_R = {Tours, t_i^{select}, Lch_N, La_N}
7. Set *Saving* = 1, *CandList* = {*All C*}, *Lch_N* = [], *La_N* = [], *D_P* = [], ΔT^S = [], *#D* = 0
8. **While** *Saving* > 0
9. **For** $j \in$ *CandList*
10. Find a tour where node j belongs to; called *SelectTour*
11. **If** *#C* in *SelectRoute* < 2
12. $\Delta T^S = -inf$
13. **Else**
14. Call Node_Removal(T_R, D_P, j, τ^T, τ^D)
15. **If** *#NCS* < *Max #NCS*
16. Call *DroneAddedSameRoute (T_R, D_P, j, τ^T, τ^D)*
17. Call *DroneAddedDiffRoute (T_R, D_P, j, τ^T, τ^D)*
18. **Else**
19. $\Delta T1_j^I$ = inf
20. $\Delta T2_j^I$ = inf
21. **End** If
22. Call TruckInsertion(T_R, D_P, j, τ^T, τ^D)
23. **End** If
24. $\Delta T_j^I = \min(\Delta T1_j^I, \Delta T2_j^I, \Delta T3_j^I)$
25. Select *T_R, D_P* associated with the node j in ΔT_j^I
26. $\Delta T_j^S = \Delta T_j^D - \Delta T_j^I$
27. **End** For
28. Find Saving = $max_{j \in CandList}\ (\Delta T_j^S)$
29. Remove node j from *CandList*
30. Update: *T_R and D_P* associated with *Saving*
31. Update: *#NCS* and *CandList*
32. **End** While

Removal operator

Given the chosen node j from the main algorithm, the *Node_Removal* operator would determine how much objective value decreases when removing node j from the particular tour *select.* There are three main steps in the procedure. The first step is to remove node j from tour *select* and re-compute the arrival time to the depot at each truck. The next step is to find the arrival time to the depot of the last truck (latest arrival time at the depot) stored in the variable called Latest Time the Truck Returns to the Depot After removing node j *(LTRDA)*. The last step is to find the difference between *LTRDA* and the Latest Time the Truck Return to the Depot Before removing node j *(LTRDB)*. The value difference is stored in variable ΔT_j^D. An example of the Removal operator is shown in Fig. 4.

In the above example, the sub figure on the left represents the original routing solution for three trucks and the updated routing solution once applying the Removal operator. It takes 20 min, 15 min and 30 min for truck 1, truck 2 and truck 3 to complete the deliveries accordingly. We assign node 8 as the node to be removed from the existing solution. The Latest Time the Truck Return to the Depot Before (LTRDB) removing node 8 is 30 min which is the duration truck 3 completes its job. Once removing node 8 from the solution, the truck 3's delivery time is updated to 20 min. The new objective for the solution, the Latest Time the Truck Returns to the Depot After (LTRDA) removing node 8, is updated to 20 min. Hence, the difference between the objective value of the original and the updated solution is 10 min, which is stored in the variable ΔT_8^D.

Insertion operator

Once node j is removed from tour select, we need to find the location to insert node j. Three types of insertion algorithms are used for the search process: *DroneAddedSameRoute, DroneAdded-DiffRoute,* and *TruckInsertion*. In *DroneAddedSameRoute*, we assign node j as a drone delivery node and insert it into one of the m tours. The path *(i, j, k)* is constructed such that the drone departs from node i, serves the customer at node j, and merges with the truck at node k (the same truck that the drone is launched from). To explain briefly how this operator works, for each tour from all the truck tours, Tour[1], Tour[2], Tour[3],…, Tour[m], we select a pair of *(i, k)* nodes where node i must precede node k to insert node j into. After inserting node j between node i and node k, we recalculate the arrival time when the truck returns on the selected tour, where node j is added. Finally, we update the latest arrival time at the depot for the whole route and calculate the difference between the Latest Time the Truck Return to the Depot After adding node j *(LTRDAI)* and the Latest Time the Truck Return to the Depot Before adding node j *(LTRDBI)*. This difference is stored in variable $\Delta T1_j^{I,p}$, where p is the index of all possible combinations of this type of insertion. The minimum of $\Delta T1_j^{I,p}$ is chosen among all possible combinations. Slightly different from *DroneAddedSameRoute*, the operator *DroneAddedDiffRoute* restricts that the drone path *(i, j, k)* must be constructed such that node i and node k are selected from different tours. Hence the drone would merge with a different truck at node k, not the one it was launched from, after visiting node j. The best difference in objective value before and after inserting node j into the route is stored in variable $\Delta T2_j^I$. Lastly, *TruckInsertion* deals with the case in which node j is assigned as a truck delivery node. Node j must be inserted between node i and node k. Both node i and node k must be located in the same tour, where node i must precede node k and the nodes must be adjacent to each other. These three operators provide different ways of searching and sorting methods. The basic idea of ADI is to look for the lower ΔT^I which tends to give the maximum possible *Saving*. We then select min($\Delta T1_j^I$, $\Delta T2_j^I$, $\Delta T3_j^I$) (line 24 of Algorithm 1) after completing the searches. Fig. 5 illustrates all three types of Insertion operators.

In the above example, all three sub figures represent the routing solution before and after applying three different types of operators. For the purpose of simplicity, we select node 8 as the node to be inserted into the original solution. In 5(a), the original objective value is 20 min, which is the Latest Time the Truck Return to the Depot Before (LTRDBI) adding node 8 into the solution. The node 8 is inserted between the depot and node 7 as a drone delivery node of the truck 3's tour. The new delivery time of the truck 3 in 5a) becomes 26 min which is equivalent to the new objective value or the Latest Time the Truck Return to the Depot After adding node 8. Consequently, the difference between the objective value of the original and the updated solution is 6 min recorded in the variable $\Delta T1_8^I$. In 5b), the node 8 is inserted as a drone delivery node between the customer node 6 of truck 2 and the customer node 7 of truck 3, resulting in the new objective value of 25 min. The difference between the objective value of the original and the updated solution is 5 min recorded in the variable $\Delta T2_8^I$. Lastly, the node 8 is inserted as a truck delivery node between the customer node 6 and the depot of the truck 2's tour, resulting in the new objective value of 27 min. The difference between the objective value of the original and the updated solution is 7 min recorded in the variable $\Delta T3_8^I$. Among the three types of insertions, $\Delta T2_8^I$ return the lowest cost of insertion and the following new solution from the second type of insertion is accepted.

5. Computational examples and results

Three experiments were conducted to evaluate the MIP formulation and the performance of the proposed algorithm. In the first experiment, we solved some generated random instances with size of 25 and 50 customer nodes using Adaptive Insertion algorithm with three types of initial mTSP heuristics described in Section 4.1. The solutions of the ADI heuristic are compared with the solutions from CPLEX solver. The details of the instance generation will be explained in Section 5.1.

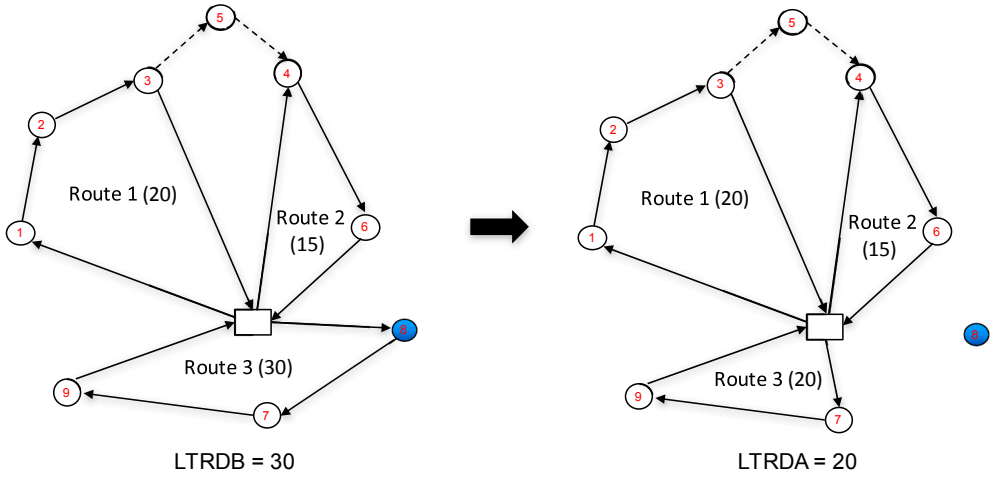


Fig. 4. Illustration of removal operator: $\Delta T_j^D = LTRDB - LTRDA = 10$.

