2020

Traveling salesman problem with time windows and drones (TSPTWD)

Bo Lan
Iowa State University

Traveling salesman problem with time windows and drones (TSPTWD)

by

Bo Lan

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Business and Technology (Supply Chain Management)

Program of Study Committee:
Yoshinori Suzuki, Major Professor
Johanna Amaya Leal
Michael Crum
Kevin Scheibe
Lizhi Wang

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2020

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor and committee chair, Dr. Yoshinori Suzuki for his continuous support. Without his mentoring, motivation, and advising I cannot finish my challenging and exciting journey to pursue a Ph.D. degree. My thanks also go to my committee members, Dr. Johanna Amaya Leal, Dr. Michael Crum, Dr. Kevin Scheibe, and Dr. Lizhi Wang. Their suggestions and comments give me more insightful thoughts to improve my research.

I also want to thank my wife Feng Xue and our daughter Yiqing (Merrina) Lan for their understanding and support.

ABSTRACT

In this dissertation, I study a relatively novel variant form of traveling salesman problem (TSP), i.e. traveling salesman problem with time windows and a drone (TSPTWD) or multiple drones (TSPTWmD) with intermediate points. Mathematical models and metaheuristics were successfully developed, and numerical experiments showed promising savings of delivery time versus conventional modes.

The first essay (Chapter 2) is an extension of one of the earliest studies of this topic which established a traveling salesman model of a truck and a drone. That model only allows the drone to be operated on customers' sites. I developed a new model to solve a novel variant of TSP, i.e. traveling salesman problem with time windows and a drone (TSPTWD) which can be operated on customers' sites and intermediate points. Computational experiments have been implemented to test my new model. The results of small size problems (less than seven customers) showed that my new model can save delivery time as large as 42.8% than the traditional pure truck TSP model. Comparing with the existing model of a truck and a drone operated only on customers' sites, my new model further increased the saving as large as 2.85%. And that saving could be even larger if I implement my model on larger size problems (with more customers).

Due to the NP-hardness, only small size problems (less than seven customers) can be solved exactly in a practical period (less than one hour). Some LP heuristics can provide relatively good solutions within a shorter time. But that computing time will also increase too large with growing problem size. Then I developed other heuristics or metaheuristics to solve large size problems within an acceptable period in the second essay.

In the second essay (Chapter 3) I developed multiple heuristics and metaheuristics to solve TSPTWD problems with as many as 100 customers which are likely the size of problems in the real world. At first, the problem of poor-quality solutions had been met and a series of improvement strategies were initiated. Those strategies were developed based on the specific nature of this novel problem TSPTWD. When those strategies were implemented the metaheuristics generated solutions with much higher quality.

Numerical experiments were designed and implemented to test the developed metaheuristics and the impact of factors. The results showed that the algorithm SA_01 outperformed SA_02 and the naïve method of TSPTWD. My model can save as large as 26% of the delivery time than the traditional pure truck TSP. The idea of my research to provide intermediate points on arcs did generate more delivery time savings than only operating the drone on customers' sites. That improvement could be as large as 4%. The results also showed that lower customer density provides more opportunities for delivery time saving.

In the third essay (Chapter 4) I developed a new mathematical model of traveling salesman problem with time windows and multiple drones (TSPTWmD) which can be operated on customers' sites and intermediate points and two metaheuristics, i.e. simulated annealing 02, 03 (SA_02, SA_03) to solve TSPTWmD problems as large as 100 customers which is similar to the size of problems in the real world.

The improvement strategies of essay 2 were implemented and multiple drones were incorporated into the new metaheuristics. Computational experiments were designed and implemented to test the developed metaheuristics and the impact of factors. The results showed that algorithm SA_03 outperformed SA_02 and it is even better than SA_01. The delivery time saving can be as large as 32.1% comparing with the traditional pure truck TSP. The idea of my

research to provide multiple drones did generate more delivery time savings than only using one drone. The results show that the delivery time could be further reduced as large as more than 5%. The impact of factors is similar as demonstrated in Chapter 3 that lower customer density provides more opportunities for delivery time saving.

CHAPTER 1. GENERAL INTRODUCTION

Introduction

In recent decades people have realized that we should pursue not only economic profits but also sustainable developments in our business activities (Butlin, 1989). Transportation is an important part of modern business and logistics operations. Traditional transportation research has been developed for almost six decades. Lots of exact algorithms and heuristics are developed to solve those problems (Laporte, 2009). In recent years urban logistics receives more attention than before because more than half of the people on the earth are living in urban areas (Donath, 2014). Increased usage of individual transportation along with the decreased usage of public transportation leads to more problems of congestion and pollution in urban areas (Rose et al., 2017b, Figliozzi, 2011).

It is shown that over 50% of the world population are living in metropolitan areas, consume about 75% of worldwide energy, and emit 80% greenhouse gasses. By 2050, those areas will house about 70% of all people (Princeton, 2020). Urban living brings convenience to residents but also leads to negative impacts, e.g. congestion and pollution, etc. At the same time, the rapid development of e-commerce raises new challenges to the last-mile delivery since they need more and more direct shipment to consumers (Savelsbergh and Van Woensel, 2016, Durand et al., 2013). Some improvements have been made to mitigate those problems, e.g. leveraging the power of big data to improve city logistics (OECD, 2015), deploying alternative fuel vehicles(AFV) to reduce pollution (Erdoğan and Miller-Hooks, 2012), implementation of off-hour or night deliveries to reduce congestion and increase efficiency (Holguín-Veras et al., 2011, Fu and Jenelius, 2017), crowdsourced logistics or sharing economy business models augmenting urban distribution capability (Castillo et al., 2018). Some emerging new

technologies have been used to solve those problems one of which is the so-called unmanned aerial vehicles (UAV) or drones (Savelsbergh and Van Woensel, 2016).

Drone deliveries could become a novel, promising mode of deliveries in modern cities. For example, Amazon CEO Jeff Bezos says 86 percent of the products Amazon delivers today weigh five pounds or less (Guglielmo, 2017). Five pounds is a common payload capacity of a normal commercial drone. Some of them can even deliver stuff as heavy as 10 pounds (Perez and Kolodny, 2017). That means most parcels' last-mile deliveries might be accomplished by drones. Due to the faster speed (Transport Topics, 2017), cheaper cost (Keeney, 2015), and lower emission (Goodchild et al., 2018) of drones, we can expect multiple benefits by deploying drones in the future logistics. Some business has begun to test deliveries by drones. For instance, Amazon initiated its first prime air delivery in 2016 (Weise, 2016) and Figure 1(a) (McNabb, 2016), DHL delivered medication to an island since September 2014 (Hern, 2014), UPS arranged a last-mile delivery test with a drone in February 2017 as shown in Figure 1(c) (Weise, 2017). The drone can be launched and received at any appropriate point on the road to help to deliver parcels to customers. Considering about 66,000 UPS drivers on the road each day, about $50 million will be saved if only one mile is reduced per driver per day over a year (Transport Topics, 2017). FAA has granted the first commercial drone delivery license to UPS (UPS, 2019). Some analyses showed that there are a fast-growing parcel delivery market and great opportunities for drones. It is estimated that 107 billion parcels were delivered in 2019 for revenue of $350 billion, which would be increased to 289 billion parcels in 2030 for revenue of $665 billion. Automated last-mile deliveries are projected to yield $48 billion in revenue by 2030 and account for about 20% of parcel deliveries (Martin, 2020). The report indicates that there are four categories of auto last-mile deliveries, i.e. drones (with vehicles), legged robots, wheeled

robots, and autonomous vehicles. Even drones only deliver a quarter of the estimated volume, it is a market of about $12 billion and 14 billion parcels. All of these indicate an emerging new mode of transportation- drone assisted parcel delivery. Due to the fast speed, low energy consumption, and low cost of drones, their involvement in the last-mile logistics should be able to reduce both cost and pollution. The research on scheduling such a new mode of transportation must be necessary.

(a) (b) (c)

Figure 1. Drones were tested or permitted to delivery parcels

Literature Review

Urban Logistics and Last-Mile Deliveries

United Nations predicts an additional 2.5 billion urban dwellers in the next 35 years. Urban evolution over the past half-century has led to a need for new urban logistics said Rose et al. (2017a). The authors pointed out several future research directions, e.g. resource management, vehicle routing problems, network design, inventory management, etc. Last-mile logistics is regarded as the last part of supply chains to provide door to door service(Lee and Whang, 2001). It is also considered as the most expensive and least efficient part of supply chains (Gevaers et al., 2011).

Traveling Salesman Problem

As an important part of the last-mile delivery, parcel delivery is typically modeled as a traveling salesman problem (Don (2013)). In this classical problem, a vehicle is required to start from a point to visit all other points one time and only one time and finally come back to the starting point. The goal of the problem is to find the shortest distance path to finish the above-described journey. This problem has been proved as an NP-hard problem (Karp, 1972). It has received a lot of attention from researchers and a variety of algorithms have been developed. There are exact algorithms (Christofides, 1970, Held and Karp, 1971, Carpaneto and Toth, 1980, Balas and Christofides, 1981, Miller and Pekny, 1991, Applegate, 2006) and heuristics (Lin and Kernighan, 1973, Kirkpatrick et al., 1983, Glover, 1977, Helsgaun, 2000). Good reviews of this problem can be found in Laporte (1992), Reinelt (1994) and, Applegate (2006).

Truck and Drones Deliveries

Using drone(s) assisted trucks to deliver parcels is a relatively new research topic. Murray and Chu (2015) is one of the earliest research to introduce a variant of the traditional traveling salesman problem (TSP) within which a drone works together with a truck to deliver parcels and the delivery time can be reduced. The authors set up a mixed-integer programming model to describe the problem and called it flying sidekick TSP (FSTSP). They developed several heuristics to solve those problems. The drone can deliver some parcels for the truck to let it go back to the depot earlier (*Figure 2* left).

Dorling et al. (2017) developed a model to solve a delivery problem of a fleet of drones based on a depot. It is similar to the traditional vehicle routing problem but replaces conventional vehicles by drones. They considered the cost of drones and solved the problem to minimize cost or minimize total delivery time subject to a budget constraint.

Agatz et al. (2018) examined a traveling salesman problem whose vehicle is assisted with a drone to implement last-mile deliveries. They considered the situation that the drone may launch and recover on customers' sites. An ILP model like the set partitioning model was developed. They made a theoretical analysis to prove the lower bound of saving. Several heuristics are also proposed to solve large size problems, such as a route first and cluster second algorithm.

Wang et al. (2017) discussed mathematical models of vehicle routing problems with drones (VRPD). One truck takes multiple drones that do not change their hosts. The authors derived and proved a few worst-case results on VRPD to demonstrate the maximum savings that can be obtained from using drones. Their research assumed that drones travel the same street network as trucks rather than Euclidean distances. Drone battery's capacity was considered as arbitrarily large and drones can only be released and picked up at customers' sites.

A research of Goodchild et al. (2018) compared two delivery models, i.e. only by trucks or only by drones. They focused on the level of CO2 emission and found that traditional vehicles' emission could be as high as seventy times of drones.

To my best knowledge, there are no studies about utilizing intermediate points with one drone or multiple drones. I attempt to fill in this gap in the literature by utilizing a finite number of discrete intermediate points on arcs. The intermediate points can provide more flexibility to the driver to choose more efficient truck and drone routes saving more delivery time compared to the existing model, where the drone is operated only on customers' sites. The pilot test of a truck and a drone by UPS (Transport Topics, 2017) shows that UPS was using intermediate points to operate their drone. Most existing studies about a truck and a drone delivery do not consider the effect of time windows. As a matter of fact, time windows are naturally required in the daily

operation of parcel delivery especially for business customers (Don, 2013). Practitioners also reported that time windows could bring a negative impact on their trucks scheduling, e.g. a truck may have to visit the same area twice due to the requirements of time windows. Time windows of customers will bring extra constraints to the model. It leads to more complex problems and reduces the quality of solutions. It is necessary to face this challenge and try to find high-quality solutions. That means the incorporation of time windows into this new mode of truck and drone(s) delivery is natural and critical to practitioners and researchers. Then I decided to incorporate time windows into my research to meet the requirements of practitioners and fill the gap in the literature. I also discussed my idea with some practitioners from transportation or parcel delivery companies. They were all interested in my idea and believed that my idea should be able to operate in the real world.

Dissertation Organization

My dissertation has a three-essay format. Chapter 2 is the first essay that develops a model of traveling salesman problem with time windows and a drone (TSPTWD) and intermediate points. I will solve the model and find exact optimal solutions for small size problems and with several LP heuristics to reduce computing time. Chapter 3 is the second essay in which several heuristics and metaheuristics will be developed to solve large-sized problems of TSPTWD. Chapter 4 is the third essay, a model of traveling salesman problem with time windows and multiple drones (TSPTWmD) with intermediate points will be introduced to solve large-sized problems with several metaheuristics. In each of the above-mentioned chapters, computational experiments were implemented, and the results showed the comparison of different algorithms, the impact of factors on the delivery time savings of my new models, etc.

Chapter 5 is the general summary of all the findings of the three essays and the possible future research.

CHAPTER 2. TRAVELING SALESMAN PROBLEM WITH TIME WINDOWS AND A DRONE (TSPTWD): LAUNCHED AND RECEIVED AT INTERMEDIATE POINTS

Model Formulation

Problem Definition and Illustrations

Some studies have been developed based on the idea of incorporating drones into traditional transportation modes. Dorling et al. (2017) examined the model of a fleet of drones deliver parcels to customers based on a depot. It is similar to traditional vehicle routing problems (VRP). In this model, drones cannot reach customers farther than their flying range. A natural extension is to let a truck carry a drone to some remote area and deploy it to serve some of the last-mile deliveries. Murray and Chu (2015) initiated a traveling salesman problem modeling the combined delivery by a truck and a mounted drone. This new approach absorbs the advantages of a truck and a drone, i.e. the larger range of a truck and the faster speed and lower cost of a drone. It could become an effective way to face the challenge of fast growing last-mile deliveries.

My first essay will be developed based on the research of Murray and Chu (2015). Their model only allows the drone to be launched and received on the customers' sites. I will generalize it to some discrete intermediate points on the arcs between customers. That will provide more flexibility to drivers and to reduce the total delivery time further, e.g. *Figure 2*. Other contributions are more generalized assumptions will be added, e.g. waiting times between the truck and the drone will be incorporated, all nodes can be used multiple times, out of route distances will be considered which exist in the real world, etc. Some decision variables will be removed from the current model to simplify the problem.

To relate my model more closely to the real world, some points may not allow the operation of launching or receiving a drone due to the restriction of some special areas, e.g. airports, gas stations, high-density residential areas, etc. If some areas cannot be used, then some nodes may have to be used more than once to receive the drone. That will also be incorporated into my model. In the daily operation, the driver may have to drive a short distance out of the main road to stop and operate the drone. Those out-of-route times will also be considered in my model because to operate the drone on intermediate points the truck may stop more times than the model only operates on customers' sites. The example in Figure 2 shows that the new model needs two more stops in the serving process from node 0 to node 6. Even the flexible options can help to save delivery time, the extra stopping times, e.g. the extra time to decelerate and accelerate again near the intermediate points on arcs (0,2) and (3,6), will be considered in the new model to get a better estimation of the time saving of the new model. The extra stopping time (t_{ex}) can be proved to be proportional to the average speed on the arcs. Details can be found in appendix A.

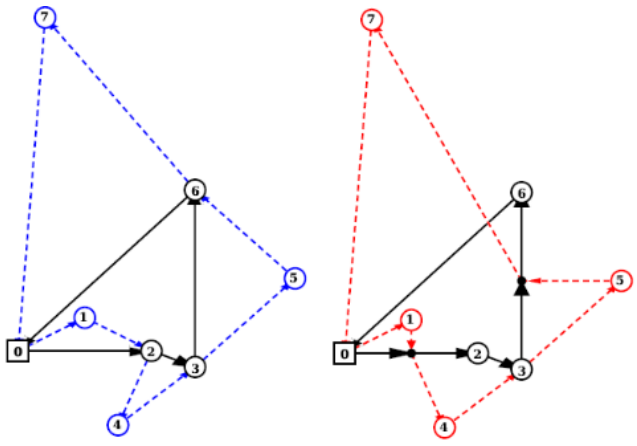


Figure 2. (left) existing model; (right) new model with intermediate points

Figure 2(left) shows the original model of Murray and Chu (2015) in which the truck driver can launch and receive the drone at customers' sites. Figure 2(right) demonstrates my new model, i.e. traveling salesman problem with time windows and with a drone (TSPTWD), which allows the drone to be launched and received at some intermediate points on arcs. Several benefits can be generated by my proposed TSPTWD. First, the total delivery time can be reduced. Second, some customers which cannot be served by the drone in the previous model (FSTSP) due to the long distance over than maximum limit of the drone might be able to be delivered by the drone by launching and receiving it at some nearer intermediate points. Third, the allowance of using the intermediate points provides more flexibility to the truck driver to meet the customers' requirements of time windows.