In the second experiment, we solved the small test instance (9 nodes) on different types of generated instances which were generated the same way as in the first experiment and compared the performance of our proposed algorithm with the optimal solution obtained by MIP. Using the same set of instances, we also use three different MIP formulations by CPLEX to solve them optimally and compare their solutions. These MIP models are: TSP (one truck), FSTSP (one truck and one drone) and mTSPD (multiple trucks and multiple drones).

In the last experiments, we demonstrate the performance of our algorithm on the well-known TSP/mTSP benchmark problems (Reinelt, 1991), which are relatively large and cannot be solved by MIP solver within reasonable time. The solutions from the ADI are compared with the optimal solutions from the min-max TSP/mTSP problem, as well as the solutions from the Adapted FSTSP heuristic. Note that the Adapted FSTSP heuristic is modified from Murray and Chu's heuristic (Murray & Chu, 2015) to solve the mTSPD problem by implementing the FSTSP heuristic for each tour of the trucks and selecting the tour with the longest arrival time back to the depot.

Regarding parameter setting, we set the truck travel time to be 1.5 time units longer than the drone travel time ($\tau_{j,i}^T = 1.5\tau_{i,j}^D$) since the drone speed is roughly about 1.5 times faster than the truck speed (Brar et al., 2015). We assume that both truck and drone travel in Euclidean space. All the algorithms were executed in Matlab on a computer with 2.7 GHz Intel Core i5 with 8 GB RAM running Windows 7 64-bit mode. All the mixed integer linear programming models were solved using GAMS 23.51 with CPLEX solver.

5.1. Instance generation

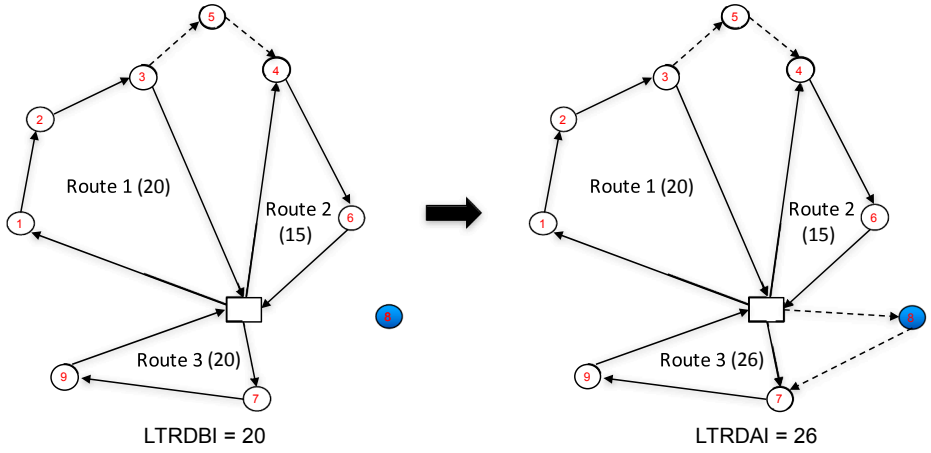
To evaluate the performance of the MIP model and ADI heuristic on the mTSPD, we generated 150 medium size test instances (25, 50 nodes), and 35 small size test instances (9 nodes). A set of customer nodes are created within the area of $1000 * 1000$ unit^2. There are mainly five types of problems based on how the nodes are distributed on the map, as shown in Fig. 6. In the type 1 problem, the depot is located at the center, at coordinate (500, 500) and surrounded by different customer nodes, which are uniformly distributed on the 1000×1000 square. The type 2 problem is similar to the Type 1 problem, except that the depot is now located at the bottom of the map at coordinate (500, 0). In the type 3 problem, the customer nodes are uniformly generated on the circle area with the radius of 500. The depot is located at coordinate (500, 500). On types I, II and III problems, we would like to see how the tour can be constructed when the customer nodes are located both close and far from the depot, and whether the location of the depot has any significant effect on the solution quality. In the type 4 problem, we assign the customer nodes to be far from the depot at coordinate (500, 500) and distribute them in the ring pattern uniformly. To generate the customers in the ring area, we first create two circles: one with a radius of 500 and one with a radius of 300. The customer nodes are randomly generated in the area outside the small circle but within the big circle. Lastly, we created a cluster problem where the customer nodes are grouped as clusters. Each cluster has a circle shape with a radius of 125 and none of the clusters are overlapping. The last two problems are designed in the way that we can evaluate solution of the routes when the customer nodes are located far away from the depot. In all problem types, we restrict that each node must be 20 units apart to prevent a situation in which two nodes locate too close or fall into the same location.

5.2. Performance of ADI on different mTSP construction heuristics in the mTSPD

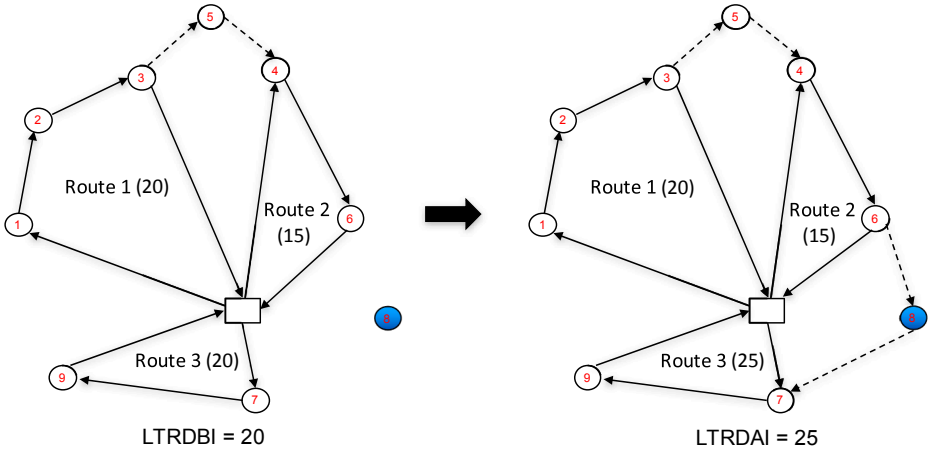
Since the mTSPD problem is a newly-defined problem, there is no solution approach with which we can evaluate the performance of our heuristic framework. One of the ways to evaluate the performance of our heuristic is to compare its solution with the solutions obtained by solving the mTSPD using MIP. In this section, we evaluate the performance of ADI under three proposed mTSP construction heuristics: Genetic Algorithm, Combined K-means/Nearest Neighbor and Random Cluster/Tour, simply referred as GA-ADI, K-means ADI, and Random ADI accordingly. The experiment was conducted using different sets of instances generated from the previous section to evaluate the performance of the ADI and compare its solutions with the best solutions from the MIP solver. Let T1, T2, T3, T4 and T5 represent the instances of the problem type 1–5 accordingly. For each instance, we set different numbers of trucks from one to five. We ran this experiment on 25 customer nodes and 50 customer nodes (medium size problem).[1] Each combination of instance was run 20 times. With 5 instances, 5 number of trucks, 3 heuristics and 2 size problem set, we generated 150 tests. Since the formulated mTSPD is NP-hard, CPLEX is not able to obtain the optimal solutions within a reasonable amount of time for the medium size problems. Therefore, we ran CPLEX for one hour (3600 s) for each instance and compare the results with the results from the proposed heuristic. We report the average objective value of each instance for all three heuristics and report the objective value found by CPLEX after running for an hour. The results are listed in Tables 1 and 2.

The results show that both GA-ADI and K-means ADI heuristics can obtain much better solution results in significantly less time compared

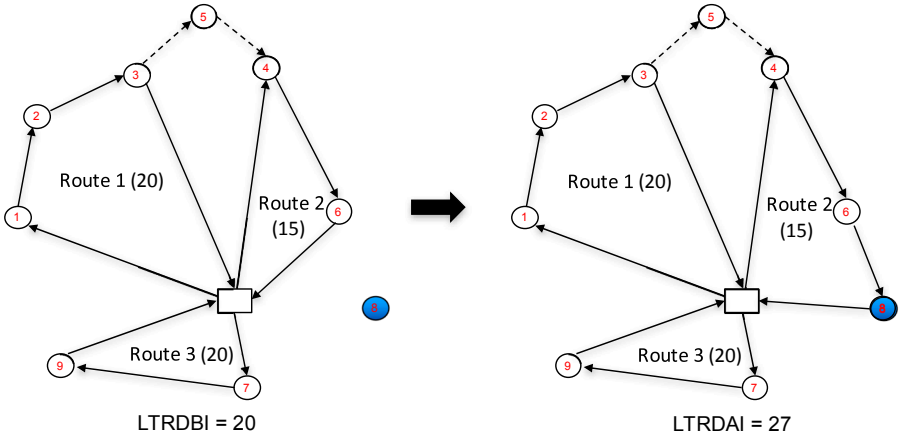
[1] The generated instances can be accessed via https://github.com/pkitjach/Drone-Problem-Sets under "Medium Size Problem" folder.



a) Illustration of DroneAddedSameRoute: $\Delta T1_j^I$ = LTRDAI – LTRDBI = 6



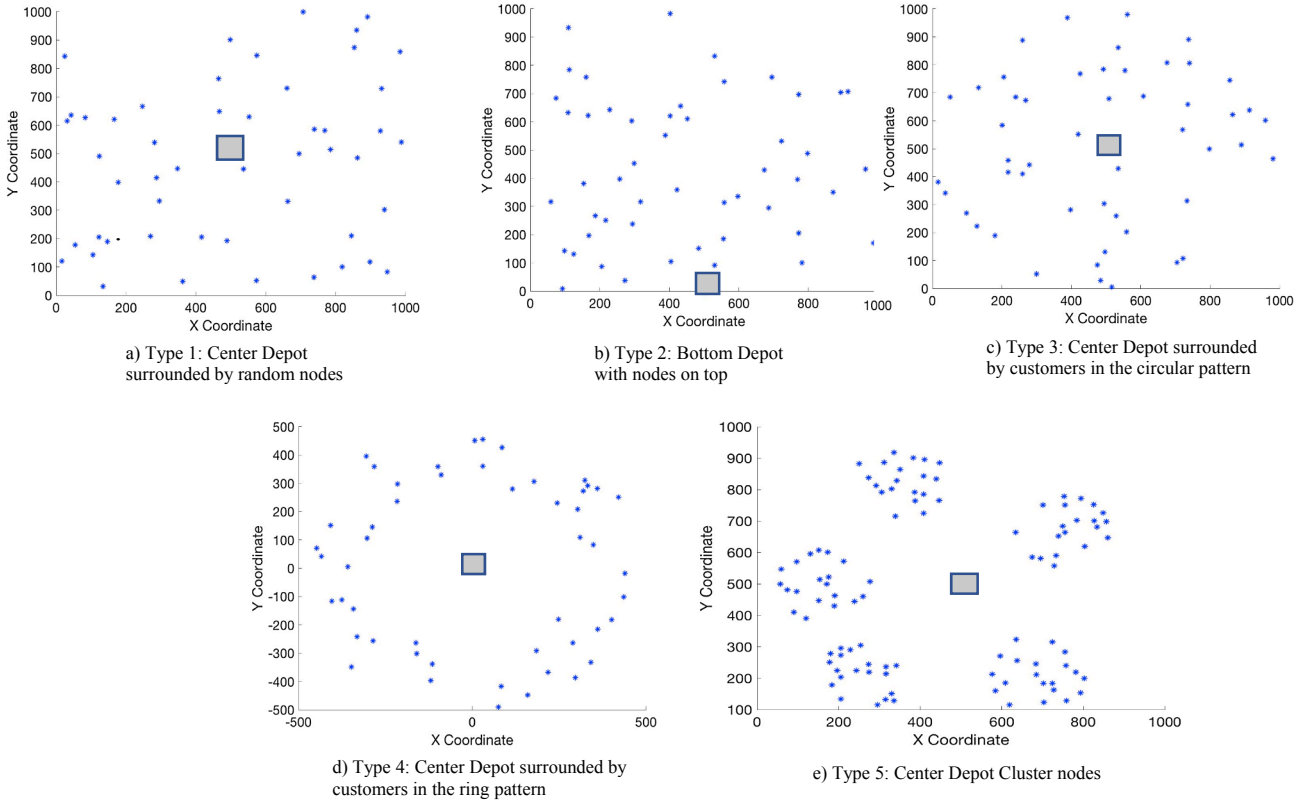
b) Illustration of DroneAddedDiffRoute: $\Delta T2_j^I$ = LTRDAII – LTRDBII = 5



c) Illustration of TruckInsertion: $\Delta T3_j^I$ = LTRDAIII – LTRDBIII = 7

Fig. 5. Illustration of all types of Insertion operators.

to CPLEX in both 25 and 50 nodes problem. On average, the GAP between these two heuristics and CPLEX is around 32–34% for 25 nodes problem and around 73–76% for 50 nodes problem. The results from the Random ADI are worse than CPLEX in term of solution quality. It can be seen that both GA-ADI and K-means ADI outperform Random ADI which indicates that the use of good mTSP heuristics can improve the quality of ADI. Comparing between GA and K-means, the former slightly performs better than the latter by providing the lower (better)



a) Type 1: Center Depot surrounded by random nodes

b) Type 2: Bottom Depot with nodes on top

c) Type 3: Center Depot surrounded by customers in the circular pattern

d) Type 4: Center Depot surrounded by customers in the ring pattern

e) Type 5: Center Depot Cluster nodes

Fig. 6. Different problem types based on the locations of depot and customer nodes.

Table 1
Comparison the results between CPLEX and the ADI performance on different mTSP construction heuristics in 25 nodes mTSPD problem.