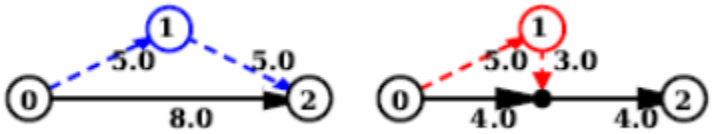


Figure 3. intermediate points help to reduce delivery time

An example is shown in Figure 3 to demonstrate the above mentioned three potential benefits by using intermediate points to launch and receive drones. The literature shows that the translational speed of commercial drones could be as fast as 45mph (Perez and Kolodny, 2017). The average speed of a parcel delivery vehicle is about 20mph (Barnitt, 2011). To simplify the calculation, we consider the drone's speed is twice the truck which means the drone uses half the time of the truck to traverse a given distance. As shown in figure 3, the solid line the route of the truck and dash lines are for the drone. The distance traveled by the truck from A to B is 8 and the traveling time interval is also 8 time units. In the previous research, the drone can only be launched and received on nodes. So the drone's flying time on the branch route $0 \rightarrow 1 \rightarrow 2$ is

(5+5)/2 = 5. That means the truck needs to wait 3 time units and the total time to serve customers 0,1,2 by the truck and drone is 8 time units. If the driver uses our proposed strategy to launch or receive the drone on any eligible intermediate points, e.g. launching on 0, delivering at 1 and receiving on the midpoint of 0 and 2. Then the traveling time from 0 to midpoint for the truck is 4 and for the drone is (5+3)/2 = 4 too. After receiving the drone, the truck continues its trip to 2 and finishes the task in 8 time units. It looks like the team of truck and drone spends the same time units (8) in the two scenarios, but time does have been saved. Note that in our proposed procedure the drone is received 4 minutes after its launch but in the previous approach it comes back to the truck is 8 minutes (flying 5 minutes plus hovering 3 minutes). In the new procedure the drone is received 4 minutes earlier (a time saving of 50%) has at least two advantages. First, the drone can be relaunched earlier than in the old way to help the truck deliver more parcels to save the total service time. Second, the drone's working time (flying and hovering) has been saved 50%, that means the energy of the drone is also saved 50% which has an obvious contribution to sustainable operation. The third advantage of the novel approach is that some ineligible customers may become eligible for drone delivery due to their long traveling distance larger than the range of the drone. In my example, if the distance of $0 \rightarrow 1 \rightarrow 2$ is longer than the traveling limit of the drone, then customer 1 was ineligible for the drone in the old model. It may become eligible in our model since the drone's traveling distance can be reduced. Forth, since time can be saved in the new model the team of the truck and drone will have more capability to meet other customers' time window requirements which may not be fulfilled with the previous model. Although we cannot state that our proposal of using intermediate points for drones can always save time and energy as high as 50%, there is evidence that our new strategy can provide more opportunities to further optimize this truck and drone delivery model.

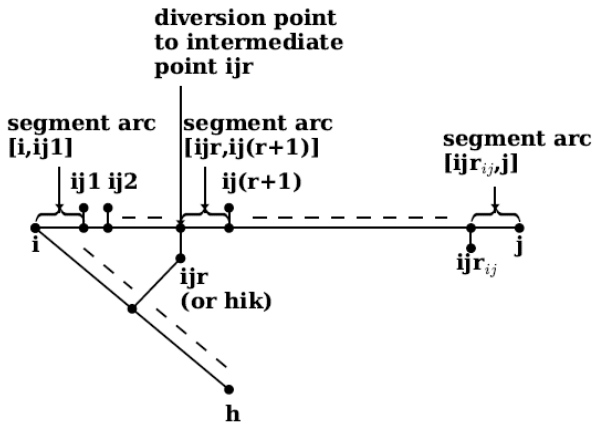


Figure 4. Intermediate points, diversion points, and segment arcs

To solve this problem with continuous intermediate points is difficult. So we discretize each arc in the following way and demonstrate it in Figure 4. Since not all the points on an arc are allowed to operate drones, e.g. on the highway, near the intersections, near an airport, etc., we choose and call those eligible points, e.g. parking lots, parks, rest areas, for launching and receiving drones on an arc as the intermediate points. To make the situation more generalized, we consider those intermediate points are not always exactly on the arc because in the real world they maybe a little bit away from the arc or out of the route. The point from where an intermediate point diverges from the main arc is called the diversion point. We denote the r^{th} intermediate point on arc ij as ijr, the out of route distance of ijr is the distance between ijr and its corresponding diversion point which is denoted as e_{ijr}. When a truck traverses on the arc ij and visits the intermediate point ijr to either launch and or receive the drone, its total distance out of the route is $2e_{ijr}$. A practitioner's pilot test (UPS, 2017) showed that normally the kind of distances are not very large (about 3~5 meters) when compared with the average distance between consecutive customers in a typical parcel delivery which is about one mile or even shorter (Don, 2013, Lammert and Walkowicz, 2012). My model intends to minimize the total

delivery time. Then the out of route distances will bring extra time not only by the extra distance but also by the extra stopping and restarting processes. The calculation of the extra time t_{ex} is discussed and calculated in Appendix A. The total number of intermediate points on arc ij is r_{ij} and the number of segment arcs separated by the intermediate points and nodes i, j is $r_{ij}+1$. To reduce the number of decision variables we keep a unique name for each customer or node (e.g. i, j, ... etc.) rather than assign multiple names ij0, ik0, il0… to one node i. More details can be found in the explanation of constraints (2g_01) ~ (2g_04). Note that some intermediate points may belong to multiple arcs. For example, ijr is the r^{th} intermediate point of arc ij and it is also the k^{th} intermediate point of arc hi, so it has another name as hik with an out of route distance e_{hik}. In my model, I define segment arcs between two neighboring points and to the incremental direction. For example, [ijr, ij(r+1)] describes the segment of arc from ijr to ij(r+1). Two special cases are the beginning and ending ones, i.e. [i, ij1] and [ijr_{ij}, j] which denotes the segment of arc from node i to the first intermediate point ij1and from the intermediate point ijr_{ij} to node j. Arcs from an intermediate point to another customer are allowed in branch routes of the drone and will be discussed later.

Proposed Model TSPTWD

To define the TSPTWD model we denote the set of customers as V = {1,2,…,n}. The depot is noted as 0, n+1, and acting as the starting and ending points of the route. The set of all the intermediate points is R. V' is the set of visits to the customers including V and dummy points of each element of V to permit multiple visits to customers. Similarly, R' is the set including R and corresponding dummy points of elements of R to allow multiple visits to the intermediate points. Time windows for all customers and the depot are $[e_i, l_i]$, $i \in V_{0,n+1}$ = {0,1,2,….,n, n+1} and $[e_0, l_0] = [e_{n+1}, l_{n+1}] = [E, L]$. Let t_{ijr} be the time instant that the truck

arrives at a point ijr $\in V^*_{0,n+1} = V'_{0,n+1} \cup R'$, and $\tau_{[ijr,\ ij(r+1)]}$ the time interval of the truck traversing the segment arc [ijr, ij(r+1)] (ijr $\in V^*_0 = V'_0 \cup R'$). Then t_{ijr}' is the time instant that the drone arrives at the intermediate point ijr$\in V^*_{0,n+1}$, and $\tau'_{[ijr1,\ k],}\ \tau'_{[k,\ lmr2]}$ the time interval of the drone traversing the branch arcs [ijr_1, k] and [k, lmr_2] ($ijr_1 \in V^*_0$, [ijr_1, k, lmr_2] $\in$B). B is the set of all the possible branch routes [ijr_1, k, lmr_2] of the drone traveling to all the customers k in the set of the eligible customers for the drone to deliver $V_d \subseteq V$. We let $t_{0j0} = t_{0j0}' = 0$, $j \in V_{n+1}$. Then, TSPTWD can be defined on a directed graph G = ($V^*_{0,n+1}, A$), whereas the set of arcs $A = \{(i,j) | i \in V'_0, j \in V'_{n+1}, i \neq j\}$

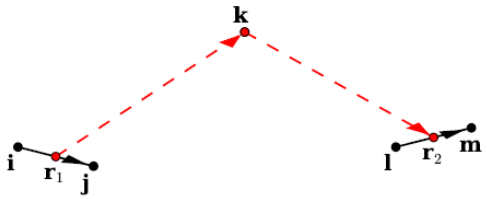


Figure 5. A drone utilizes two intermediate points to visit a customer

In my model, the drone can be launched from all the customers' sites, intermediate points, and the starting point depot, i.e. the set $V^*_0 = V'_0 \cup R'$. It can be received at all the customers' sites, intermediate points, and the ending point depot, i.e. the set $V^*_{n+1} = V'_{n+1} \cup R'$. A branch route [$ijr_1$, k, lmr_2] denotes the route traversed only by the drone from point ijr_1 to customer k and then back to lmr_2 as shown in Figure 5. The drone also can be launched or received at the depot. Any point (except the depot) can be used multiple times, by assigning dummy points to all customers and intermediate points, to launch or receive the drone since in

the real world some points might be suitable for multiple utilization or their neighbors are not allowed to operate the drone. Not all customers' parcels can be delivered by the drone due to the maximum loading weight, limited flying duration of the drone or a drone restricted area near an airport, power plants, high-density residential areas, etc. The set of those customers which are eligible to be served by the drone is $V_d \subseteq V$. The maximum flying duration of the drone is denoted as F. All the possible branch routes [ijr_1, k, lmr_2] visiting all the customers in V_d form the set B. Note that $ijr_1 \neq k \neq lmr_2$, but ijr_1 and lmr_2 could be on the same arc. Note that $ijr_1 \neq lmr_2$ means that within one branch route the drone is not allowed to be launched and received at the same (mathematical) point in the model but the drone can come back to the same physical points multiple times by using the dummy points. The driver needs some time to launch and receive the drone which are denoted as s_L and s_R. In most cases, the drone and the truck cannot arrive at the receiving point at the same time. So two decision variables w_{ijr} and w'_{ijr} are used to indicate the waiting time interval of the truck or the drone for their counterparts at the point $ijr \in V^*_{n+1}$. Another auxiliary binary decision variable $p_{[der1,\ ijr2]}$ (der_1, $ijr_3 \in V' \cup R'$, $ijr_3 \neq der_1$) is introduced. It equals 1 when the truck visits the point der_1 earlier than ijr_3, otherwise 0. This is used to prevent crossing branch routes as shown in Figure 6. That means the launching time instant of the drone on ijr_3 cannot be between the time instants of a pair previous launching and receiving points (i.e. der_1, ghr_2).

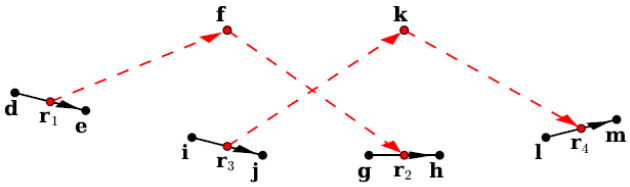


Figure 6. Overlapping branch routes of a drone should be prohibited

Notations of parameters and decision variables are described in Table 1.

Table 1. Sets, parameters, and decision variables of the TSPTWD model

Sets, Parameters:	
0, n+1	Depot
V	Set of all customers {1,2,…..,n}
V_d	Set of eligible customers that can be served by the drone. $V_d \subseteq V$
$V_{0,n+1}$	$V_{0,n+1} = V \cup \{0, n+1\} = \{0,1, \ldots\ldots, n, n+1\}$
V_0	$V_0 = V \cup \{0\} = \{0,1, \ldots\ldots, n\}$
V_{n+1}	$V_{n+1} = V \cup \{n+1\} = \{1, \ldots\ldots, n, n+1\}$
V'	Set of visits to the customers, including V and dummy points of each element of V to permit multiple visits to them.
$V'_0, V'_{n+1}, V'_{0,n+1}$	$V'_0 = V' \cup \{0\}, V'_{n+1} = V' \cup \{n+1\}, V'_{0,n+1} = V' \cup \{0, n+1\}$
s_{ij}	Number of segments on the arc ij within which there are s_{ij}-1 intermediate points (Figure 3). $i \in V_0$, $j \in V_{n+1}$
R_{ij}	Set of all intermediate points on the arc ij. R_{ij} = {ij1, ij2,…, ijr, …, ij(s_{ij}-1)}, $i \in V_0$, $j \in V_{n+1}$
R	Set of all the intermediate points

Table 1 Continued

Table 1. Sets, parameters, and decision variables of the TSPTWD model

R'	Set of visits to all the intermediate points, including R and dummy points of each element of R to permit multiple visits to them.
$R'_0, R'_{0,n+1}$	$R'_0 = R' \cup \{0\}, R'_{0,n+1} = R' \cup \{0, n+1\}$
$V^*, V_0^*, V_{n+1}^*, V_{0,n+1}^*$	$V^* = V' \cup R', V_0^* = V_0' \cup R', V_{n+1}^* = V_{n+1}' \cup R', V_{0,n+1}^* = V_{0,n+1}' \cup R'$
[ijr$_1$, k, lmr$_2$]	a branch route of the drone departing from the truck at the point of ijr$_1$ to visit a customer k then meet the truck at the point lmr$_2$. $ijr_1 \in V_0^*, lmr_2 \in V_{n+1}^*, ijr_1 \neq k \neq lmr_2, k \in V_d$.
B	Set of all the possible branch routes [ijr$_1$, k, lmr$_2$] of the drone.
$\tau_{[ijr,\ ij(r+1)]}$	Time interval of the truck traversing from the diversion point of ijr to the diversion point of ij(r+1). ijr $\in V_0^*$.
t_{ex}	The extra stopping time of the truck traversing out-of-route to decelerate and then accelerate again at an intermediate point. Details can be found in Appendix A.
$\tau'_{[ijr1,\ k]}, \tau'_{[k,\ lmr2]}$	Time interval of the drone traversing from an intermediate point ijr$_1$ to customer k and from k to lmr$_2$. ijr$_1$ $\in V_0^*$, [ijr$_1$, k, lmr$_2$] $\in$B
s_L	Preparing time interval by the driver for launching the drone.
s_R	Receiving and recovering time interval of the drone which may include the driver changing batteries etc.
F	The maximum flying time of the drone.
[e_i, l_i]	Time windows for all customers and the depot. i$\in V_{0,n+1}$.
[E, L]	The time window of the depot, [e_0, l_0] = [e_{n+1}, l_{n+1}] = [E, L].
δ	$\delta > 0$, a small enough positive number.
M	M > 0, a big enough positive number.

Table 1 Continued

Table 1. Sets, parameters, and decision variables of the TSPTWD model

Decision variables:	
x_{ij}	Binary decision variable indicating whether the arc ij is traversed by the truck. $\forall i \in V_0', j \in V_{n+1}', i \neq j$.
$y_{[ijr_1,k,lmr_2]}$	Binary decision variable indicating whether the branch route $[ijr_1, k, lmr_2]$ is traversed by the drone. $ijr_1 \in V_0^*$, $lmr_2 \in V_{n+1}^*$, $ijr_1 \neq k \neq lmr_2$, $k \in V_d$.
z_{ijr}	Binary decision variable indicating whether the intermediate point ijr is visited by the truck either to launch or receive the drone or do both. $ijr \in V_{0,n+1}^*$.
$p_{[der_1,ijr_3]}$	Binary decision variable equals to 1 when the truck visits the point der_1 earlier than ijr_3, and 0 otherwise. $der_1, ijr_3 \in V^*$, $ijr_3 \neq der_1$. Referring to Figure 5. $p_{[0,ijr]} = 1 \; \forall ijr \in V^*$.
$t_{ijr} \geq 0$	Arrival time instant of the truck at a point $ijr \in V_{0,n+1}^*$. ($t_0 = 0$)
$t'_{ijr} \geq 0$	Arrival time instant of the drone at a point $ijr \in V_{0,n+1}^*$. ($t'_0 = 0$). They will be removed in the simplified model TSPTWD2.
$w_{ijr} \geq 0$	Waiting time interval of the truck (for the drone) at point $ijr \in V_{n+1}^*$. They will be removed in the simplified model TSPTWD2.
$w'_{ijr} \geq 0$	Waiting time interval of the drone (for the truck) at point $ijr \in V_{n+1}^*$. They will be removed in the simplified model TSPTWD2.