Instance	m	MIP		GA-ADI				K-means ADI				Random ADI			
		Obj	Time (sec)	Min Obj	Avg Obj	Avg GAP from MIP	Avg Time (sec)	Min Obj	Avg Obj	Avg GAP from MIP	Avg Time (sec)	Min Obj	Avg Obj	Avg GAP from MIP	Avg Time (sec)
T1	1	3868	3600	2788	2962.60	−23.41	43.11	2909	3047.55	−21.21	32.52	4693	5277.60	78.14	30.75
	2	2524	3600	1725	1816.05	−28.05	52.25	1646	1795.65	−28.86	40.73	2977	3410.55	87.80	48.90
	3	2135	3600	1350	1451.80	−32.00	65.46	1249	1453.35	−31.93	48.43	2106	2668.30	83.79	61.45
	4	1527	3600	1172	1243.55	−18.56	72.91	1273	1296.35	−15.10	56.17	1810	2262.85	81.97	67.43
	5	1338	3600	1133	1187.80	−11.23	77.93	1136	1199.85	−10.33	62.67	1490	1985.40	67.15	75.50
T2	1	5147	3600	2918	3076.10	−40.24	43.67	3170	3258.75	−36.69	29.35	4663	5594.55	81.87	29.37
	2	3916	3600	1887	2084.20	−46.78	53.45	2503	2580.30	−34.11	41.94	2859	3743.45	79.61	48.03
	3	2772	3600	1813	1970.80	−28.90	67.01	1834	2085.20	−24.78	47.76	2574	3101.80	57.39	56.36
	4	2573	3600	1757	1915.30	−25.56	76.78	1831	1957.95	−23.90	64.21	2295	2703.05	41.13	66.26
	5	2555	3600	1768	1842.95	−27.87	80.92	1869	1930.50	−24.44	66.68	2172	2481.70	34.66	70.82
T3	1	4130	3600	2349	2518.65	−39.02	40.26	2450	2561.80	−37.97	31.34	4329	4929.00	95.70	32.97
	2	2825	3600	1580	1655.50	−41.40	50.52	1695	1790.20	−36.63	41.07	2514	3059.55	84.81	51.31
	3	1741	3600	1282	1320.50	−24.15	55.14	1177	1270.5	−27.02	47.54	1935	2444.15	85.09	62.97
	4	1974	3600	1111	1155.55	−41.46	61.53	1072	1105.65	−43.99	55.90	1767	2005.40	73.55	71.85
	5	1768	3600	1045	1096.80	−37.96	64.13	1010	1037.65	−41.31	60.78	1376	1804.70	64.54	77.83
T4	1	6079	3600	2711	2825.85	−53.51	42.11	2757	2868.80	−52.81	31.92	3884	5592.90	97.92	32.95
	2	3014	3600	1721	1770.00	−41.27	52.12	1732	1765.70	−41.42	39.34	2786	3407.45	92.51	49.46
	3	2178	3600	1376	1441.85	−33.80	66.16	1345	1397.70	−35.83	46.81	2333	2933.80	103.47	58.98
	4	1966	3600	1210	1293.00	−34.23	71.84	1185	1215.95	−38.15	53.88	2082	2553.90	97.52	70.29
	5	1949	3600	1098	1198.95	−38.48	73.53	1167	1169.70	−39.98	54.39	1816	2224.30	85.52	77.77
T5	1	3053	3600	2021	2293.10	−24.89	45.09	2080	2329.00	−23.71	29.89	3407	4138.15	80.46	30.84
	2	2379	3600	1287	1317.60	−44.62	54.58	1287	1322.90	−44.39	39.10	2210	2689.75	104.14	46.72
	3	1828	3600	1087	1081.45	−40.84	60.50	1078	1104.35	−39.59	45.67	1792	2149.55	98.77	56.81
	4	1498	3600	957	982.30	−34.43	65.63	907	965.55	−35.54	49.04	1554	1945.10	98.01	68.02
	5	1436	3600	820	846.95	−41.02	71.14	1067	1126.45	−21.56	58.35	1467	1656.15	95.54	58.35
Mean			3600			−34.15	60.31			−32.45	47.0			82.04	56.0

Table 2
Comparison the results between CPLEX and the ADI performance on different mTSP construction heuristics in 50 nodes mTSPD problem.

Instance	m	MIP		GA-ADI				K-means ADI				Random ADI			
		Obj	Time (sec)	Min Obj	Avg Obj	Avg GAP from MIP	Avg Time (sec)	Min Obj	Avg Obj	Avg GAP from MIP	Avg Time (sec)	Min Obj	Avg Obj	Avg GAP from MIP	Avg Time (sec)
T1	1	21,766	3600	3712	4049.45	−81.40	92.99	3564	4017.10	−81.54	89.11	9957	10893.3	169.01	86.38
	2	12,383	3600	2340	2500.95	−79.80	91.10	2472	2604.15	−78.97	92.21	6101	7177.30	186.98	110.75
	3	10,721	3600	1801	2031.30	−81.05	103.69	1779	1971.05	−81.62	101.6	4503	5582.60	174.83	118.90
	4	6421	3600	1580	1778.90	−72.30	117.23	1532	1899.05	−70.42	107.1	3708	4641.90	160.94	128.63
	5	5296	3600	1410	1500.65	−71.66	128.08	1433	1495.95	−71.75	118.4	3221	4093.95	172.81	135.52
T2	1	25,330	3600	3761	4297.75	−83.03	91.13	4175	4508.50	−82.20	88.45	10,746	12091.2	181.34	87.22
	2	12,942	3600	2537	2810.35	−78.29	98.80	2693	3041.35	−76.50	86.68	6091	7330.25	160.83	110.04
	3	9293	3600	2287	2428.00	−73.87	104.79	2336	2548.10	−72.58	99.46	5020	5819.15	139.67	121.06
	4	6622	3600	2134	2237.10	−66.22	106.60	2218	2349.40	−64.52	105.5	4194	5204.40	132.64	130.41
	5	5692	3600	2056	2168.50	−61.90	116.79	1970	2104.60	−63.03	111.3	3434	4686.30	116.11	137.83
T3	1	24,094	3600	3526	3883.65	−83.88	91.17	3751	4027.25	−83.29	102.9	8301	10016.1	157.91	88.25
	2	11,826	3600	2078	2310.60	−80.46	90.66	2259	2448.25	−79.30	103.4	4961	6128.70	165.24	113.16
	3	8342	3600	1595	1797.80	−78.45	98.90	1468	1794.18	−78.49	111.5	4127	4910.10	173.12	118.57
	4	6999	3600	1481	1571.40	−77.55	111.05	1406	1576.95	−77.47	111.5	3206	4098.70	160.83	130.23
	5	5469	3600	1250	1406.15	−74.29	124.85	1272	1447.85	−73.53	126.8	2935	3539.30	151.70	142.68
T4	1	23,201	3600	3344	3541.25	−84.74	94.44	3343	3570.40	−84.61	87.41	9327	11526.7	225.50	86.85
	2	11,381	3600	2110	2225.75	−80.44	92.99	2164	2315.15	−79.66	91.75	6135	7063.35	217.35	108.37
	3	8587	3600	1664	1876.40	−78.15	100.64	1791	1858.10	−78.36	96.87	4542	5355.85	185.43	115.76
	4	7306	3600	1537	1591.25	−78.22	111.35	1498	1545.50	−78.85	105.9	3583	4663.45	193.07	131.45
	5	6423	3600	1369	1403.40	−78.15	126.30	1335	1394.25	−78.29	116.5	3053	4046.60	188.34	139.74
T5	1	10,945	3600	2768	2970.35	−72.86	94.87	3254	3378.75	−69.13	86.03	7635	9032.55	204.09	89.47
	2	6566	3600	1707	1882.65	−71.33	98.76	2964	3155.55	−51.94	109.6	4486	5763.65	206.15	108.32
	3	5281	3600	1442	1511.10	−71.39	105.24	1410	1687.95	−68.04	95.12	3498	4395.40	190.87	117.51
	4	4166	3600	1239	1336.70	−67.91	110.12	1409	1456.40	−65.04	116.2	2975	4106.65	207.22	123.59
	5	3752	3600	1045	1215.30	−67.61	125.02	1394	1447.45	−61.42	117.2	2606	3478.35	186.21	145.15
Mean			3600			−75.80	105.1			−74.02	103.18			176.33	117.03

objective values in most of the instances. We also think that the solutions from GA can be improved with the right parameters on various setting. From this experiment, we decide to use GA to create the initial mTSP solution to construct the mTSPD solution in other experiments.

5.3. Comparison of different MIP models and ADI heuristic on small size problems

In this experiment, we want to evaluate the performance of the ADI heuristic when comparing with the exact solution from the MIP solver, which is only applicable to small-scale problem instances. We test the performance of the mTSP-ADI algorithm on different types of generated problems to examine the quality of the solutions. Beside comparing the solution of the ADI heuristic with the optimal mTSPD solution obtained from the MIP solver, we also compare the mTSPD solutions with solutions from the TSP and FSTSP models to evaluate the potential gain of implementing mTSPD model over the related last mile delivery models. We solved all MIP formulations using CPLEX to obtain the optimal solution.

A collection of 35 test instances[2] was generated (7 instances for each problem type, i.e. T1A, T1B,…, T1G for Type1 problem). Each instance contains 8 customer nodes and one depot node distributed in the area based on the problem type described in the previous section 5.1. For mTSPD, we only use one truck in the operation but allow multiple drones in the setting. For FSTSP, only one truck and one drone can be used in the operation. Finally, only a truck can be used in TSP model. We ran ADI heuristic 20 times for each instance and report the Best Objective, Average Objective and Average Time (seconds) of these 20 runs. We also report the Best GAP (%) and Average GAP (%) which is the difference between the Best ADI objective and the Optimal Objective and the difference between the Average ADI objective and the Optimal Objective, both in term of percentage accordingly. We are able to find the optimal objective values for TSP-MIP and mTSPD-MIP after a short period of time and report the objective value of the FSTSP-MIP solution after running the solver for 3600 s.

The column Best GAP (%) reported in Table 3 shows that ADI can find all optimal solutions (except in T5D-T5G) while consuming significantly less computational time than the mTSPD MIP formulation. The Average GAP is about less than 1% on most instances indicating that the algorithm performs quite well in all small size problems. The results of the comparison among different MIP models show that mTSPD outperforms the TSP and FSTSP with a gap value of −39.82% and −18.80% on average accordingly. These values indicate that on average the total time to complete the delivery by using one truck and multiple drones is 39% faster than the total delivery time using the truck alone and around 18.80% faster than the total delivery time using one truck and one drone. The gap is calculated by $\text{GAP} = 100\left(\frac{\text{Mean mTSPD Obj} - \text{Mean TSP / FSTSP Obj}}{\text{Mean TSP / FSTSP Obj}}\right)$ which is shown in the last two rows of the Table 3. For illustration, we graphically show the solutions obtained from the TSP, FSTSP, and mTSPD on one of the T1A instance in Fig. 7.

5.4. Robustness of proposed algorithm

In this section, we want to verify the robustness of the proposed algorithm on various settings. Using the same problem instances presented in Section 5.3, we ran the ADI on three initial mTSP construction heuristics and measure the optimality gap among the three of them. We report the GAP for each problem type (T1-T5) on each of the three mTSP heuristics: GA, K-mean/NN and Random. Additionally, we ran the algorithm 20 times on each problem, and record the standard deviation (STD) of each problem. The results are demonstrated in the Fig. 8 and Table 4 below:

The chart in Fig. 8 demonstrates that the percentage gap for ADI-GA

[2] The generated instances can be accessed via https://github.com/pkitjach/Drone-Problem-Sets under “Small Size Problem” folder.

Table 3
Comparison of the results obtained from ADI heuristic and different MIP models on small size problems.

Instance	TSP-MIP	FSTSP-MIP		mTSPD						
				MIP		ADI				
	Opt Objective	Objective	Time (sec)	Objective	Time (sec)	Best Objective	Best GAP (%)	Average	Average GAP (%)	Average Time (Sec)
T1A	2594	1702	3600	1442	111.30	1442	0	1458.35	1.13	4.01
T1B	2947	1930	3600	1422	60.40	1422	0	1423.65	0.12	4.25
T1C	2488	1764	3600	1350	62.42	1350	0	1350.00	0.00	4.61
T1D	2800	1929	3600	1469	63.63	1469	0	1469.00	0.00	7.84
T1E	2655	1673	3600	1193	112.22	1193	0	1198.30	0.44	5.63
T1F	1773	1325	3600	1105	83.45	1105	0	1105.00	0.00	5.44
T1G	2255	1541	3600	1186	67.49	1186	0	1186.00	0.00	7.53
T2A	2578	2050	3600	1932	85.33	1932	0	1937.90	0.31	5.26
T2B	3036	2130	3600	1662	92.45	1662	0	1662.00	0.00	5.38
T2C	2498	2143	3600	1725	87.40	1725	0	1733.95	0.52	5.06
T2D	2702	2315	3600	1721	76.58	1721	0	1721.00	0.00	5.76
T2E	2703	2206	3600	1509	67.54	1509	0	1509.00	0.00	4.83
T2F	2989	2205	3600	1880	73.44	1880	0	1882.85	0.15	5.82
T2G	2633	2221	3600	1925	74.63	1925	0	1925.00	0.00	8.08
T3A	2849	1668	3600	1181	86.20	1181	0	1181.00	0.00	5.07
T3B	2556	2027	3600	1481	73.68	1481	0	1498.85	1.21	5.57
T3C	1832	989	3600	647	80.30	647	0	647.35	0.05	5.44
T3D	2383	1650	3600	1261	71.69	1261	0	1272.75	0.93	4.46
T3E	2324	1589	3600	1148	118.45	1148	0	1148.00	0.00	5.29
T3F	2615	1642	3600	1426	73.44	1426	0	1426.00	0.00	5.74
T3G	2576	1710	3600	1509	76.01	1509	0	1512.90	0.26	6.41
T4A	2479	1880	3600	1613	77.60	1613	0	1636.45	1.45	3.97
T4B	2625	2054	3600	1563	71.27	1563	0	1575.45	0.80	4.32
T4C	2005	1543	3600	1481	79.88	1481	0	1498.85	1.21	5.57
T4D	2428	1726	3600	1548	78.27	1548	0	1552.50	0.29	4.46
T4E	2348	1848	3600	1589	97.25	1589	0	1613.60	1.55	6.27
T4F	2699	2123	3600	1426	74.13	1426	0	1426.00	0.00	5.74
T4G	1942	1667	3600	1358	74.47	1358	0	1360.25	0.17	7.02
T5A	1405	1136	3600	1066	93.22	1066	0	1066.00	0.00	5.70
T5B	1322	1059	3600	944	135.60	944	0	944.00	0.00	4.54
T5C	1322	1015	3600	878	131.76	878	0	878.00	0.00	4.67
T5D	1348	1190	3600	1133	123.22	1138	0.44	1138.50	0.49	4.68
T5E	1351	1056	3600	1008	77.90	1014	0.60	1018.55	1.05	4.31
T5F	1579	1315	3600	1270	95.32	1283	1.02	1287.80	1.40	4.38
T5G	1518	1383	3600	1188	140.21	1202	1.18	1215.80	2.34	3.75
Mean	2290	1697		1378		1379		1385		
GAP (%) from TSP	–	−25.89		−39.82						
GAP (%) from FSTSP	34.94	–		−18.80						

is zero for all types of problems which indicates that the algorithm is able to find the optimal solution in the small size problems. On the contrary, ADI with Kmean-NN performs quite poorly with the optimal gap ranging from 0% to 13% depending on the problem type. It is possible that the search starting with mTSP solution from Kmean-NN often gets stuck with the local optimal solution and can only reach the global optimal solution in only certain types of problems. Lastly, the ADI is actually performing quite well when starting with random mTSP solutions in the small size problem partly because of the small search space based on the problem size. From the relative standard deviation table, the standard deviation (STD) is relatively small ($< 2\%$) compared to the average value for all problem types. These STD values are higher for ADI with Kmean-NN and Random heuristics accordingly. Therefore, this experiment shows that the ADI-GA algorithm is robust against different problem types from the consistent zero optimality gap and small standard deviation.