The objective of this original model TSPTWD1 is to minimize the total delivery time.

$$(TSPTWD1) \;\; Min \; t_{n+1} \tag{1}$$

I make sure the routes of the truck and drone are valid.

$$s.t. \sum_{h\epsilon V_0', h\neq k} x_{hk} + \sum_{ijr_1\epsilon V_0^*} \sum_{lmr_2\epsilon V_{n+1}^*} y_{[ijr_1,k,lmr_2]} = 1 \quad \forall k \epsilon V \tag{2a}$$

$$\sum_{j\epsilon V_{n+1}'} x_{oj} = 1 \tag{2b}$$

$$\sum_{i\epsilon V_0'} x_{i(n+1)} = 1 \tag{2c}$$

$$\sum_{i\epsilon V_0', i\neq j} x_{ij} = \sum_{i\epsilon V_{n+1}', i\neq j} x_{ji} \quad \forall j \epsilon V' \tag{2d}$$

$$\sum_{k\epsilon V_d} \sum_{lmr_2\epsilon V_{n+1}^*} y_{[ijr_1,k,lmr_2]} \leq 1 \quad \forall ijr_1 \epsilon V_0^* \tag{2e}$$

$$\sum_{ijr_1\epsilon V_0^*} \sum_{k\epsilon V_d} y_{[ijr_1,k,lmr_2]} \leq 1 \quad \forall lmr_2 \epsilon V_{n+1}^* \tag{2f}$$

$$2y_{[ijr_1,k,lmr_2]} \leq x_{ij} + x_{lm} \qquad \forall [ijr_1, k, lmr_2] \epsilon \{B | \forall ijr_1, lmr_2 \epsilon R'\} \tag{2g\_01}$$

$$2y_{[i,k,lmr_2]} \leq \sum_{j\epsilon V_{n+1}'\setminus\{i,k\}} x_{ij} + x_{lm} \qquad \forall [i, k, lmr_2] \epsilon \{B | \forall i \epsilon V_0', lmr_2 \epsilon R'\} \tag{2g\_02}$$

$$2y_{[ijr_1,k,m]} \leq x_{ij} + \sum_{l\epsilon V_0'\setminus\{m,k\}} x_{lm} \qquad \forall [ijr_1, k, m] \epsilon \{B | \forall ijr_1 \epsilon R', m \epsilon V_{n+1}'\} \tag{2g\_03}$$

$$2y_{[i,k,m]} \leq \sum_{j\epsilon V_{n+1}'\setminus\{i,k\}} x_{ij} + \sum_{l\epsilon V_0'\setminus\{m,k\}} x_{lm} \qquad \forall [i, k, m] \epsilon \{B | \forall i \epsilon V_0', m \epsilon V_{n+1}'\} \tag{2g\_04}$$

Constraint (2a) states that each customer must be visited once by either the truck or the drone. Note that there is not a one-time constraint for the truck to all the dummy points since due to the time tracking the truck is not allowed to visit a (mathematical) point more than one time. But they can visit the same physical points multiple times by using the corresponding dummy points. Constraints for the drone visiting the dummy points are not written in the model either because the minimization of returning time will naturally prevent the drone from visiting the unrequired dummy points to 'waste' time. Equations (2b), (2c) require the truck to leave and

back to the depot only one time respectively. The truck's flow in equals the flow out of a customer is restricted by equation (2d). Constraints (2e), (2f) limit that the drone cannot be launched or received more than one time at any point. The set of inequalities (2g) is set to assure that the truck must traverse an arc if the drone is launched or received at any point on the arc. Multiple launches and receptions on the same arc or the same point are allowed by using the dummy points. As a matter of fact, (2g) can be described concisely as

$$2y_{[ijr_1,k,lmr_2]} \leq x_{ij} + x_{lm} \quad \forall [ijr_1, k, lmr_2] \, \epsilon \, B$$

But this formulation will force each customer to be named multiple times, e.g. customer 1 has to be named as [1,2,0] (the beginning intermediate point on arc (1,2)), [1,3,0], [1,4,0] etc. That will increase the complexity of the model significantly due to the increased number of binary decision variables such as $y_{[ijr_1,k,lmr_2]}$ etc. So, I choose to separate (2g) into four groups of constraints to maintain a unique name for each customer to reduce the number of decision variables, but the total number of constraints for (2g) does not change. The four constraints represent four scenarios respectively, i.e. both starting and ending points are intermediate points (2g_01), one of them is an intermediate point the other is a customer (2g_02, 2g_03) and both of them are customers (2g_04).

I set the time tracking of the truck.

$$\sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y_{[ijr_1,k,lmr_2]} + \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y_{[lmr_2,n,pqr_3]} \leq 2z_{\,lmr_2} \leq 2 \sum_{k \epsilon V_d} \sum_{ijr_1 \epsilon V_0^*} y_{[ijr_1,k,lmr_2]} +$$

$$2 \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y_{[lmr_2,n,pqr_3]} \qquad \forall lmr_2 \, \epsilon \, V_{0,n+1}^* \qquad (3a)$$

$$t_{lmr_2} \geq t_{lm(r_2-1)} + s_L \left(\sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y_{[lm(r_2-1),n,pqr_3]} \right) + \frac{1}{2} t_{ex} z_{\,lm(r_2-1)} + \tau_{[lm(r_2-1),lmr_2]}$$

$$+ \frac{1}{2} t_{ex} z_{\,lmr_2} + w_{lmr_2} + s_R \left(\sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y_{[ijr_1,k,lmr_2]} \right) - M_{1[lm(r_2-1),lmr_2]} \cdot (1 - x_{lm})$$

$$\forall lmr_2 \epsilon V_{n+1}^*, lm(r_2-1)\epsilon V_0^* \quad (3b)$$

$$e_k\left(1-\sum_{ijr_1\epsilon V_0^*}\sum_{lmr_2\epsilon V_{n+1}^*} y_{[ijr_1,k,lmr_2]}\right) \le t_k \le l_k\left(1-\sum_{ijr_1\epsilon V_0^*}\sum_{lmr_2\epsilon V_{n+1}^*} y_{[ijr_1,k,lmr_2]}\right) \forall k\epsilon V_{0,n+1} (3c)$$

Constraints (3a) determines $z_{lmr_2} = 1$ when an intermediate point is visited by the truck either to launch or receive the drone or to do both. Arrival time instants of the truck are defined by constraint (3b) within which the waiting time for the drone, the receiving and launching times, and the time of out-of-route are considered. The M_1 is a big enough positive number. To make the constraints more tightly, we can set $M_{1[lm(r_2-1),lmr_2]} = max(l_l, l_m) + \tau_{[lm(r_2-1),lmr_2]} - min(e_l, e_m) + \frac{1}{2}t_{ex} + \frac{1}{2}t_{ex} + \left(F - \tau_{[lm(r_2-1),lmr_2]}\right) + s_R + s_L = max(l_l, l_m) - min(e_l, e_m) + t_{ex} + F + s_R + s_L$. Time window constraints of customers visited by the truck are described by constraint (3c).

I set the time tracking of the drone.

$$t'_{ijr_1} \ge t_{ijr_1} - L\left(1-\sum_{k\epsilon V_d}\sum_{lmr_2\epsilon V_{n+1}^*} y_{[ijr_1,k,lmr_2]}\right) \quad \forall ijr_1 \epsilon V_0^* \quad (4a)$$

$$t'_{ijr_1} \le t_{ijr_1} + L\left(1-\sum_{k\epsilon V_d}\sum_{lmr_2\epsilon V_{n+1}^*} y_{[ijr_1,k,lmr_2]}\right) \quad \forall ijr_1 \epsilon V_0^* \quad (4b)$$

$$t'_k \ge t'_{ijr_1} + \left(s_L + \tau'_{[ijr_1,k]} + L\right)\left(\sum_{lmr_2\epsilon V_{n+1}^*} y_{[ijr_1,k,lmr_2]}\right) - L \quad \forall k \epsilon V_d, ijr_1\epsilon V_0^* \quad (4c)$$

$$t'_{lmr_2} \ge t'_k + \tau'_{[k,lmr_2]} + w'_{lmr_2} + s_R + M_{2[k,lmr_2]}\left(\sum_{\forall ijr_1 \epsilon V_0^*} y_{[ijr_1,k,lmr_2]} - 1\right) \forall k\epsilon V_d, lmr_2\epsilon V_{n+1}^* (4d)$$

$$t'_{lmr_2} \ge t_{lmr_2} - L\left(1-\sum_{ijr_1\epsilon V_0^*}\sum_{k\epsilon V_d} y_{[ijr_1,k,lmr_2]}\right) \quad \forall lmr_2 \epsilon V_{n+1}^* \quad (4e)$$

$$t'_{lmr_2} \le t_{lmr_2} + L\left(1 - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y_{[ijr_1,k,lmr_2]}\right) \quad \forall lmr_2 \epsilon V_{n+1}^* \qquad (4f)$$

$$e_k \sum_{ijr_1 \epsilon V_0^*} \sum_{lmr_2 \epsilon V_{n+1}^*} y_{[ijr_1,k,lmr_2]} \le t'_k \le L + (l_k - L) \sum_{ijr_1 \epsilon V_0^*} \sum_{lmr_2 \epsilon V_{n+1}^*} y_{[ijr_1,k,lmr_2]} \quad \forall k \epsilon V \qquad (4g)$$

$$t'_{lmr_2} - t'_{ijr_1} \le F + (L - F)\left(1 - y_{[ijr_1,k,lmr_2]}\right) \quad \forall [ijr_1, k, lmr_2] \epsilon B \qquad (4h)$$

Constraints (4a), (4b) and (4e), (4f) guarantee the time consistency between the drone and the truck at the launching and receiving points. Constraints (4c), (4d) traces the arrival time instants on the branch routes of the drone. $M_{2[k,lmr\_2\ ]}$ is a positive big enough number. To make the constraints more tightly, we can set $M_{2[k,lmr_2]} = l_k - min(e_l, e_m) + \tau'_{[k,lmr_2]} + F - \min_{ijr_1 \epsilon V_0^*} \{\tau'_{[ijr_1,k]} + \tau'_{[k,lmr_2]} \mid k \epsilon V_d, lmr_2 \epsilon V_{n+1}^*, [ijr_1, k, lmr_2] \epsilon B\} + s_R + \frac{1}{2} t_{ex} = l_k - min(e_l, e_m) + F - \min_{ijr_1 \epsilon V_0^*} \{\tau'_{[ijr_1,k]} \mid k \epsilon V_d, [ijr_1, k, lmr_2] \epsilon B\} + s_R + \frac{1}{2} t_{ex}$. Customers' time windows and flying time limit for the drone are restricted by constraints (4g), (4h) respectively.

Crossed branch routes as shown in Figure 6 are prohibited by constraints (5a)~(5c).

$$t'_{ijr_3} - t'_{der_1} \le M_{3a[der_1,ijr_3]} \cdot p_{[der_1,ijr_3]} \quad \forall der_1 \epsilon V_0^*, ijr_3 \epsilon \{V^* | ijr_3 \ne der_1\} \qquad (5a)$$

$$t'_{der_1} - t'_{ijr_3} \le M_{3b[der_1,ijr_3]} \cdot (1 - p_{[der_1,ijr_3]}) \quad \forall der_1 \epsilon V_0^*, ijr_3 \epsilon \{V^* | ijr_3 \ne der_1\} \qquad (5b)$$

$$t'_{ijr_3} \ge t'_{ghr_2} - L(3 - \sum_{f \epsilon V_d} y_{[der_1,f,ghr_2]} - \sum_{k \epsilon V_d} \sum_{lmr_4 \epsilon V_{n+1}^*} y_{[ijr_3,k,lmr_4]} - p_{[der_1,ijr_3]})$$

$$\forall der_1 \epsilon V_0^*, ghr_2 \epsilon V_{n+1}^*, ijr_3 \epsilon \{V^* | ijr_3 \ne der_1\} \qquad (5c)$$

$M_{3a[der_1,ijr_3]}$ and $M_{3b[der_1,ijr_3]}$ are two positive big enough numbers. To make the constraints more tightly, we can set them equal to L - E. If we want to make them even tighter, their values can be $M_{3a[der_1,ijr_3]} = \max(l_i, l_j) - \min(e_d, e_e)$, $M_{3b[der_1,ijr_3]} = \max(l_d, l_e) - \min(e_i, e_j)$.

Decision variables are defined in (6a)~(6f).

$$t_0 = t'_0 = 0 \tag{6a}$$

$$t_{ijr}, t'_{ijr} \geq 0 \quad \forall ijr \in V^*_{0,n+1} \tag{6b}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in V'_0, j \in V'_{n+1}, i \neq j \tag{6c}$$

$$y_{[ijr_1,k,lmr_2]} \in \{0,1\} \quad \forall ijr_1 \in V^*_0, k \in V_d, lmr_2 \in V^*_{n+1}, ijr_1 \neq k \neq lmr_2 \tag{6d}$$

$$F \geq w_{ijr}, w'_{ijr} \geq 0 \quad \forall ijr \in V^*_{n+1} \tag{6e}$$

$$p_{[der_1,ijr_3]} \in \{0,1\} \quad \forall der_1 \in V^*_0, ijr_3 \in \{V^* | ijr_3 \neq der_1\} \tag{6f}$$

$$z_{ijr} \in \{0,1\} \quad ijr \in V^*_{0,n+1} \tag{6g}$$

There are several actions to reduce the complexity of this complicated model, e.g. reducing the number of decision variables.

First, I just define binary decision variables x_{ij} for the arcs of the truck rather than define $x_{[ijr-1,ijr]}$ for each segment on arcs. That has helped to reduce a lot of binary decision variables and constraints.

Second, there is a problem in this model that w_{lmr2}, w'_{lmr2} are unidentified (not uniquely determined) in 3b, 4d, e.g. they can be any value between 0 and their real values. Experiments also showed these results. I may deal with it by introduce $-\delta \sum_{k \in V'} w_k - \delta \sum_{k \in V'} w'_k$ into the objective function to maximize w_{lmr2}, w'_{lmr2}, whereas δ is a small enough positive number. Even it can solve the problem, it is not a good practice for optimization to introduce unrelated DVs to the objective function and we do not implement it even δ is very small, e.g. δ = 1e-06. After observation, I found that decision variables w_{lmr2}, w'_{lmr2}, t'_k can be removed from the model and calculated after the optimization. This is discussed in the following section.

I noticed that each branch route has a fixed distance and traveling time. When a branch route of the drone is chosen in the model, i.e. $y_{[ijr_1,k,lmr_2]} = 1$, the beginning and ending nodes' arrival time instants are set equal to those of the truck. Then the drone's visiting time instant to a customer and the waiting times of both the truck and drone can be calculated after the optimization. With that, I can remove some decision variables, i.e. w_{lmr2}, w'_{lmr2}, t'_k and some constraints. The new simplified model is shown as the following.

Simplified model TSPTWD2

Since the drone's arrival times at launching and receiving nodes are the same as those of the truck, then we will determine them after the optimization process as well as the waiting times of the truck and the drone. Table 1 also can be used to describe all the parameters and decision variables of the simplified model TSPTWD2 except the decision variables t'_{ijr}, w_{ijr}, and w'_{ijr} should be removed.