5.5. Performance of the larger instances on benchmark problem

In this experiment, we evaluate the performance of the ADI heuristic on the well-known TSP/mTSP benchmark problems from TSPLIB (Reinelt, 1991) with the min-max objective function. Each instance was solved with mTSPD-ADI with the change of the number of trucks {1, 2, 3 and 5}. We ran 20 replications for each instance with different numbers of trucks. We compare the average objective value obtained from the proposed ADI algorithm with the optimal objective of min-max TSP/mTSP for each instance. Since the TSP/mTSP does not allow the drone in the operation, the purpose of this experiment is to show how much operational time we could save by implementing multiple drones along with multiple trucks delivery strategy. We also tested the same set of instances with the Adjusted FSTSP heuristic and compared the performance of the algorithm with ADI.

The results are shown in Table 5. Overall, both the mTSP-ADI and Adjusted FSTSP heuristics return lower objectives value than the mTSP optimal values, as indicated by the negative values of GAP (%) from Optimal mTSP column. On average, the ADI heuristic returns the average gap lower than the min-max TSP/mTSP optimal solution by 20.95%. The gap is calculated by $GAP = 100\left(\frac{Avg\ \ Obj - Optimal\ \ Obj\ \ of\ \ TSP/mTSP}{Optimal\ \ Obj\ \ of\ \ TSP/mTSP}\right)$. Similarly, the Adjusted FSTSP heuristic returns the solutions better than the min-max TSP/mTSP optimal solutions by 8.14%. This finding demonstrates that assigning drones along with trucks to make deliveries is more effective than purely using trucks to make deliveries. Although the heuristic might not return the optimal solution, the high average gap still shows that it is more beneficial to implement multiple drones in the operation. It can be seen that the effect of drones become less significant once the number of trucks increase indicated by the GAP closer to zero. In addition, the ADI heuristic obtains a lower gap value than the Adjusted FSTSP heuristic's gap (about 13.84% on average), which can be noticed in the last column of Table 5. This value indicates that allowing multiple drones in delivery operations can significantly accelerate

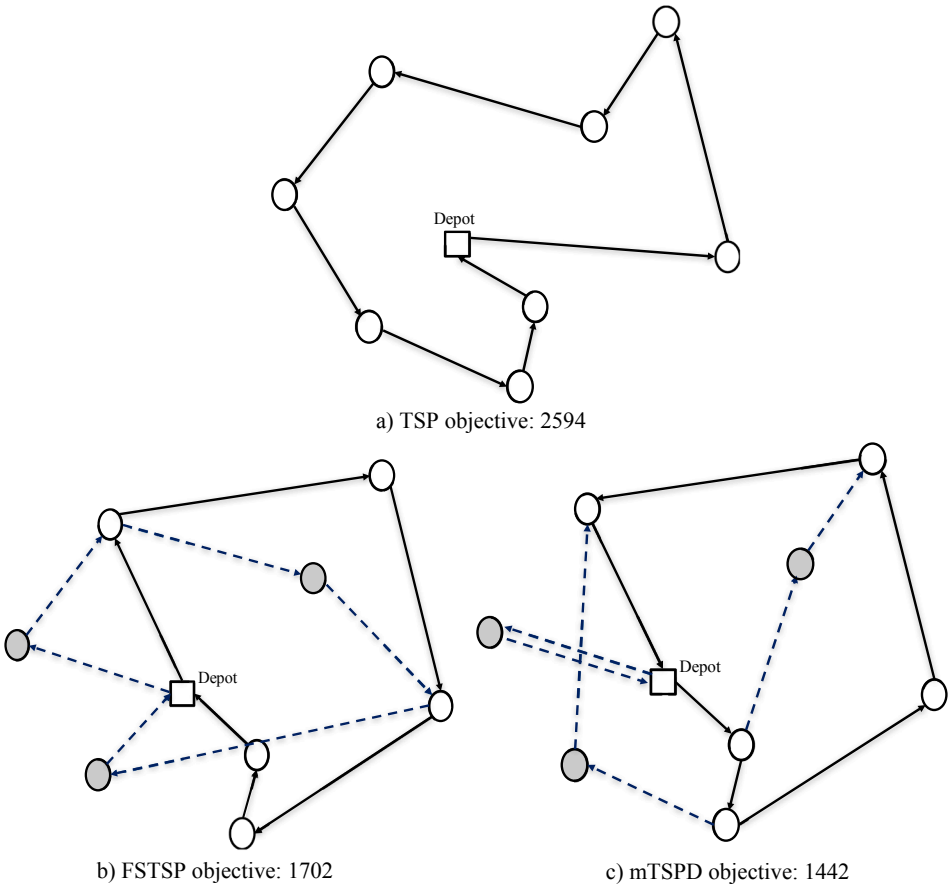


Fig. 7. The illustration of the T1A solution obtained by CPLEX for all model formulation. The solid lines represent truck paths and dot lines represent drone paths.

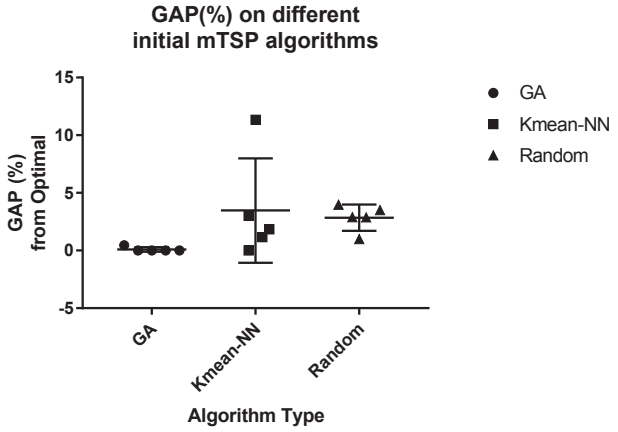


Fig. 8. Optimal GAP(%)of ADI on different mTSP construction heuristics.

Table 4
Relative standard deviation of each problem type among.

Problem Type	Relative STD (%)		
	GA	Kmean-NN	Random
T1	0.51	2.37	7.58
T2	0.37	0.59	5.69
T3	1.04	1.74	8.45
T4	1.79	3.32	6.43
T5	0.53	0.91	3.36

deliveries. In term of computational time, Fig. 9a) shows that the runtime of the ADI heuristic rises quickly as the number of nodes increases. Similarly, Fig. 9b) demonstrates the increase in the runtime once the number of trucks increase on different instances.

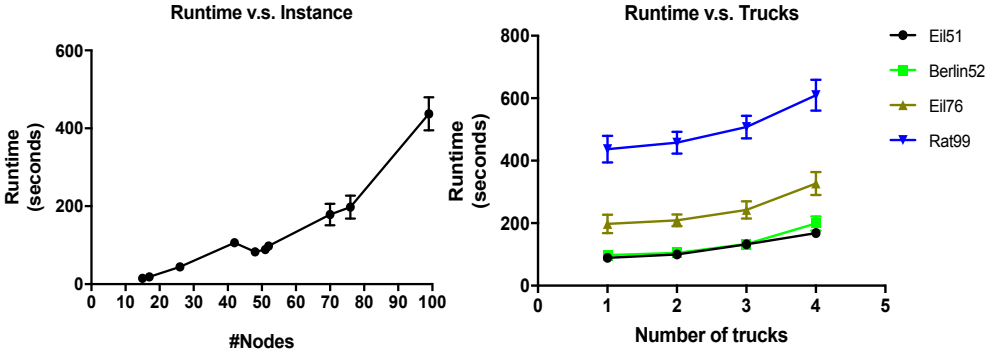
6. Conclusions and future work

The work presented in this paper presents an insightful study on the application of drones in last mile delivery. It primarily focuses on the truck drone routing problem with the goal to reduce the transportation time in the last mile delivery which could potentially help the companies improve their operational decisions as the e-commerce trend is growing. We propose a new routing model, the Multiple Traveling Salesman Problem with Drones (mTSPD), which implements both trucks and drones in the last mile delivery. The model is a variation of the classic TSP problem and the extension of the previous FSTSP model. We generalize the FSTSP to the mTSPD in which we allow multiple drones and multiple trucks to perform deliveries. The MIP formulation is mathematically constructed to model the mTSPD based on the delivery scenario described in Section 2 to solve for the solution optimally.

Because of the NP-hardness of the mTSPD, an Adaptive Insertion algorithm (ADI) which builds up the tours from the initial mTSP solution, has been proposed. The algorithm consists of two phases: building the mTSP solutions and applying removal and insertion operators to the initial mTSP solution to construct the mTSPD solution. We have proposed three mTSP construction heuristics: Genetic Algorithm, Combined K-means/Nearest Neighbor and Random Cluster/Tour. The experiment in Sections 5.2 and 5.4 of the paper shows that GA can find better solutions within acceptable computation times and provide the lowest standard deviation when testing with different problem sets.

Table 5
Results of the ADI and Adapted FSTSP heuristics solutions on the benchmark problems. The negative sign on GAP (%) indicates a better objective value from the optimal objective of TSP/mTSP on both ADI and Adapted FSTSP Heuristic.

Instance	# Trucks (m)	Optimal Objective for TSP/mTSP	ADI				Adapted FSTSP Heuristic				GAP (%) between heuristics
			Avg Obj	GAP (%) from Optimal mTSP	Avg Customers served by Drones	Avg Time (s)	Avg Obj	GAP (%) from Optimal mTSP	Avg Customers served by Drones	Avg Time (s)	
p01	1.00	284.38	203.92	−28.29	5.00	15.03	246.17	−13.43	2.75	7.42	17.16
gr17	1.00	2085.00	1494.63	−28.31	5.45	18.85	1833.85	−12.05	1.80	7.71	18.50
fri26	1.00	937.00	732.55	−21.82	8.65	44.35	857.15	−8.52	2.55	9.05	14.54
dantzig42	1.00	699.00	475.07	−32.04	14.00	80.38	632.61	−9.50	3.70	13.72	24.90
att48	1.00	33524.00	22485.09	−32.93	16.00	82.87	30291.50	−9.64	3.55	15.92	25.77
st70	1.00	675.00	475.01	−29.63	23.00	178.57	633.21	−6.19	4.10	43.69	24.98
eil51	1.00	426.00	292.35	−31.37	17.00	88.90	380.6	−10.66	3.80	14.27	23.19
	2.00	222.73	173.44	−22.13	16.55	99.93	198.80	−10.74	5.10	15.79	12.76
	3.00	159.57	137.37	−13.91	15.15	132.45	140.10	−12.20	6.60	16.48	1.95
	5.00	123.96	108.64	−12.36	14.25	168.08	109.90	−11.34	8.10	16.57	1.15
berlin52	1.00	7542.00	4818.15	−36.12	17.00	97.82	6882.37	−8.75	3.65	24.23	29.99
	2.00	4110.21	3003.30	−26.93	16.40	104.96	3612.04	−12.12	4.75	20.71	16.85
	3.00	3244.37	2508.66	−22.68	15.30	133.25	2704.92	−16.63	6.55	21.60	7.26
	5.00	2440.92	2069.87	−15.20	13.90	198.67	1975.74	−19.06	8.05	21.11	−4.76
eil76	1.00	538.00	374.23	−30.44	25.00	197.97	513.59	−4.54	3.80	40.34	27.13
	2.00	280.85	230.83	−17.81	24.55	209.10	278.31	−0.90	5.30	48.27	17.06
	3.00	197.34	185.70	−5.90	23.40	242.00	196.90	−0.22	7.00	57.77	5.69
	5.00	150.30	148.05	−1.50	21.10	326.84	144.48	−3.87	10.40	75.42	−2.47
rat99	1.00	1211.00	946.36	−21.85	33.00	437.30	1225.08	1.16	4.25	94.25	22.75
	2.00	728.71	643.71	−11.66	31.70	457.58	736.68	1.09	5.80	157.56	12.62
	3.00	597.55	546.93	−8.47	30.00	507.87	581.67	−2.66	8.05	96.68	5.97
	5.00	531.87	480.72	−9.62	27.35	609.80	487.56	−8.33	11.15	97.29	1.40
Mean			−20.95				−8.14			13.84	



a) The Runtime v.s. Number of nodes b) The Runtime v.s. Trucks plot

Fig. 9. Run time based on (a) the number of nodes and (b) the number of trucks.

Five types of problems based on the nodes distribution and depot location are generated to verify the proposed MIP formulation and the ADI algorithm performance. Results revealed that in small size generated problems, both solvers and GA-ADI reached to optimal solution but GA-ADI solved the problem significantly faster than CPLEX. In addition, by comparing the results of the mTSPD with other existing last mile delivery models (TSP/FSTSP), we can see that the delivery time can be significantly improved by using multiple drones in the route planning.

In large size instances, we conduct another set of experiment in which we use our proposed heuristic to solve some of the benchmark min-max TSP/mTSP instances from TSPLIB. The obtained solutions are compared with the optimal solution of these benchmark problems and the best found solution from the Adapted FSTSP heuristic (allowing one drone per truck and the drone has to return to the original truck it is launched from). We find that the algorithm returns solutions with a lower average objective when compared with the optimal objective of the truck-only operation the ones solved by an Adapted FSTSP heuristic. The experimental results show that the proposed model using multiple drones and trucks along with the heuristic provides shorter delivery completion time than simply using trucks alone, multiple trucks, and a single truck-drone in the operation.

Future research could integrate other realistic constraints in the current model, such as time windows, endurance, cost, and capacity when operating drones in a different circumstance. From an algorithmic perspective, we look forward to more effective algorithms (e.g. Adaptive Large Neighborhood Search) to solve the mTSPD problem with a better objective value and lower computational time. It would be interesting to solve the model with other metaheuristic algorithms like simulated annealing or memetic algorithms. Lastly, we expect to work on developing a new Vehicle Routing Problem with Drones (VRPD) by extending the proposed mTSPD to take into account the truck capacity as well as the drone capacity.

References

Adams, E. (2017). A stronger, faster predator drone is headed for european airspace. https://www.wired.com/2016/12/stronger-faster-predator-drone-soon-prowl-europes-airspace/.