(7) is the same as (1).

$$(TSPTWD2)\ Min \quad t_{n+1} \tag{7}$$

Constraints (8a~8g) are the same as (2a~2g).

$$s.t. \sum_{h\epsilon V_0', h\neq k} x_{hk} + \sum_{ijr_1\epsilon V_0^*} \sum_{lmr_2\epsilon V_{n+1}^*} y_{[ijr_1,k,lmr_2]} = 1 \quad \forall k\ \epsilon\ V \tag{8a}$$

$$\sum_{j\epsilon V_{n+1}'} x_{oj} = 1 \tag{8b}$$

$$\sum_{i\epsilon V_0'} x_{i(n+1)} = 1 \tag{8c}$$

$$\sum_{i\epsilon V_0', i\neq j} x_{ij} = \sum_{i\epsilon V_{n+1}', i\neq j} x_{ji} \quad \forall j \epsilon V' \tag{8d}$$

$$\sum_{k\epsilon V_d} \sum_{lmr_2\epsilon V_{n+1}^*} y_{[ijr_1,k,lmr_2]} \leq 1 \quad \forall ijr_1 \epsilon V_0^* \tag{8e}$$

$$\sum_{ijr_1\epsilon V_0^*} \sum_{k\epsilon V_d} y_{[ijr_1,k,lmr_2]} \leq 1 \quad \forall lmr_2 \epsilon V_{n+1}^* \tag{8f}$$

$$2y_{[ijr_1,k,lmr_2]} \leq x_{ij} + x_{lm} \qquad \forall [ijr_1, k, lmr_2] \epsilon \{B | \forall ijr_1, lmr_2 \epsilon R'\} \tag{8g_01}$$

$$2y_{[i,k,lmr_2]} \leq \sum_{j\epsilon V_{n+1}' \setminus \{i,k\}} x_{ij} + x_{lm} \qquad \forall [i, k, lmr_2] \epsilon \{B | \forall i \epsilon V_0', lmr_2 \epsilon R'\} \tag{8g_02}$$

$$2y_{[ijr_1,k,m]} \leq x_{ij} + \sum_{l\epsilon V_0' \setminus \{m,k\}} x_{lm} \qquad \forall [ijr_1, k, m] \epsilon \{B | \forall ijr_1 \epsilon R', m \epsilon V_{n+1}'\} \tag{8g_03}$$

$$2y_{[i,k,m]} \leq \sum_{j\epsilon V_{n+1}' \setminus \{i,k\}} x_{ij} + \sum_{l\epsilon V_0' \setminus \{m,k\}} x_{lm} \qquad \forall [i, k, m] \epsilon \{B | \forall i \epsilon V_0', m \epsilon V_{n+1}'\} \tag{8g_04}$$

(9a), (9c) are the same as (3a), (3c). (9b) is almost the same as (3b) except the truck's waiting time w_{lmr_2} is removed.

$$\sum_{ijr_1\epsilon V_0^*} \sum_{k\epsilon V_d} y_{[ijr_1,k,lmr_2]} + \sum_{n\epsilon V_d} \sum_{pqr_3\epsilon V_{n+1}^*} y_{[lmr_2,n,pqr_3]} \leq 2z_{lmr_2} \leq 2 \sum_{k\epsilon V_d} \sum_{ijr_1\epsilon V_0^*} y_{[ijr_1,k,lmr_2]} +$$

$$2 \sum_{n\epsilon V_d} \sum_{pqr_3\epsilon V_{n+1}^*} y_{[lmr_2,n,pqr_3]} \qquad \forall lmr_2 \epsilon V_{0,n+1}^* \tag{9a}$$

$$t_{lmr_2} \geq t_{lm(r_2-1)} + s_L \left(\sum_{n\epsilon V_d} \sum_{pqr_3\epsilon V_{n+1}^*} y_{[lm(r_2-1),n,pqr_3]} \right) + \frac{1}{2} t_{ex} z_{lm(r_2-1)} + \tau_{[lm(r_2-1),lmr_2]}$$

$$+ \frac{1}{2} t_{ex} z_{lmr_2} + s_R \left(\sum_{ijr_1\epsilon V_0^*} \sum_{k\epsilon V_d} y_{[ijr_1,k,lmr_2]} \right) - M_{1[lm(r_2-1),lmr_2]} \cdot (1 - x_{lm})$$

$$\forall lmr_2 \epsilon V_{n+1}^*, lm(r_2-1) \epsilon V_0^* \tag{9b}$$

$$e_k \left(1 - \sum_{ijr_1\epsilon V_0^*} \sum_{lmr_2\epsilon V_{n+1}^*} y_{[ijr_1,k,lmr_2]} \right) \leq t_k \leq l_k \left(1 - \sum_{ijr_1\epsilon V_0^*} \sum_{lmr_2\epsilon V_{n+1}^*} y_{[ijr_1,k,lmr_2]} \right) \forall k \epsilon V_{0,n+1} \tag{9c}$$

(4a), (4b), (4c), (4e), (4f) are removed. (10a) sets constraints on the truck's arrival times at the launching and receiving nodes of a drone's branch route. $M_{2[ijr_1,k,lmr_2]}$ is a big enough positive number. To make the constraints more tightly, we can set $M_{2[ijr_1,k,lmr_2]} = max(l_i, l_j) - min(e_l, e_m) + \tau'_{[ijr_1,k]} + \tau'_{[k,lmr_2]} + s_R + s_L + t_{ex}$. Constraints in (10b) enforce the time windows of the customers visited by the drone. (10c) is similar to (4h) within which t' are replaced by t.

$$t_{lmr_2} \geq t_{ijr_1} + s_L + \tau'_{[ijr_1,k]} + \tau'_{[k,lmr_2]} + s_R + M_{2[ijr_1,k,lmr_2]} \cdot \left(y_{[ijr_1,k,lmr_2]} - 1\right) \quad \forall [ijr_1, k, lmr_2] \epsilon B \qquad (10a)$$

$$e_k \sum_{lmr_2 \,\epsilon\, V^*_{n+1}} y_{[ijr_1,k,lmr_2]} \leq t_{ijr_1} + (s_L + \tau'_{[ijr_1,k]}) \sum_{lmr_2 \,\epsilon\, V^*_{n+1}} y_{[ijr_1,k,lmr_2]}$$

$$\leq (l_k - L) \sum_{lmr_2 \,\epsilon\, V^*_{n+1}} y_{[ijr_1,k,lmr_2]} + L \quad \forall ijr_1 \epsilon \{V^*_0 | [ijr_1, k, lmr_2] \epsilon B\}, k \epsilon V_d \qquad (10b)$$

$$t_{lmr_2} - t_{ijr_1} \leq F + (L - F)\left(1 - y_{[ijr_1,k,lmr_2]}\right) \quad \forall [ijr_1, k, lmr_2] \epsilon B \qquad (10c)$$

Constraints (11a), (11b), (11c) are similar to (5a), (5b), (5c) whose t' are replaced by t.

$$t_{ijr_3} - t_{der_1} \leq M_{3a[der_1,ijr_3]} \cdot p_{[der_1,ijr_3]} \quad \forall der_1 \epsilon V^*_0, ijr_3 \epsilon \{V^* | ijr_3 \neq der_1\} \qquad (11a)$$

$$t_{der_1} - t_{ijr_3} \leq M_{3b[der_1,ijr_3]} \cdot (1 - p_{[der_1,ijr_3]}) \quad \forall der_1 \epsilon V^*_0, ijr_3 \epsilon \{V^* | ijr_3 \neq der_1\} \qquad (11b)$$

$$t_{ijr_3} \geq t_{ghr_2} - L(3 - \sum_{f \epsilon V_d} y_{[der_1,f,ghr_2]} - \sum_{k \epsilon V_d} \sum_{lmr_4 \epsilon V^*_{n+1}} y_{[ijr_3,k,lmr_4]} - p_{[der_1,ijr_3]})$$

$$\forall der_1 \epsilon V^*_0, ghr_2 \epsilon V^*_{n+1}, ijr_3 \epsilon \{V^* | ijr_3 \neq der_1\} \qquad (11c)$$

(6a)~(6f) are readjusted as (12a)~(12e) which removed decision variables of t', w and w'. (12d_02) is added to reduce the complexity of the model. It restricted that branch routes of the drone with a flying time longer than the maximum amount will not be generated when setting the model.

$$t_0 = 0 \qquad (12a)$$

$$t_{ijr} \geq 0 \quad \forall ijr \in V^*_{0,n+1} \qquad (12b)$$

$$x_{ij} \in \{0,1\} \quad \forall i \in V'_0, j \in V'_{n+1}, i \neq j \qquad (12c)$$

$$y_{[ijr_1,k,lmr_2]} \in \{0,1\} \quad \forall ijr_1 \in V^*_0, k \in V_d, lmr_2 \in V^*_{n+1}, ijr_1 \neq k \neq lmr_2 \qquad (12d\_01)$$

$$y_{[ijr_1,k,lmr_2]} = 0 \quad if \ \tau'_{[ijr_1,k]} + \tau'_{[k,lmr_2]} > F \qquad (12d\_02)$$

$$p_{[der_1,ijr_3]} \in \{0,1\} \quad \forall der_1 \in V^*_0, ijr_3 \in \{V^* | ijr_3 \neq der_1\} \qquad (12e)$$

$$z_{ijr} \in \{0,1\} \quad ijr \in V^*_{0,n+1} \qquad (12f)$$

Calculate t', w, w' after the optimization as the following.

$$t'_{ijr_1} = t_{ijr_1},\ t'_{lmr_2} = t_{lmr_2},\ t'_k = t_{ijr_1} + s_L + \tau'_{[ijr_1,k]} \quad \forall y_{[ijr_1,k,lmr_2]} = 1 \qquad (13a)$$

$$w_{lmr_2} = t_{lmr_2} - t_{lm(r_2-1)} - s_L * y_{[lm(r_2-1),k,lmr_2]} - \frac{1}{2} t_{ex} z_{lm(r_2-1)} - \tau_{[lm(r_2-1),lmr_2]} - \frac{1}{2} t_{ex} - s_R$$

$$\forall y_{[ijr_1,k,lmr_2]} = 1 \qquad (13b)$$

$$w'_{lmr_2} = t_{lmr_2} - t'_k - \tau'_{[k,lmr_2]} - s_R \quad \forall y_{[ijr_1,k,lmr_2]} = 1 \qquad (13c)$$

Development of LP Heuristics

In this chapter, some small-sized problems of TSPTWD will be solved to find exact solutions. Since the traveling salesman problem (TSP) is an NP-hard problem, as a generalization of TSP this new model is also an NP-hard problem. The computational time to solve larger-sized instances would increase exponentially. For example, when the number of stops increased to 8 and the maximum number of intermediate points on each arc is set as one, i.e. generate randomly zero or one intermediate point, the computing time could be as long as eight hours. If we increase the maximum number of intermediate points on each arc that will increase the computing time even longer. Furthermore, the multiple usages of nodes in the model also increase the complexity dramatically. In that case, even 4~5 customer-size problems cannot be solved within a reasonable time. So, I am going to focus only on instances with less than

seven customers to find optimal solutions. To find approximate solutions for larger size problems will be studied in the following essays of my dissertation.

Since the computing time for searching exact optimal solutions would be very long for TSPTWD, I will develop several LP(linear programming)-based heuristics to try to reduce the computational time. LP-based or LP heuristics means problems are still solved by LP solvers but will reduce computing time by introducing heuristics.

The first heuristic (heu0) is an LP rounding heuristic. All the binary decision variables will be relaxed and solved. Then the top 25% of relaxed binary decision variables that are nearer to integers will be rounded to their nearest integers (0 or 1) and never changed again. In the next iteration, the other top 25% relaxed binary decision variables will be rounded again. Until finally there are no more decision variables that can be rounded or it becomes infeasible.

The second heuristic (heu01) is a relax-and-fix heuristic (Wolsey, 1998). In my model, there are two groups of binary decision variables, i.e. x_{ij} for choosing arcs of the truck and $y_{[ijr_1,k,lmr_2]}$ for choosing branch routes of the drone. There could be two versions of heu01. In the first version heu01_01, the group of decision variables x_{ij} will be relaxed at first and then solve $y_{[ijr_1,k,lmr_2]}$ exactly. Next, $y_{[ijr_1,k,lmr_2]}$ will be fixed to the solved values and then solve x_{ij} exactly. The second version heu01_02 will reverse this process, i.e. first relax $y_{[ijr_1,k,lmr_2]}$ to solve x_{ij} exactly and then fix the values of x_{ij} to solve $y_{[ijr_1,k,lmr_2]}$ exactly. Since the optimal solution is not guaranteed, the results of the two versions will be compared to choose the better one.

The fourth one (heu02) is a hybrid of a traditional heuristic idea and the LP solver, i.e. it is inspired by the fundamental idea of Granular Tabu Search (GTS) in Toth and Vigo (2003).

Those researchers observed that in most cases only about the top 10~20% shortest arcs in a vehicle routing problem (VRP) have high probabilities to appear in the optimal or high-quality solutions. So, in their modified Tabu Search (TS), they removed most of the other ~80% 'long' arcs (except some important 'long' arcs). With a well-designed procedure, their GTS generated good quality solutions with significantly reduced computing time. According to my observation of small-sized problems, about the top 40% rather than 20% short arcs appeared in the optimal solutions since there are fewer arcs with a smaller number of customers. Then I will incorporate the idea of this heuristic into my LP solving process. That means in the LP solver, e.g. Gurobi, I will only maintain the binary decision variables 'active' (0,1 are all allowed) for the top 40% shortest (promising) arcs and those arcs directly connected to the depot. The values of all the binary decision variables of the other arcs will be set to zero, i.e. to 'prohibit' those 'long' arcs. I hope it will generate good quality solutions with reduced computing time. Drone's branch routes starting and ending on those removed 'long' arcs will also be forbidden in the LP solver too since if those arcs are not allowed to be used by the truck then the drone cannot be launched or received there.

The fifth heuristic (heu03) is developed by combining heu01 and heu02. First, I get the restricted (GTS) model from heu02 and then solve it by heu01_02. The goal of this heuristic is trying to capture the advantages of both heu02 and heu01_02, i.e. the smaller sized problem will be solved by another heuristic to further reduce the computing time. But the quality of this heuristic might be lower since it places more restrictions on models.

Computational Experiments:

I arranged the following computational experiments to test how my new model can generate further benefits to the practice of a truck and a drone delivery mode by providing intermediate points on arcs and the effectiveness of the several LP heuristics.

Experimental Factors and Parameters

A numerical experiment has been designed and implemented. The values or ranges of the factor and parameters are discussed in this section.

The number of intermediate points on each arc, may affect the delivery time saving and it was set as the factor in the following experiments. In my model, there are limited number of discrete intermediate points on each arc since in the real world some areas may not be suitable as drone operating points, e.g. gas stations, crossings, etc. As TSP is an NP-hard problem, my model as a generalized variant of TSP is also an NP-hard problem. To reduce the computing time of an LP solver I decided to restrict the number of intermediate points on each arc to no more than one since a larger number of intermediate points will increase the number of decision variables exponentially. So, I set this factor's values value between 0 and 1 within which 0 means there are no intermediate points, or the drone can only be operated on customers' sites as the existing model.

Generally, the values or ranges of parameters are based on literature or technical reports. For example, more than 80% of parcels are suitable for drone delivery, i.e. weighing less than five pounds, (Guglielmo, 2017), or percentage of customers having requirements of time windows is 4% and the random ratio of time windows vs the total delivery time is between [20%~65%] (Don, 2013). I set 10% of customers cannot be served by the drone due to their restricted locations, e.g. near airports, power plants, in high-density areas, etc. This is based on feedback from practitioners and researchers. Note that these 10% customers might overlap with those 80% drone-eligible

customers. For example, a customer is a drone-eligible customer since his/her parcel weighing less than five pounds, but we may not be able to serve him/her by the drone since the customer locates in a restricted area or falls in the set of that 10% customers. The values and ranges of all selected parameters can be found in Table 2.

The threshold percentile for GTS normally is 20% top short arcs (Toth and Vigo, 2003). But after some tests of my models, I chose to use 40% in my heu_03 because for small size problems (3~6 customers) there are not too many arcs to be chosen. To get a better solution, I had to increase that threshold value. In practice, we normally consider more than one hour is 'too long' for an LP solver to find the optimal solution. That is why I set one hour as the maximum computing limit for an LP solver.

Table 2. Factors and selected parameters of the computational experiments of LP solution

Category	Values or ranges	Data source
Experimental factors		
Number of intermediate points per arc	[0, 1]	-
Number of customers	[3, 4, 5, 6]	-
Parameters for instances		
Length of side (of a squared area)	5 miles	A
percentage of customers eligible for drone delivery due to light weights of parcels	80%	C
percentage of customers cannot be served by the drone due to their restricted locations	10%	-
percentage of customers with time windows	4%	B
Ratio of time windows vs total delivery time	[20%~65%]	B
Distances of the truck	Euclidean distance	-
Distances of the drone	Euclidean distance	-
Average speed of the truck (serving time included)	12 mph	A, B
Average speed of the drone	40 mph	E
Serving time to launch or receive a drone (s_L, s_R)	1 minute	D
Maximum flying time of a drone (F)	30 minutes	E
Extra stopping time of the truck traversing out-of-route to operate drone(s) (t_{ex})	0.1466 minute	G
Parameters for heuristics or the LP solver		
Heu_03, GTS threshold percentile	40%	-
Maximum computing time in the LP solver	1 hour	-

Data source: A = Lammert and Walkowicz (2012), Don (2013); B = Don (2013); C = Guglielmo (2017); D = Murray and Chu (2015); E = Perez and Kolodny (2017); F = Xiao et al. (2012); G = Appendix A

Experimental Design

To test the delivery time saving of my new model and the performance of those several heuristics, I set experiments of small size instances as follows. For each number of customers, 10 random instances were generated based on the parameters shown in Table 2. Then each instance will be solved eight times. They are exact optimal solutions of pure truck TSP, truck and a drone operated on customers' sites, truck and a drone operated on customers' sites and intermediate points, heu00, heu01_01, heu01_02, heu02, heu03. For this given number of customers, the average cost of ten instances was saved. Some heuristics may not find feasible solutions for some instances, then only the feasible solutions were calculated. The gaps between each heuristic solution and the optimal one were also calculated. Then the average gap of ten instances was saved as the results for each algorithm. There was a similar process for the computing time of each method.