Agatz, N., Bouman, P., & Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science*.
Arora, K., Agarwal, S., & Tanwar, R. (2016). Solving TSP using genetic algorithm and nearest neighbour algorithm and their comparison. *International Journal of Scientific & Engineering Research, 7*(1), 1014–1018.
Balcik, B., Beamon, B. M., & Smilowitz, K. (2008). Last mile distribution in humanitarian relief. *Journal of Intelligent Transportation Systems, 12*(2), 51–63.
Bamburry, D. (2015). Drones: Designed for product delivery. *Design Management Review, 26*, 40–48.
Bektas, T. (2006). The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega, 34*(3), 209–219.
Berg, M. D. (1991). On rectilinear link distance. *Computational Geometry, 1*(1), 13–34.
Bertazzi, L., Golden, B., & Wang, X. (2015). Min-max vs. min-sum vehicle routing: A worst-case analysis. *European Journal of Operational Research, 240*(2), 372–381.
Bonnington, C. (2017). President trump wants to make drone delivery happen. Don't expect much (besides crashes). http://www.slate.com/blogs/future_tense/2017/10/27/drone_ delivery_isn_t_on_its_way_even_after_president_trump_s_latest_directive.html.
Bouman, P., Agatz, N., & Schmidt, M. (2018). Dynamic programming approaches for the traveling salesman problem with drone. *Networks, 72*(4), 528–542.
Brar, S., Rabbat, R., Raithatha V., Runcie, G., Yu A. (2015) Drones for deliveries. http://scet.berkeley.edu/wp-content/uploads/ConnCarProjectReport-1.pdf.
Browne, T. (2017). Police drones market increases with FAA Rules, test cases, robotics business review. https://www.roboticsbusinessreview.com/security/police-drones-market-increases-faa-rules-test-cases/.
Buchmueller, D., Green, S., Kalyan, A., & Kimchi, G. (2016). U.S. patent no. US20160257401A1. Washington, DC: U.S. Patent and Trademark Office.
Campbell, J., Sweeney, D., & Zhang, J. (2017). Strategic design for delivery with trucks and drones (Tech.).
Campbell, A. M., Vandenbussche, D., & Hermann, W. (2008). Routing for relief efforts. *Transportation Science, 42*(2), 127–145.
Carlsson, J. & Song, S. (2017). Coordinated logistics with a truck and a drone. http://www-bcf.usc.edu/~jcarlsso/horseflies.pdf.
Carter, A. E., & Ragsdale, C. T. (2006). A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European Journal of Operational Research, 175*(1), 246–257.
Daknama, R., & Kraus, E. (2017). Vehicle routing with drones. arXiv preprint arXiv:1705.06431.
Desrochers, M., & Laporte, G. (1991). Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters, 10*(1), 27–36.
Dorling, K., Heinrichs, J., Messier, G. G., & Magierowski, S. (2017). Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47*(1), 70–85.
Dorr L. & Duquette, A. (2016). Press release – DOT and FAA finalize rules for small unmanned aircraft systems. https://www.faa.gov/news/press releases/news_story.cfm?newsId = 20515.
Drexl, M. (2012). Synchronization in vehicle routing—A survey of VRPs with multiple synchronization constraints. *Transportation Science, 46*(3), 297–316.
Ferrandez, S. M., Harbison, T., Weber, T., Sturges, R., & Rich, R. (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management, 9*(2), 374.
Gendreau, M., Hertz, A., & Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research, 40*(6), 1086–1094.
Gentry, N., Hsieh, R., & Nguyen, L. (2016). U.S. patent no. 9527605. Washington, DC: U.S. Patent and Trademark Office.
Glaser, A. (2017). Watch Amazon's Prime Air make its first public U.S. drone delivery. https://www.recode.net/2017/3/24/15054884/amazon-prime-air-public-us-drone-delivery.
Goel, A., & Kok, L. (2012). Truck driver scheduling in the United States. *Transportation Science, 46*(3), 317–326.
Gross, D. (2013). Amazon's drone delivery: How would it work? http://www. cnn.com/2013/12/02/tech/innovation/amazon-drones-questions/index.html.
Ha, Q., Deville, Y., Pham, Q., & Hà, M. (2018). On the min-cost traveling salesman problem with drone. *Transportation Research Part C, 86*, 597–621.
Heath, N. (2015). The long-range drone that can keep up with a car and fly for an hour. https://www.techrepublic.com/blog/european-technology/the-long-range-drone-that-can- keep-up-with-a-car-and-fly-for-an-hour/.
Hern, A. (2014). DHL launches first commercial drone 'parcelcopter' delivery service. https://www.theguardian.com/technology/2014/sep/25/german-dhl-launches-first commercial-drone-delivery-service.
Hong, I., Kuby, M., & Murray, A. (2017). A deviation flow refueling location model for continuous space: A commercial drone delivery system for Urban Areas. *Advances in Geocomputation Advances in Geographic Information Science,* 125–132.
Huang, M., Smilowitz, K., & Balcik, B. (2011). Models for relief routing: Equity, efficiency and efficacy. *Procedia – Social and Behavioral Sciences, 17*, 416–437.
Hughes, M. (2017). UPS successfully delivered a package with a drone. https://thenextweb.com/insider/2017/02/21/ups-successfully-delivered-package-drone/.
Jain, A. K. (2010). Data clustering: 50 Years beyond K-means. *Pattern Recognition Letters, 31*(8), 651–666.
Kivelevitch, E. H., Cohen, K., & Kumar, M. (2013). *A binary programming solution to the min-max multiple-depots, multiple traveling salesman problem. AIAA Infotech@Aerospace (I@A) conference*.
Lee, H., Chen, Y., Gillai, B., & Rammohan (2016). Technological disruption and innovation in last-mile delivery. https://www.gsb.stanford.edu/sites/gsb/files/publication-pdf/vcii-publication-technological-disruption-innovation-last-mile-delivery.pdf.
Li, J., Sun, Q., Zhou, M., & Dai, X. (2013). *A new multiple traveling salesman problem and its genetic algorithm-based solution. 2013 IEEE international conference on systems, man, and cybernetics*.
Lu, Q., & Dessouky, M. M. (2006). A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research, 175*(2), 672–687.
Mathew, N., Smith, S. L., & Waslander, S. L. (2015). Planning paths for package delivery in heterogeneous multirobot teams. *IEEE Transactions on Automation Science and Engineering, 12*(4), 1298–1308.
McFarland, M. (2017). U.S. government warms up to drone delivery. http://money.cnn.com/2017/10/25/technology/business/drones-trump-local/index.html.
Meola, A. (2016). The FAA just put up a major roadblock to widespread drone usage. http://www.businessinsider.com/faa-wont-change-airspace-rules-for-drones-until-2019- 2016-5.
Murray, C. C., & Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies, 54*, 86–109.
Patil, S. (2016). Drone delivery: A game changer for logistics space? http://techstory.in/drone-delivery/.
Perez, S., & Kolodny, L. (2017). UPS tests show delivery drones still need work. https://techcrunch.com/2017/02/21/ups-tests-show-delivery-drones-still-need-work/.
Poikonen, S., Wang, X., & Golden, B. (2017). The vehicle routing problem with drones: Extended models and connections. *Networks, 70*(1), 34–43.
Ponza, A. (2015). Optimization of drone-assisted parcel delivery. http://tesi.cab.unipd.it/51947/1/Andrea_Ponza_Analisi_e_ottimizzazione_nell%27uso_di_droni_ per_la_consegna_di_prodotti.pdf.
Rao, B., Gopi, A. G., & Maione, R. (2016). The societal impact of commercial drones. *Technology in Society, 45*, 83–90.
Reinelt, G. (1991). TSPLIB—A traveling salesman problem library. *ORSA Journal on Computing, 3*(4), 376–384.
Rose, C. (2013). Amazon's Jeff Bezos looks to the future. https://www.cbsnews.com/news/amazons-jeff-bezos-looks-to-the-future/.
Sedighpour, M., Yousefikhoshbakht, M., & Darani, N. (2011). An effective genetic algorithm for solving the multiple traveling salesman problem. *Journal of Optimization in Industrial Engineering, 8*, 73–79.
Smith, D. (2013). Amazon prime air: 5 Major weaknesses of the proposed drone delivery service. http://www.ibtimes.com/amazon-prime-air-5-major-weaknesses-proposed-drone-delivery-service-1491978.
Ungerleider, N. (2016). The technology that gets a package from the warehouse to your house. https://www.fastcompany.com/3066534/the-technology-that-gets-a-package-from-the-warehouse-to-your-house.
Vincent, J. (2017). Google's Project Wing has successfully tested its air traffic control system for drones. At: https://www.theverge.com/2017/6/8/15761220/google-project-wing-drone-air-traffic-control-tests.
Wang, X., Poikonen, S., & Golden, B. (2016). The vehicle routing problem with drones: Several worst-case results. *Optimization Letters, 11*(4), 679–697.
Zito, D. & Radocaj, L. (2016). First test delivery of the workhorse group horsefly drone featured on NBC-TV affiliate. https://globenewswire.com/news-release/2016/01/07/800262/0/en/First-Test-Delivery-of-the-Workhorse-Group-HorseFly-Drone-Featured-on-NBC-TV-Affiliate.html.