I implemented the algorithms and experiments by using Python (3.7) to establish LP models and solve them in Gurobi (8.1.1). The program was run on a desktop PC (Core i7 2.4GHz quad-core PC with 32GB of memory, Windows system).

Results

Since TSP is an NP-hard problem, TSPTWD as a generalized problem of TSP is also an NP-hard problem. The tests in Table 3 show that a commercial solver can only solve problem size as large as six customers within a reasonable time. The results showed that for most instances heu0 generated infeasible solutions, so I will not report its results in this section.

First, average optimal delivery times and savings (based on pure truck TSP) are compared which is summarized in Table 3. It shows that the existing model, operating the drone only on customers' sites, can save delivery time 32.35% on average. My new model, operating

the drone on customers' sites and intermediate points on arcs, can save delivery time 33.98% on average and could be as large as 42.8%. Comparing with the existing model my new model can increase saving 1.48% on average. The trend of saving is increasing versus the number of customers. It means if we implement my new model with more customers there might be even larger savings. Those results are also demonstrated in Figure 7.

Second, Figure 7(a) shows that the delivery time saving versus the number of customers is non-monotonic. It could be caused by randomness since each scenario generated ten random instances and used their average as the results. There could be other reasons to explain this non-monotonic result. In the analysis of the center gravity strategy in Chapter 3, it will be shown that customer density may bring an impact on the delivery time saving rather than the number of customers. In the results part of Chapter 3, it will be shown that time windows also bring some unpredictable randomness to instances. Further experiments demonstrated that the monotonic trend would appear with respect to customer density after time windows were removed.

Table 3. Delivery time (minute) and saving percentages

#_cus	TSP (min)	TSPTWD _cus[a] (min)	saving% cus[a]	TSPTWD_ inter_p[b] (min)	saving% inter_p[b]	increased saving %
3	45.84	28.76	36.69%	28.59	37.63%	0.93%
4	53.11	36.52	29.94%	35.87	32.46%	2.52%
5	61.40	35.76	40.96%	35.12	42.80%	1.84%
6	67.52	51.94	21.82%	50.86	24.67%	2.85%
Average	56.97	38.25	32.35%	37.61	33.98%	1.48%

[a] TSPTWD_cus= delivery time of TSPTWD operating the drone only on customers' sites; saving% cus = (TSP- TSPTWD_cus)/ TSP

[b] TSPTWD_inter_p= delivery time of TSPTWD operating the drone on customers' sites and intermediate points; saving% inter_p = (TSP- TSPTWD_inter_p)/ TSP

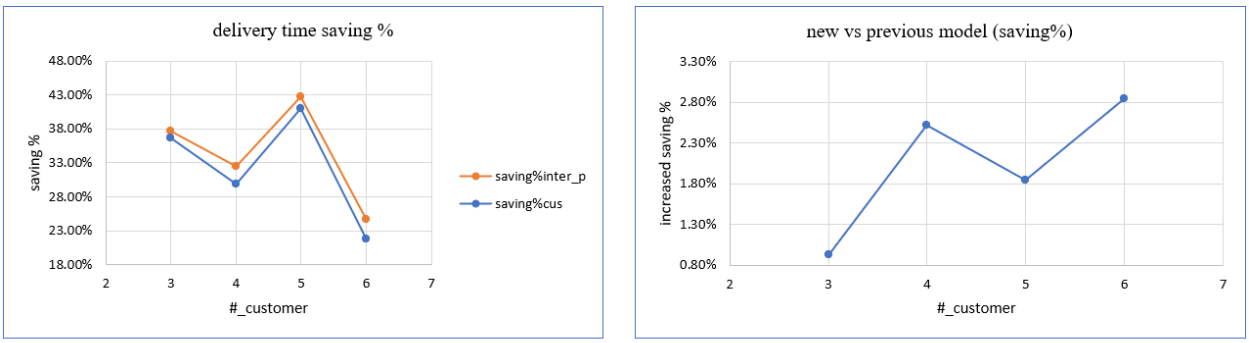


Figure 7. Delivery time and saving percentages

Third, I compared the results of several proposed LP heuristics with the exact optimal solutions. The principal is the larger the gap the worse the heuristic. Table 4 demonstrates the gaps of those LP heuristics. It is interesting that heu01_02 is much better than heu01_01. That means the sequence of relax-and-fix affects the quality of solutions. I observed and found that due to the special structure of the model, heu01_01 almost always denied the usage of the drone and generate the results as a pure traveling salesman problem without a drone. That is why heu01_01 is almost always worse than heu01_02. The smallest average gap indicates that heu02 (1.43%) is the best heuristic as I expected and heu01_01 (33.28%) is the worst.

Table 4. Gaps between heuristic results and optimal solutions

#_cus	gap01_01	gap01_02	gap_02	gap_03
3	37.01%	2.07%	1.75%	2.07%
4	30.94%	4.34%	0.00%	4.34%
5	41.92%	10.55%	3.23%	10.55%
6	23.26%	1.29%	0.26%	0.42%
Average	33.28%	4.57%	1.43%	4.78%

Forth, the computing times of optimal and heuristic methods were compared in Table 5. For the exact optimal method, the computing time increased dramatically and easily exceeded

one hour. That was why I terminated the exact computing process for problem size greater than or equal to seven customers. Within heuristics, the best one heu02 has the highest average computing time (66.67 seconds). That is the usual tradeoff we have seen in many cases. The computing time of the exact method will increase too large to run it. My experiments showed that even those heuristics will consume a very long time to get a solution.

Table 5. Computing time (second) of optimal and heuristic LP

#_customer	optimal	heu01_01	heu01_02	heu02	heu03
3	0.61	0.16	0.05	0.34	0.05
4	13.57	0.59	0.42	4.53	0.40
5	230.25	2.82	2.45	29.77	2.22
6	2,563.70	14.21	8.77	273.42	17.14
Average	702.03	4.44	2.92	66.67	4.31

Conclusion

In Chapter 2 I developed a new model to solve a novel variant of TSP, i.e. traveling salesman problem with time windows and a drone (TSPTWD). Some approaches have been implemented to reduce the complexity of this problem.

Computational experiments have been implemented to test my new model and several LP heuristics. The results of small size problems (less than seven customers) showed that my new model can save delivery time as large as 42.8% than the traditional pure truck TSP model. Comparing with the existing model of a truck and a drone operated only on customers' sites, my new model further increased the saving as large as 2.85%. And that saving could be even larger if we implement my model to larger size problems (with more customers). Also please note that to reduce the complexity of the model for the solver, we restrict that there is only one intermediate point on each arc. As a matter of fact, in the real world, there should be able to find

multiple intermediate points utilized on each arc. Then the increased flexibility would also increase savings even further.

Due to the NP-hardness, only small size problems (less than seven customers) can be solved exactly in a practical period (less than one hour). Some LP heuristics can provide relatively good solutions within a shorter time. But that computing time will also increase too large to accept. More intermediate points on each arc is also a promising approach to get more delivery time savings, but that will increase the computing complexity. That means I should develop other heuristics or metaheuristics to solve large size problems within an acceptable period. That is my goal for the next chapter.

CHAPTER 3. METAHEURISTICS TO SOLVE TRAVELING SALESMAN PROBLEM WITH TIME WINDOWS AND A DRONE (TSPTWD): LAUNCHED AND RECEIVED AT INTERMEDIATE POINTS

Development of Metaheuristics

Due to the NP-hardness of the TSPTWD in essay 1, only small-sized problems can be solved exactly in a reasonable time. The second essay will develop some metaheuristics to solve larger sized problems in essay 1. Those metaheuristics should be able to solve the problems in a shorter time with good quality solutions.

Several heuristics or metaheuristics can be considered to search for good solutions, such as savings method, tabu search (TS), simulated annealing (SA), genetic algorithm, ant colony algorithm, etc. The Clarke and Wright savings method (Clarke and Wright, 1964) is simple to implement and fast. For most vehicle routing problems it provides good solutions (Toth and Vigo, 2002). Some research shows that this savings method normally can provide a fair good solution, i.e. generally the gap between its solution and the optimal one is less than 10% (Ballou and Agarwal, 1988). To improve this result I may need to combine the original savings method with some other algorithms (Laporte, 2009). Tabu search has been shown as a promising algorithm to provide good solutions (Toth and Vigo, 2002). But the implementation of TS is also very time consuming. Some improvements have been made to mitigate it. One of them is Toth and Vigo (2003). The simulated annealing (SA) algorithm is inspired by the physical annealing process of metals to reach the lowest energy level through a random process. The computing time of this algorithm is much shorter than that of TS and the quality of solutions is also good. The convergence of the SA search to the optimal solution under some assumptions has also been proved. Genetic and ant colony algorithms are also popular in vehicle routing problems. They both can provide good quality solutions. But coding transportation problems especially the

proposed more complicated model into those algorithms might be challenging and tuning parameters is also not an easy task.

According to the above analysis, most of the existing algorithms are designed for traditional transportation problems. To solve my new model, I may need to modify some of the above-mentioned algorithms or even combine some of them to get better results. To implement any heuristics, we need to calculate the distance matrix which contains the distances between each pair of points in a problem. Before developing heuristics, I would like to discuss the problem of the huge size of the distance matrixes of TSPTWD problems and how I dealt with this challenge.

Please note that the size of distance matrixes of distances between each pair of points increases dramatically when we are going to solve a real size problem, e.g. 100 or more customers. Although the total number of arcs between customers is not very large, the fast-growing number of intermediate points on all arcs will make the size of distance matrixes very large. An approximate calculation shows that such a distance matrix requires computer memory as large as about 10G(Giga) bytes. That is too large for most computers' RAM. There are at least three possible approaches to handle this problem. First, ignore infeasible arcs in distance matrixes. For example, the distance matrix of the truck does not save distance from a customer to any intermediate points on such arcs which does not include this customer since a truck cannot travel out an existing arc. And the distance matrix of the drone will not include arcs visiting non-drone-eligible customers. Also, the drone's distance matrix can ignore all arcs between intermediate points which are the majority of the matrix. Second, drone's arcs require more than the maximum flying time will be ignored. Third, since I am going to implement heuristics, we can remove some unpromising arcs from the distance matrix to reduce the storage size. For

example, we can maintain only the shortest 20~30% arcs as mentioned by Toth and Vigo (2003). I did not use the second and third approaches due to the characteristics of my algorithm which may use all feasible arcs. Finally, I implemented the first idea and used sparse matrixes to implement it. Sizes of matrixes have been reduced to about only 1/1,000~ 1/100 of the previous unoptimized matrixes.

Modified Savings Method Traveling Salesman Problem with Time Windows (TSPTW)

To solve large size TSPTWD problems, I developed several heuristics. They are a traditional Clark and Wright Savings Method (Clarke and Wright, 1964), a naïve heuristic of TSPTWD, and two Simulated Annealing (SA) algorithms.

First, I used the Savings Method to find a good solution a pure truck traveling salesman problem (TSP). Although there are some versions of Savings Method to incorporate time windows, e.g. Solomon (1987), I chose the following modified version since it is relatively simple and can generate good solutions satisfying time windows. So I used the 2-opt operator (Figure 8) 1,000 times on the primary solution of the Savings Method tried to find higher quality solutions that satisfied the time windows.

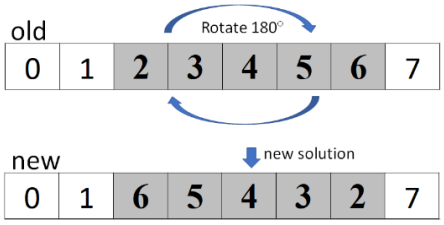


Figure 8. 2-opt operation

Naïve Method of TSPTWD

Second, I developed a heuristic to find a truck and drone solution based on a TSP solution found by the Savings Method. The main idea of this heuristic is to find drone-visit customers and launching and receiving points randomly. This process will be repeated 1,000 times try to find good solutions satisfying time windows and the limited flying duration of the drone. This heuristic is described in Algorithm 1.

Algorithm 1. naïve heuristic for TSPTWD

Input:

Solution: string sol (solution of the truck of traveling salesman problem)

Time windows of each customer and the depot: TW

Penalty weights to the violations of time windows and the maximum flying time of the drone, ω_{TW}, ω_F.

τ, τ', V_d, s_L, s_R, t_{ex}, F (symbols are defined in Table 1)

Function naïve heuristic for TSPTWD:

Step 1. choose drone-visit customers randomly from V_d and pull them out of the original truck route to get a shrunk truck route.

Step 2. insert all the possible intermediate points on all arcs of the shrunk truck route in Step 1. If there are not enough number of points in the left truck route to support all drone-visit customers chosen in Step 1, then go back to Step 1 until it is satisfied.

Step 3. Randomly choose launching and receiving points with a number = 2*(number of drone-visit customers) from the updated truck route in Step 2. Most points can be chosen no more than two times (a point can be used as both a receiving and a launching point) except the start and end depot no more than one time.

Step 4.1 The chosen drone-visit customers will be visited by the drone accordingly based on those chosen launching and receiving points. The total delivery time can be calculated and added to the cost.

Step 4.2 If any customer's serving time violates its time window, the absolute values of violation will be multiplied by ω_{TW} and added to the cost.

Step 4.3 If any flying time of the drone's branch routes exceeds the maximum flying time, the absolute values of violation will be multiplied by ω_F and added to the cost.

Step 5. Use 2-opt one time on the original truck route and go back to step 1 to find truck and drone solution and associated cost.

Step 6. Repeat step 5 1,000 times and return the solution with the lowest cost.

Return *cost_sol, truck_route, drone_branches*

End Function

Some tests showed that sometimes this heuristic can generate better solutions than those of the Savings Methods but not always. One of the reasons is that the drone-visit customers and launching/receiving points are all chosen randomly. They may not be well matched which means they might be far away from each other which could decrease the quality of solutions. Even though this heuristic is relatively simple, it only generates poor quality solutions. That is why it is called a naïve heuristic for TSPTWD. There are still some other reasons for those poor quality solutions which will be discussed in the following section of the simulated annealing algorithm. In the following sections, I will develop some better metaheuristics to improve the quality of solutions.

Simulated Annealing (SA) Framework for TSPTWD

Simulated annealing (SA) algorithm is a method introduced by Kirkpatrick et al. (1983) to find the global optimal of combinatorial optimization. It has been successfully used in many areas including solving traveling salesman problems (Cerny, 1985, Kirkpatrick et al., 1983). It simulates the physical annealing process with a simple and well-designed searching algorithm to avoid trapped in some local minimal.

I use a one-dimension string to describe a truck a route as shown in Figure 8. The first and last numbers represent the depot. Since there are time windows, the depot cannot be represented as a unique number in the string of the solution. There are different operators for the searching process of SA. I chose a popular one 2-opt (Figure 8) in all my SA algorithms. Cost of

a truck solution (cost(*sol*)) or a truck and a drone ($cost^{TD}$(*transformed_TD*$_{new}$)) is defined in principle 3 of Algorithm 3.

To increase the efficiency of my algorithms, both the SA algorithms in Chapter 3 are developed based on the same SA framework. This framework is an extension of the traditional SA algorithm (Gendreau and Potvin, 2010, Xiao et al., 2012). My SA framework is described in Algorithm 2. Whenever it is necessary to calculate the cost of a solution of a truck and a drone, the SA framework will take two steps. First, transforming a truck solution to a truck and drone solution by calling an algorithm SA_n, i.e. SA_01, or SA_02 which will be discussed in the following sections. Although SA_01, SA_02 are quite different, they return the same format two-dimension matrix (transformed_TD) to represent a truck and a drone solution as shown in Figure 12(f, g) which is transformed from the one-dimension truck solution. Second, the returned solution of a truck and a drone is passed to an algorithm (cost_TD) to calculate its corresponding cost as shown in Algorithm 3.

Algorithm 2. SA framework for TSPTWD

Input:

Solution: string sol_{ini} (initial solution of the truck route of a traveling salesman problem)

Parameters of the instance, e.g. τ, τ', V_d, s_L, s_R, t_{ex}, F, e_i, l_i, E, L (symbols are defined in Table 1)

Parameters of SA (α, T_{end}, LL, ω_{TW}, ω_F) are defined in Table 6

Function SA framework for TSPTWD:

sol := sol_{ini}; sol_{best} := sol_{ini} #set the initial solution as the incumbent and best solutions

Initial temperature T_0 = the maximum cost deviation between two neighboring solutions by running 1000 times the operator on the initial solution

T := T_0 #set the initial temperature as the incumbent temperature

transformed_TD$_{best}$:= SA_n(sol_{ini}) #transform a truck solution to a truck and drone solution

$cost^{TD}_{best}$:= cost_TD(*transformed_TD*$_{best}$) #calculate cost of a truck and drone solution.

While T > T_{end} **do**

For i = 1 to LL **do**

Using the operator to generate sol_{new} based on *sol*

If cost(sol_{new})[a] ≤ cost(*sol*) #Metropolis algorithm

$sol := sol_{new}$

If cost(sol_{new}) ≤ cost(sol_{best})

$sol_{best} := sol_{new}$

$transformed\_TD_{new}$:= SA_n(sol_{new})

$cost_{new}^{TD}$:= cost_TD($transformed\_TD_{new}$)

If $cost_{new}^{TD} < cost_{best}^{TD}$

$transformed\_TD_{best} := transformed\_TD_{new}$

$cost_{best}^{TD} := cost_{new}^{TD}$

End if

End if

Else

sol := sol_{new} with probability $e^{-(cost(sol_{new}) - cost(sol))/T}$

End if

End for

T := T*α

End while

Return $cost_{best}^{TD}$, $transformed\_TD_{best}$

End Function

---

a: Due to the special structure of the truck route of SA_02, when the cost is calculated all the -1s should be removed before it is passed into the function of cost(*sol*).

Algorithm 3. Calculate the cost of a truck and a drone solution (cost_TD)

Input:

transformed_TD #two-dimension matrix, e.g. Figure 12(f, g), to represent a truck and a drone solution

Parameters of the instance, e.g. τ, τ', V_d, s_L, s_R, t_{ex}, F, e_i, l_i, E, L (symbols are defined in Table 1)

Function cost_TD:

Principle 1- The later arrival time of the truck or the drone is the arrival time of the point.

Principle 2- The arrival time back to the depot is the delivery time.

Principle 3- *cost* = delivery time + ω_{TW}*violation of time windows + ω_F*violation of maximum flying time of a drone

Return *cost*

End Function

Improvement Strategies to Improve the Quality of Solutions

Some primary tests showed that naïve method of TSPTWD generated poor quality solutions for many instances which means the cost of a truck and a drone is even large than a pure truck TSP solution. It was surprising to me that I expected more delivery time savings for a truck and a drone mode especially when intermediate points are provided. I studied those instances and found the following several possible reasons leading the poor quality.

First, as a generalized form of TSP, the complexity (or the size of the solution space) of TSPTWD is much larger than that of TSP which means that in the same period it would be more difficult to find a good solution for TSPTWD.

Second, the launching and receiving points are chosen randomly. They may not well 'matched' with drone-visit customers, which might hinder the attempt to save delivery time. If the drone-visit customer is too far away, then it may violate the maximum flying time restriction, e.g. Figure 9(a). If the truck traveling time between the launching and receiving points is shorter than the drone's flying time, then there will be a truck waiting time. The truck's waiting time

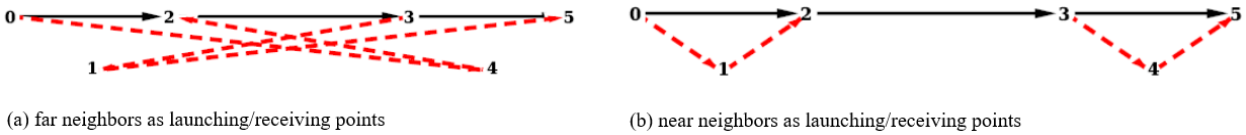
will increase the total delivery time. On the other hand, if the truck traveling time between the launching and receiving points is too long, then the drone has to wait at the receiving point, and finally, it may violate the maximum flying time restriction.

Third, in the naïve method of TSPTWD, drone-visit customers are randomly chosen and pulled out of the original truck route. Those selected customers may not always save time. If we consider the traversing time of the truck, when we pull out a customer and let it be visited by the drone, there should be saved traveling time by the truck. For example in Figure 12(c), after truck route 0-1-2 is replaced by 0-2, and customer 1 is visited by the drone. Then there should traveling time saving $t_{01} + t_{12} - t_{02}$. On the other hand, to visit customer 1 by the drone the driver must spend extra serving time (s_L, s_R, t_{ex}) to launch and receive the drone. If the saved traveling time is larger than the extra serving time ($t_{01} + t_{12} - t_{02} > s_L + s_R + 2*t_{ex}$) then this drone-visit customer can save time. Otherwise, it is increasing the total delivery time.

Only after finding approaches to mitigate the above-mentioned problems will I be able to reduce the delivery time. The following are several improvement strategies I developed to be implemented in the new SA algorithms to improve the quality of solutions.

Strategy 1- near-neighbors strategy: According to the analysis of the above second point, randomly chosen launching and receiving points may no 'match' well with the drone-visit customers. It means sometimes the drone-visit customers are too far away from their launching and receiving points, Figure 9(a). That is why I want to introduce the idea of near neighbors. Its general principle is that we should try to find some neighbors in proximity of a chosen drone-visit customer as its launching and receiving points, Figure 9(b). In the following SA_01, SA_02, I will use two different ways to find some near neighbors for chosen drone-visit customers.

Normally they can find better solutions than the Savings Method and the naïve method of TSPTWD.



(a) far neighbors as launching/receiving points (b) near neighbors as launching/receiving points

Figure 9. near-neighbors strategy

Strategy 2- Center of Gravity (CG) strategy: Based on the third reason for the poor quality solutions from the naïve method of TSPTWD, sometimes the saved traveling time from the truck may be less than the extra serving time of the drone. For example in Figure 10, if the truck traveling time saving is less than the extra serving time of the drone or $t_{01} + t_{12} - t_{02} < s_L + s_R + 2*t_{ex}$ then choosing customer 1 to be visited by the drone would increase our delivery time instead of reducing of it. To mitigate this problem, we should try to pick customers far away from the truck's route to save more time than the extra serving time for the drone. It is worth noting that this analysis suggests that customers far away from others may save more delivery time. In a given area it means lower customer density may lead to more savings. This will be observed in my numerical experiments. Then the next challenge is how to find those drone-eligible customers far away from the truck's route. Normally we can consider a TSP truck route as an approximate shape of a circle (Figure 10). Concentrating drone deliveries to customers further to the center of the circle should increase the probability to reduce delivery time, e.g. customer 4 in Figure 10. Therefore, I developed a strategy called the center of gravity (CG) strategy. CG is a traditional heuristic for single location problems (Ballou, 2004). Here I just need to use the unweighted average approach, i.e. the x or y coordinates of CG is equal to the arithmetic average of the x or y coordinates of all customers and depot. After finding the location of CG, I can calculate the Euclidean distances between CG and each customer. Then whenever a

SA algorithm wants to choose drone-visit customers randomly, it will choose them based on a distribution proportional to a customer' Euclidean distance to CG rather than with an even distribution. Such that the further a customer is from CG the more likely it will be chosen as a drone-visit customer and save more time for the truck and drone system. After implementing this CG strategy, the quality of solutions was enhanced again. Taking a step further, I tried to increase the power of Euclidean distance to CG of customers from the first to the second. The quality of the solutions looked even better. Finally, I increased it to the fourth power because larger powers did not bring significant savings to delivery time.

Strategy 3- Picking and dropping strategy: As explained in the third reason for the poor quality solutions from the naïve method of TSPTWD, sometimes the saved traveling time from the truck may not be larger than the extra serving time of the drone. If that situation happens on a chosen customer, then I should not let the drone visit it since it is wasting time rather than saving time. This means that every time after I picked up a customer out a truck route and try to visit it by a drone, I should calculate whether it is a promising pick (saved truck traveling time is larger than the extra serving time of the drone) or unpromising pick. For example in Figure 10, if the truck's traveling times $t_{01} + t_{12} - t_{02} < s_L + s_R + 2*t_{ex}$ then picking up customer 1 is unpromising, I should drop it back to the truck's route. Otherwise, it is a promising pick (customer 4 in Figure 10, $t_{34} + t_{45} - t_{35} > s_L + s_R + 2*t_{ex}$) and I will keep that customer as a drone-visit customer. That is why I call this strategy as picking and dropping strategy. With this simple and effective criterion, I can make a deterministic decision for any randomly chosen drone eligible customer whether I should accept it as a drone-visit customer without wasting truck delivery time. Furthermore, I will not choose drone-visit customers randomly any more. I pick all of them and evaluate with the above-mentioned criterion to decide whether I will remain a customer as a drone-visit

customer or drop it back to the truck route. Tests showed that this strategy can increase the quality of solutions even further. This strategy is better than the CG strategy since CG strategy only randomly chooses some customers with higher probability to generate time saving, but the picking and dropping strategy can let me pick all drone-eligible customers and evaluate them one by one, retaining all promising one and dropping back all unpromising ones to maximize time saving. Although I did not use the CG strategy in the following metaheuristics, I still want to report it here because it is a good idea. Other researchers and I may want to use it in the future.

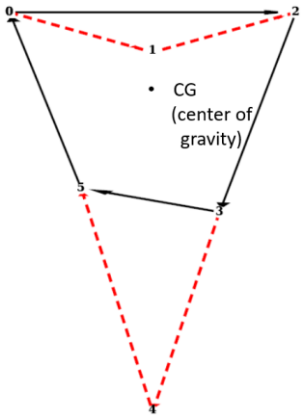


Figure 10. Center of Gravity (CG) strategy; Picking and dropping strategy

Strategy 4- Expanding and shrinking strategy: There are still other opportunities to improve the quality of solutions. As explained in the second reason for the poor quality of solutions of the naïve method of TSPTWD, if the truck traveling time between the launching and receiving points is shorter than the drone's flying time then there will be truck waiting time which will increase the total delivery time. To reduce this truck waiting time, I can expand the truck route between the launching and receiving points. That means I can exchange the launching point with another one to the backward direction or exchange the receiving point with another one to the forward direction of the truck route. Then the 'expanded' truck route between the new launching and receiving points will take a longer time and reduce the truck waiting time.

For example in Figure 11(a), the drone is launched at customer 2 and visits customer 3 and received by the driver at another customer 4. If the truck will wait for the drone at customer 4, then I can expand the truck's receiving point from 4 to customer 5 (Figure 11(b)) or even further until there is no truck waiting time. A similar process can be implemented for another scenario that if the drone's flying time is shorter than the truck's traveling time between the launching and receiving points (0-1-2 in Figure 11(a)), then the drone has to wait for the truck which will not increase the total delivery time but may lead to violation of the maximum flying time restriction. To reduce the drone's waiting time, I can shrink the drone's flights by exchanging the launching point with another one to the forward direction or exchanging the receiving point with another to the backward direction of the truck route. For example, 0-1-2 is shrunk to 0-1-7 where point 7 is an intermediate point on arc (0, 2), Figure 11(b). This operation will shorten the drone's flying time to reduce its waiting time at the receiving point. So, I call this strategy an expanding and shrinking strategy. This strategy will be implemented after the two-dimension matrix of the truck and drone solution has been generated. Since the shrinking actions can provide more opportunities (customers or intermediate points) for the expanding action to choose, this strategy will be done as shrinking first and expanding second.

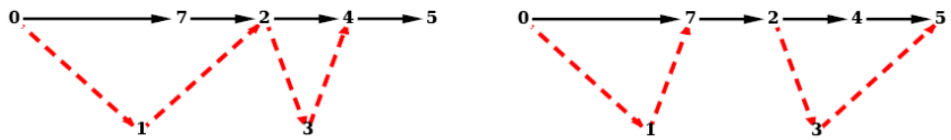


Figure 11. Expanding and shrinking strategy

Simulated Annealing 01 (SA_01)

This algorithm (SA_01) was developed to mitigate the problems discussed above. It is based on the Algorithm 2 SA framework. When we need to transform a truck route into a truck and drone solution, we will use this algorithm. SA_01 will utilize three of the improvement strategies (excluding the CG strategy) to get good quality truck and drone solutions. This process is briefly described in Algorithm 4 SA_01. A detailed example is illustrated in the following paragraph and Figure 12.

Figure 12(a) shows an original tuck route that is passed into this algorithm. According to the picking and dropping strategy, all drone-eligible customers will be picked up at first, i.e. customers 1,2,3,4 in Figure 12(b). To make a feasible solution, half of the consecutive picked up customers will be dropped back to be the launching or receiving points. In Figure 12(b) 1,2,3,4 are consecutive customers, there are not enough launching and receiving points on the truck route (0-5). Then customers 2, 4 will be dropped back to the truck route as launching or receiving points which are shown in Figure 12(b, c). By utilizing the near-neighbors strategy, we choose 0, 2, and 2, 4 as launching and receiving points for customers 1, 3 respectively. Based on the picking and dropping strategy, the unpromising customers will be dropped back to the truck route. Suppose in Figure 12(c) the saved truck traveling time $t_{01} + t_{12} - t_{02} < s_L + s_R + 2*t_{ex}$ which is an unpromising customer. And suppose customer 3 is a promising one. Then we will drop back customer 2 and maintain customer 3 as a drone-visit customer to save delivery time by picking up customers. The result is shown in Figure 12(d). Until now we only used customers as launching and receiving points. According to my new model, I should explore solutions by using intermediate points. Then the next step is to randomly shift the existing launching and receiving points to their nearby intermediate points but not further than the next customer. Suppose the

algorithm randomly searching the nearby neighborhood of the launching point 2 and choosing an intermediate point 7 between customers 1 and 2. Then intermediate point 7 becomes the launching point of the drone to visit customer 3. Similarly, an intermediate point 8 between customers 2, 4 is chosen as the updated receiving point. The truck and drone solution is shown in Figure 12(e). Next, this truck and drone solution will be transformed into a two-dimension matrix as illustrated in Figure 12(f). The first row in that matrix indicates the truck route (7, 8 are intermediate points between customers 1, 2, and 2, 4). The '-1's in the second and fourth rows show the launching and receiving points of the drone. In Figure 12(f) the two '-1's indicate points 7, 8 are launching and receiving points. The third row indicates that customer 3 will be visited by the drone after it is launched from intermediate point 7 and received at intermediate point 8. The last step before returning is to use the expanding and shrinking strategy. If in this example the truck traveling time on 7-2-8 is shorter than the drone's flying time on 7-3-8, then there is truck waiting time which should be avoided as much as possible. Here we need to use the expanding strategy to try to expand the truck's route 7-2-8. The algorithm will search points out of the range of 7-2-8 as far as possible, i.e. cannot expand into other launching and receiving ranges. Suppose the algorithm chooses depot (5) as the new receiving points such that the truck traveling time on 7-2-8-4-5 is greater than or equal to the drone's flying time on 7-3-5. Then there will be no truck waiting time on this segment and the quality of the solution has been improved. Finally, the matrix of the truck and drone solution (*transformed_TD*) after implementing the expanding and shrinking strategy will be returned and this algorithm is done.

Algorithm 4. Simulate Annealing 01 (SA_01. A detailed illustration example can be found in the main body)

Input:

String *sol* (initial solution of the truck route of a traveling salesman problem, e.g. Figure 12(a))

Parameters of the instance, e.g. τ, τ', V_d, s_L, s_R, t_{ex}, F (symbols are defined in Table 1)

Function SA_01:

Step 1. Pick up all customers in V_d out of the truck route as potential drone-visit customers, e.g. Figure 12(a, b).

Step 2. Drop half of the consecutive picked up customers to be sure there are enough near neighbors as launching or receiving points, e.g. Figure 12(b, c).

Step 3. Choose the precedent and successive neighbors as the launching and receiving points of drone-visit customers, e.g. Figure 12(c)

Step 4. According to the picking and dropping strategy, drop unpromising drone-visit customers back to the truck route, e.g. Figure 12(c, d).

Step 5. Randomly shift the launching and receiving points to nearby intermediate points to explore more solutions (not further than one customer), e.g. Figure 12(d, e).

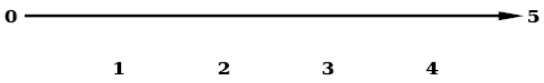
Step 6. Generate a two-dimension matrix *transformed_TD* to represent a truck and a drone solution, e.g. Figure 12(f).

Step 7. Implement the expanding and shrinking strategy to improve the quality of a solution further, e.g. Figure 12(g).

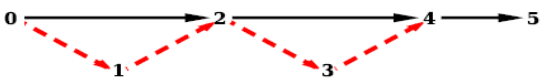
Return *transformed_TD*

End Function

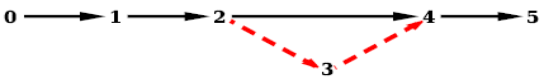
(a) original truck route



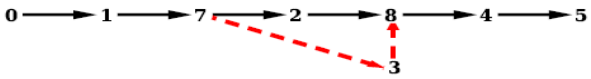
(b) half of the consecutive customers will be dropped back



(c) will drop back '1' if it is unpromising



(d) random shifting of launching/receiving points will be made



(e) truck and drone solution before expanding and shrinking strategy

truck route	0	1	7	2	8	4	5
launching points			-1				
drone-visit customers			3				
receiving points					-1		

(f) corresponding transformed solution

truck route	0	1	7	2	8	4	5
launching points			-1				
drone-visit customers			3				
receiving points							-1

(g) transformed solution after expanding and shrinking strategy

Figure 12. An illustrative example of SA_01 (details can be found in section SA_01).

Simulate Annealing 02 (SA_02)

This algorithm (SA_02) was developed to mitigate the problems discussed above. SA_02 is distinguished from SA_01 since it uses some '-1's to indicate launching and receiving points. After '-1's are inserted and adjusted properly, this adjusted string of solution can be input into Algorithm 2 SA framework to search for good solutions of a truck and a drone. The following paragraph will demonstrate how to insert '-1's. When we need to transform a truck route into a truck and drone solution, we will use this algorithm SA_02. SA_02 will utilize one of the improvement strategies, i.e. expanding and shrinking strategy to get good quality truck and drone solutions. This process is briefly described in Algorithm 5 SA_02. A detailed example is illustrated after the paragraph about inserting '-1's.

Figure 13(a) shows an original truck route within which 0, 6 represent the depot, and 1~5 are customers. To adjust it as a suitable string solution for SA_02, we need to insert $2*|V_d|$ '-1's randomly between the start and end depots, Figure 13(b). If there are greater than or equal to

two ‘-1’s following the depot ‘0’, then one of those ‘-1’s will be moved to the end of the solution string (after the depot 6) to be sure the depot at the end can be a receiving point. After that we have a well-adjusted truck route string solution (Figure 13(c)) to put into Algorithm 2 SA framework. The SA operator will operate on the customers and ‘-1’s between the start and end depot. When we need to calculate the truck cost of an adjusted form just described for SA_02, we need to remove all ‘-1’s temporarily.

Figure 13(c~g) will use an example to illustrate how to transfer a truck route of SA_02 into a truck and drone solution. We iterate each point after 0 the depot in Figure 13(c). The first ‘-1’ after 0 indicates that depot 0 can be a launching point (Principle 1 of Algorithm 5). The two ‘-1’s after 1 means customer 1 can be both a launching and a receiving point (Principle 2). But there are not drone eligible customers between 0 and 1. So we have to ignore the two ‘-1’s after 1 (Principle 3). Then the ‘-1’ after 3 indicates that 3 can be a receiving point. Since there is a potential launching point 0 and two potential drone-visit customers 1,2 (suppose both of them are drone eligible customers), we choose 1 randomly as the drone-visit customer out of 1, 2 (Principle 4). Then we can find two ‘-1’s after 4 which means customer 4 can be a launching and a receiving point. But there is not a launching point in front of 4, customer can only be a launching point even it has two ‘-1’s following. Customer 5 is followed by a ‘-1’ but it cannot be a receiving point since there are no crone eligible customers between 4 and 5 (Principle 3). Finally, there is a ‘-1’ after depot 6 which can be a receiving point. There is one drone eligible customer 5 (suppose it is) between 4 and 6. Then customer 5 will be visited by the drone and 4, 6 are the launching and receiving points. The result is demonstrated in Figure 13(d). Until now we only used customers as launching and receiving points. According to my new model, I should explore solutions by using intermediate points. Then the next step is to randomly shift the

existing launching and receiving points to their nearby intermediate points but not further than the next customer. Suppose the algorithm randomly searching the nearby neighborhood of the launching point 0 and choosing an intermediate point 7 between depot 0 and customer 2. Then intermediate point 7 becomes the launching point of the drone to visit customer 1. Similarly, an intermediate point 8 between customers 2, 3 is chosen as the updated receiving point. The next pair of launching and receiving points 4, 6 do not change since this shifting process is random. The truck and drone solution is shown in Figure 13(e). Next, this truck and drone solution will be transformed into a two-dimension matrix as illustrated in Figure 13(f). The first row in that matrix indicates the truck route (7, 8, 9 are intermediate points between depot 0, customers 2 and customers 2, 3 and 4, 6). The '-1's in the second and fourth rows show the launching and receiving points of the drone. In Figure 13(f) the four '-1's indicate that points 7, 8, and 4, 6 are launching and receiving points. The third row shows that customers 1, 5 will be visited by the drone after it is launched from intermediate point 7 and customer 4 respectively. The last step before returning is to use the expanding and shrinking strategy. We will do shrinking at first and then expanding. If the drone's flying time on 4-5-6 is shorter than the truck's traveling time on 4-9-6, then there is drone waiting time. We can use the shrinking strategy to reduce it. Suppose the algorithm shrinks the launching point from customer 4 to the intermediate point 9 such that the drone waiting is reduced and there is not truck waiting time. After this shrinking operation, we cannot reduce the truck and drone delivery time directly, but we can reduce the probability of exceeding the maximum flying time and provide more opportunities for the following expanding strategy. If in this example the truck traveling time on 7-2-8 is shorter than the drone's flying time on 7-1-8, then there is truck waiting time which should be avoided as much as possible. Here we need to use the expanding strategy to try to expand the truck's route 7-2-8. The

algorithm will search points out of the range of 7-2-8 as far as possible but cannot expand into other launching and receiving ranges. Suppose the algorithm chooses customer 3 as the new receiving point such that the truck traveling time on 7-2-8-3 is greater than or equal to the drone's flying time on 7-1-3. Then there will be no truck waiting time on this segment and the quality of the solution has been improved. Finally, Figure 13(g) shows the matrix of the truck and drone solution (*transformed_TD*) after implementing the expanding and shrinking strategy which will be returned and this algorithm SA_02 is done.

Due to the nature of SA_02, it does not use the picking and dropping strategy. So, I am expecting that SA_01 may generate better solutions than SA_02. The following computational experiments will test it.

Algorithm 5. Simulate Annealing 02 (SA_02. A detailed illustration example can be found in the main body)

Input:

String *sol* (initial solution of the truck route with '-1's, e.g. Figure 13(c))

Parameters of the instance, e.g. τ, τ', V_d, s_L, s_R, t_{ex}, F (symbols are defined in Table 1)

Function SA_02:

Step 1. Generate a truck and drone solution based on the following principles, e.g. Figure 13(d).

Principle 1: If a customer or the depot is followed by a '-1', then that customer or depot is eligible to be a launching or receiving point.

Principle 2: If a customer or the depot is followed by greater than or equal to two '-1's, then that customer or depot is eligible to be a launching and a receiving point.

Principle 3: If there are no drone-eligible customers between a pair of launching and receiving points, then ignore this potential receiving point. But the launching point is still active.

Principle 4: If there are multiple drone-eligible customers between a pair of launching and receiving points, then one of them is randomly chosen as the drone-visit customer.

Step 2. Randomly shift the launching and receiving points to nearby intermediate points to explore more solutions (not further than one customer), e.g. Figure 13(e).

Step 3. Generate a two-dimension matrix *transformed_TD* to represent a truck and a drone solution, e.g. Figure 13(f).

Step 4. Implement the expanding and shrinking strategy to improve the quality of a solution further, e.g. Figure 13(g).

Return *transformed_TD*

End Function

0	1	2	3	4	5	6

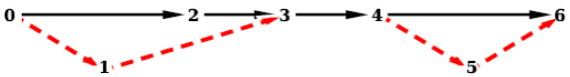
(a) original truck route

0	-1	-1	1	-1	-1	2	3	-1	4	-1	-1	5	-1	6

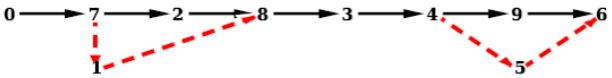
(b) insert '-1's randomly

0	-1	1	-1	-1	2	3	-1	4	-1	-1	5	-1	6	-1

(c) Move a '-1' to the end if necessary



(d) identify drone-visit customers, launching and receiving points



(e) insert intermediate points; random shifting launching & receiving points

truck route	0	7	2	8	3	4	9	6
launching points		-1				-1		
drone-visit customers		1				5		
receiving points				-1				-1

truck route	0	7	2	8	3	4	9	6
launching points		-1					-1	
drone-visit customers		1					5	
receiving points					-1			-1

Figure 13. An illustrative example of SA_02 (one drone)

Computational Experiments:

I arranged the following computational experiments to test the effectiveness of those heuristics and how my new model can provide further benefits to the practice of a truck and a drone delivery mode.

Experimental Factors and Parameters

A full factorial experiment has been designed and implemented. The design and values or ranges of factors and parameters are discussed in this section.

According to my studies, several factors may affect the delivery time saving and they are set as factors in the following experiments. They are the length of sides of a square area that contains customers to be served. Literature shows that the serving areas of typical parcel delivery carriers could be as small as 1.5*1.5 mile^2 or as large as 8*8 mile^2, (Lammert and Walkowicz, 2012), number of customers as large as about 100 (Don, 2013). In my model, there are limited discrete intermediate points on each arc since in the real world some areas may not be suitable as drone operating points, e.g. gas stations, crossings, etc. I set its value between 0 and 3 within which 0 means there are no intermediate points, or the drone can only be operated on customers' sites as the existing model.

Generally, the values or ranges of parameters are based on literature or technical reports. For example, more than 80% of parcels are suitable for drone delivery, i.e. weighing less than five pounds, (Guglielmo, 2017), or percentage of customers having requirements of time windows is 4% (normally they are business customers) and the random ratio of time windows vs the total delivery time is between [20%~65%] (Don, 2013). For example, if the total delivery time is about ten hours (8am~6pm) then a 50% random ratio means the time window for this customer is between 8am~1pm. I set 10% of customers cannot be served by the drone due to their restricted

locations, e.g. near airports, power plants, in high-density areas, etc. This is based on feedback from practitioners and researchers. Note that this 10% of customers might overlap with those 80% drone-eligible customers. For example, a customer is a drone-eligible customer since his/her parcel weighing less than five pounds, but we may not be able to serve him/her by the drone since the customer locates in a restricted area or falls in the set of that 10% customers. Values and ranges of all selected parameters can be found in Table 6.

Some parameters of simulated annealing (SA) were set based on literature, e.g. initial temperature, others were based on tests of my experiments, e.g. ending temperature, level loop and penalty weights of time windows, and maximum flying time. A summary of their values can be found in Table 6.

Table 6 Factors and selected parameters of the computational experiments

Category	Values or ranges	Data source
Experimental factors		
Length of side (of a squared area)	[1.5, 5, 8] miles	A
Number of customers	[60, 80, 100]	B
Number of intermediate points per arc	[0, 1, 2, 3]	-
Number of drones (for Chapter 4)	[1, 2, 3]	-
Parameters for instances		
percentage of customers eligible for drone delivery due to light weights of parcels	80%	C
percentage of customers cannot be served by the drone due to their restricted locations	10%	-
percentage of customers with time windows	4%	B
Ratio of time windows vs total delivery time	[20%~65%]	B
Distances of the truck	Manhattan distance	D
Distances of the drone	Euclidean distance	-
Average speed of the truck (serving time included)	12 mph	A, B
Average speed of the drone	40 mph	E
Serving time to launch or receive a drone (s_L, s_R)	1 minute	D
Maximum flying time of a drone (F)	30 minutes	E
Extra stopping time of the truck traversing out-of-route to operate drone(s) (t_{ex})	0.1466 minute	G

Table 6 continued

Table 6 Factors and selected parameters of the computational experiments

Parameters for simulated annealing		
Operator	2-opt	F
Initial temperature (T_0)	in SA framework.	F
Cooling ratio (α)	0.99	F
Ending temperature (T_{end})	1.0	-
Level loop (LL)	1,000	-
Penalty weight of time windows (ω_{TW})	1.0	-
Penalty weight of maximum flying time (ω_F)	6.0	-
Number of random runs for each instance	10	-

Data source: A = Lammert and Walkowicz (2012), Don (2013); B = Don (2013); C = Guglielmo (2017); D = Murray and Chu (2015); E = Perez and Kolodny (2017); F = Xiao et al. (2012); G = Appendix A

Experimental Design

To compare those developed heuristics and effectiveness of my new model, I employed full factorial experiments with three factors, i.e. length of sides of a square area (three values), number of customers (three values), number intermediate points on each arc (four values). I call each combination of the values of the three factors as a scenario. For each of such a scenario, I generated five instances randomly. Each of the instances would be solved by four algorithms, i.e. Savings Method, naïve method of TSPTWD, SA_01, SA_02. Due to the randomness of the four algorithms, I have let each of them take ten runs based on the same instance. First, the Savings Method started from the initial solution run ten times. The average cost of the feasible solutions as the result of this algorithm of this instance. The best solution to the Savings Method will be the initial solution for all other algorithms. Each of them will take ten runs and save the average cost of feasible solutions as the result of this algorithm of this instance. The average cost of the five instances as the result of a scenario of an algorithm. Those results will be saved as the final results of these experiments. Then totally the experiments will take 3*3*4*5*4*10 = 7,200 runs.

I implemented the algorithms and experiments mainly in Julia (1.0.4) which is a high-performance language designed for scientific computation (Bezanson et al., 2017). The program was run on a desktop PC (Core i7 2.4GHz quad-core PC with 32GB of memory, Windows system).

Results

The results of the average delivery times of the computational experiments can be found in Appendix C. The impacts of the factors are reported as follows.

The comparison of algorithms is summarized in Table 7. The feasibility percentage shows how many feasible solutions were found within all of the 450 runs. It shows that SA_01 has the highest percentage (100%) of feasibility and the naïve method of TSPTWD has the lowest one (85.8%). The average delivery time (of feasible solutions) demonstrates how good the quality is for the solutions found by an algorithm. The result of the Savings Method for TSP can be considered as the benchmark. SA_01 found solutions with lowest average delivery time (203.94 minutes) while the naïve method found solutions of a truck and a drone with even higher delivery time (279.71 minutes) than pure truck mode (246.55 minutes). The main reason for the naïve method found worse solutions than Savings Method is that it did not utilize any improvement strategies, such that the truck's waiting time is high and the saved time by truck route may not be large enough than the operating time of the drone. SA_01 can always find feasible solutions and with higher quality. Then in the following part of this chapter, I will report data from this algorithm to show the impacts of factors.

Table 7. Comparison of algorithms

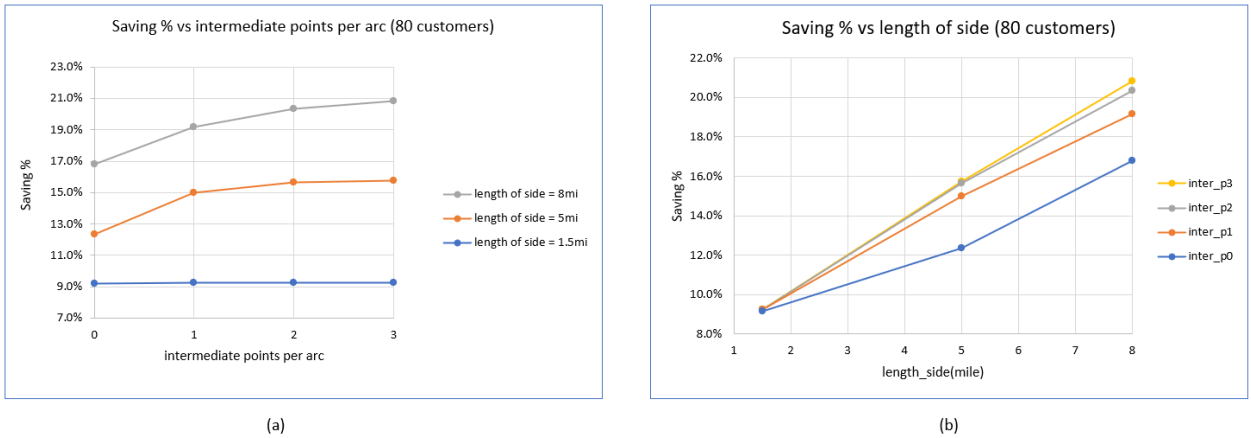
	Savings_Method	naive_TSPTWD	SA_01	SA_02
feasibility %	98.9%	85.8%	100.0%	99.0%
average delivery time (minute)	246.55	279.71	203.94	217.28

Table 8 illustrates the impacts of the number of intermediate points on each arc. For a given number of customers and area, normally the delivery time saving percentage will increase when more intermediate points are provided. We can also find that for a given number of customers, a larger area will lead to more savings. It implies that lower customer density provides more opportunities for delivery time saving by deploying a drone. Figure 14(a) shows a chart of a typical situation.

Table 8 also shows the impact of the length of side (or size of an area) on delivery time saving percentage. For a given number of customers, if we increase the length of side (or size of an area) then we will find more delivery time savings. That is consistent with my analysis of the CG strategy. This trend is also demonstrated in Figure 14(b). This figure also infers that the marginal saving percentages decrease when there is a larger number of drones. It implies that too many drones may not bring us large percentages of delivery time savings. A possible reason is that an excessive number of drones require the truck driver's collaboration. The driver must wait until the last drone is received at a point, then he or she can continue the journey to the next customer. Those collaborations and waiting may increase the total delivery time which diminishes the savings from a larger number of drones.

Table 8. Delivery time saving % with one truck one drone, SA_01 metaheuristic.

#_cus	length_side(mile)	inter_p0	inter_p1	inter_p2	inter_p3
60	1.5	5.8%	5.8%	5.8%	5.8%
60	5	18.1%	20.9%	21.4%	21.4%
60	8	20.6%	22.8%	22.8%	22.8%
80	1.5	9.2%	9.3%	9.3%	9.3%
80	5	12.4%	15.0%	15.7%	15.7%
80	8	16.8%	19.2%	20.4%	20.8%
100	1.5	6.4%	6.4%	6.5%	6.5%
100	5	11.8%	14.5%	14.9%	15.6%
100	8	21.7%	25.6%	26.0%	26.0%



(a) (b)

Figure 14. Delivery time saving % vs # of intermediate points or length of side, 1 drone

The data of the impacts of the number of customers is also contained in Table 8. For a given area, increasing the number of customers did not lead to a monotonic trend of saving percentage, which was a surprising result. A typical chart is shown in Figure 15(a). After analyzing the nature of these problems, I think perhaps the reason is that random instances were generated with different numbers of customers, random time windows, random locations. Those would bring unpredictable randomness to the problems. The literature also supports my analysis that sometimes algorithms generate unexpected solutions (with longer traveling time) due to the constraints of some customers' time windows (Holland et al., 2017). Then I ran the experiment again with no time windows. A typical result is shown in Figure 15(b) with a monotonic decreasing trend of delivery time saving versus the number of customers in a given area. It is consistent with my analysis that higher customer density leads to lower savings. This result also supports my analysis in Chapter 2 that delivery time saving is non-monotonic versus the number of customers. Please note that the saving percentages are reduced when time windows are removed. A possible explanation is that time windows make it more difficult to find good solutions, e.g. UPS stated that their truck may visit an area twice due to time windows (Holland

et al., 2017), such that there might be more chances to save time when there are time windows and vice versa.

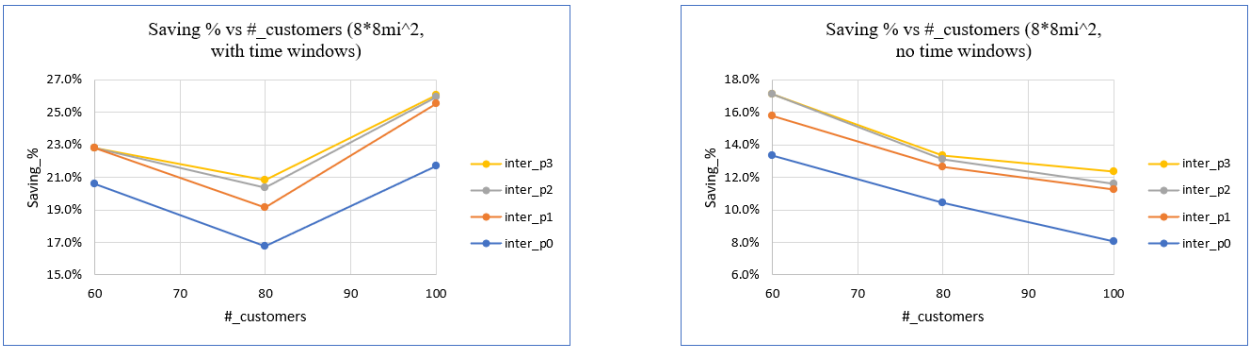


Figure 15. Typical charts of delivery time saving % vs the number of customers

Conclusion

In Chapter 3 I developed multiple heuristics and metaheuristics to solve TSPTWD problems as large as 100 customers which is comparable to the size of problems in the real world.

The problem of poor-quality solutions had been met and a series of improvement strategies had been initiated. Those strategies were developed based on the specific nature of this novel problem TSPTWD. When those strategies were implemented the metaheuristics could generate much higher quality solutions.

Numerical experiments were designed and implemented to test the developed metaheuristics and the impact of factors. The results showed that the algorithm SA_01 outperformed SA_02 and the naïve method of TSPTWD. The idea of my research to provide intermediate points on arc did generate more delivery time savings than only operating the drone on customers' sites. The results show that the delivery time could be further reduced as much as 4% and the actual time saving could be as much as 26%. UPS says for its 55,000 drivers, even a

reduction of one mile per driver per day will lead to about $50 million annual saving (UPS, 2016), i.e. about $2.5 per mile. The typical operating cost (including purchasing, electricity, maintenance, monitoring costs) of a drone is about $0.044 per mile (Keeney, 2015). Based on the experiments, it may be assumed to save one mile for the truck the drone flies two miles. The one-mile cost reduction with a drone is $2.4 or $48 million per year. One mile is about 1% of a typical route of UPS. Then the 26% saving from my model is approximately equivalent to an annual saving of $1.25 billion. This huge saving would be attractive to practitioners and customers.

As I analyzed in the part of the center of gravity (CG) improvement strategy, lower customer density may provide more opportunities for delivery time saving. The impact of the size of areas supports this analysis. But the impact of the number of customers is not so predictable. This is due to the randomness of time windows. When all time windows were removed the results are consistent as what I analyzed.

The computational results also showed that too many intermediate points on an arc do not provide directly proportional time savings. That means a limited number of intermediate points on arcs would be enough to provide the most savings. It is consistent with the situation in the real world that due to some constraints, e.g. gas stations, crossings, bus stops, etc., we can only provide a limited number of intermediate points on arcs.

CHAPTER 4. TRAVELING SALESMAN PROBLEM WITH TIME WINDOWS AND MULTIPLE DRONES (TSPTWMD): LAUNCHED AND RECEIVED AT INTERMEDIATE POINTS

Model Formulation

Problem Definition and Illustrations

The results of Chapter 3 (essay 2) demonstrated a large amount of delivery time saving (as large as 26%) by providing intermediate points to a truck and a drone system. A natural extension of the previous model is to utilize not only one drone but multiple drones for one truck, e.g. Figure 16(f, g). The benefits of multiple drones are obvious that several drones can serve different customers at the same time with the truck to increase delivery efficiency and reduce delivery time. In Chapter 4 (essay 3) I will focus on developing a new model of traveling salesman problem with time windows and multiple drones (TSPTWmD) with intermediate points. The assumptions are the same as in Chapter 3 for TSPTWD. If drones arrive at a point earlier than the truck, they must hover in the air to wait for the truck. When the truck arrives, the driver will operate the first drone at first and then the second and the third.

Proposed Model TSPTWmD

The model is developed based on the model of essay 1 (Chapter 2), i.e. (1) ~ (6f). Parts of the new model is similar to the one I developed in essay 1. It is assumed that there are n_d drones working together with the truck and the set of drones D = {1, 2,…, n_d}. The challenge of this new one is how to collaborate multiple drones with the truck and also between each of them. To accomplish those goals we introduce a new set of binary decision variables $Z_{ijr}^{\alpha,\beta}$ to determine the relative sequence of operating drones on each point. We assume that after operating (receiving and/or launching) a drone completely the truck driver will operate the next drone rather than mixing operations of different drones. This assumption will not only simplify the model but also

easier for a driver to implement in the real world. Another new set of binary decision variables z_{ijr}^{α} is to indicate whether the α^{th} drone is operated on the point ijr. Decision variables $t_{ijr}^{\prime\alpha}$ are the arrival times of the α^{th} drone on the point ijr. The explanation of the sets, parameters, and the mathematical model is as the following.

Notations of parameters and decision variables are described in Table 9.

Table 9. Sets, parameters, and decision variables of the TSPTWmD model

Sets, Parameters:	
0, n+1	Depot
V	Set of all customers {1,2,…..,n}
V_d	Set of eligible customers which can be served by the drone. $V_d \subseteq V$
$V_{0,n+1}$	$V_{0,n+1} = V \cup \{0, n+1\} = \{0,1, \ldots\ldots, n, n+1\}$
V_0	$V_0 = V \cup \{0\} = \{0,1, \ldots\ldots, n\}$
V_{n+1}	$V_{n+1} = V \cup \{n+1\} = \{1, \ldots\ldots, n, n+1\}$
V'	Set of visits to the customers, including V and dummy points of each element of V to permit multiple visits to them.
$V_0', V_{n+1}', V_{0,n+1}'$	$V_0' = V' \cup \{0\}, V_{n+1}' = V' \cup \{n+1\}, V_{0,n+1}' = V' \cup \{0, n+1\}$
s_{ij}	Number of segments on the arc ij within which there are s_{ij}-1 intermediate points (Figure 3). $i \in V_0$, $j \in V_{n+1}$
R_{ij}	Set of all intermediate points on the arc ij. R_{ij} = {ij1, ij2,…, ijr, …, ij(s_{ij}-1)}, $i \in V_0$, $j \in V_{n+1}$
R	Set of all the intermediate points
R'	Set of visits to all the intermediate points, including R and dummy points of each element of R to permit multiple visits to them.

Table 9 Continued.

Table 9. Sets, parameters, and decision variables of the TSPTWmD model

$R'_0, R'_{0,n+1}$	$R'_0 = R' \cup \{0\}, R'_{0,n+1} = R' \cup \{0, n+1\}$
$V^*, V^*_0, V^*_{n+1}, V^*_{0,n+1}$	$V^* = V' \cup R', V^*_0 = V'_0 \cup R', V^*_{n+1} = V'_{n+1} \cup R', V^*_{0,n+1} = V'_{0,n+1} \cup R'$
[ijr$_1$, k, lmr$_2$]	a branch route of the drone departing from the truck at the point of ijr$_1$ to visit a customer k then meet the truck at the point lmr$_2$. $ijr_1 \in V^*_0, lmr_2 \in V^*_{n+1}, ijr_1 \neq k \neq lmr_2, k \in V_d$.
B	Set of all the possible branch routes [ijr$_1$, k, lmr$_2$] of drones.
n_d	The number of drones.
D	Set of all drones {1,2,…..,n_d}.
$\tau_{[ijr,\ ij(r+1)]}$	Time interval of the truck traversing from the diversion point of ijr to the diversion point of ij(r+1). ijr $\in V^*_0$.
t_{ex}	The extra stopping time of the truck traversing out-of-route to decelerate and then accelerate again at an intermediate point. Details can be found in Appendix A.
$\tau'_{[ijr1,\ k]}, \tau'_{[k,\ lmr2]}$	Time interval of the drone traversing from an intermediate point ijr$_1$ to customer k and from k to lmr$_2$. ijr$_1$ $\in V^*_0$, [ijr$_1$, k, lmr$_2$] $\in$B
s_L	Preparing time interval by the driver for launching the drone.
s_R	Receiving and recovering time interval of the drone which may include the driver changing batteries etc.
F	Maximum flying time of the drone.
[e_i, l_i]	Time windows for all customers and the depot. i$\in V_{0,n+1}$.
[E, L]	The time window of the depot, [e_0, l_0] = [e_{n+1}, l_{n+1}] = [E, L].
δ	$\delta > 0$, a small enough positive number.
M	M > 0, a big enough positive number.

Table 9 Continued.

Table 9. Sets, parameters, and decision variables of the TSPTWmD model

Decision variables:	
x_{ij}	Binary decision variable indicating whether the arc ij is traversed by the truck. $\forall i \in V_0', j \in V_{n+1}', i \neq j$.
$y^{\alpha}_{[ijr_1,k,lmr_2]}$	Binary decision variable indicating whether the branch route $[ijr_1, k, lmr_2]$ is traversed by the α^{th} drone. $ijr_1 \in V_0^*$, $lmr_2 \in V_{n+1}^*$, $ijr_1 \neq k \neq lmr_2$, $k \in V_d$, $\alpha \in D$.
z^{α}_{ijr}	Binary decision variable indicating whether the intermediate point ijr is visited by the truck either to launch or receive the α^{th} drone or do both. $ijr \in V_{0,n+1}^*$, $\alpha \in D$.
z_{ijr}	Binary decision variable indicating whether the intermediate point ijr is visited by the truck either to launch or receive a drone or do both for multiple drones. $ijr \in V_{0,n+1}^*$.
$Z^{\alpha,\beta}_{ijr}$	Binary decision variable, equals 1 if the α^{th} drone is operated (receiving or launching) earlier than the β^{th} drone at the intermediate point ijr. The value is 0 if they are operated in the opposite sequence. $\alpha, \beta \in D,\ ijr \in V_{0,n+1}^*$.
$p^{\alpha}_{[der_1,ijr_3]}$	Binary decision variable, equals to 1 when the α^{th} drone visits the point der_1 earlier than ijr_3, and 0 otherwise. $\alpha \in D$, der_1, $ijr_3 \in V^*$, $ijr_3 \neq der_1$. Referring to Figure 5. $p^{\alpha}_{[0,ijr]} = 1\ \forall ijr \in V^*$.
$t_{ijr} \geq 0$	Arrival time instant of the truck at the point $ijr \in V_{0,n+1}^*$ after all drones have been launched.

Table 9 Continued.

Table 9. Sets, parameters, and decision variables of the TSPTWmD model

$t'^{\alpha}_{ijr} \geq 0$	Arrival time instant of the α^{th} drone at the point ijr$\in V^*_{0,n+1}$ after the drone has been received, $\alpha \in D$.

The objective of this model TSPTWmD is to minimize the total delivery time.

$$(TSPTWmD)\ \ Min\ t_{n+1} \tag{14}$$

The following constraints make sure the routes of the truck and drones are valid.

$$s.t. \sum_{h\epsilon V'_0, h\neq k} x_{hk} + \sum_{\alpha\epsilon D} \sum_{ijr_1\epsilon V^*_0} \sum_{lmr_2\epsilon V^*_{n+1}} y^{\alpha}_{[ijr_1,k,lmr_2]} = 1 \quad \forall k\ \epsilon\ V \tag{15a}$$

$$\sum_{j\epsilon V'_{n+1}} x_{oj} = 1 \tag{15b}$$

$$\sum_{i\epsilon V'_0} x_{i(n+1)} = 1 \tag{15c}$$

$$\sum_{i\epsilon V'_0, i\neq j} x_{ij} = \sum_{i\epsilon V'_{n+1}, i\neq j} x_{ji} \quad \forall j\ \epsilon\ V' \tag{15d}$$

$$\sum_{k\epsilon V_d} \sum_{lmr_2\epsilon V^*_{n+1}} y^{\alpha}_{[ijr_1,k,lmr_2]} \leq 1 \quad \forall ijr_1\ \epsilon\ V^*_0, \alpha\epsilon D \tag{15e}$$

$$\sum_{ijr_1\epsilon V^*_0} \sum_{k\epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} \leq 1 \quad \forall lmr_2\ \epsilon\ V^*_{n+1}, \alpha\epsilon D \tag{15f}$$

$$2y^{\alpha}_{[ijr_1,k,lmr_2]} \leq x_{ij} + x_{lm} \quad \forall [ijr_1,k,lmr_2]\epsilon\ \{B|\forall ijr_1, lmr_2\ \epsilon\ R'\}, \alpha\epsilon D \tag{15g\_01}$$

$$2y^{\alpha}_{[i,k,lmr_2]} \leq \sum_{j\epsilon V'_{n+1}\setminus\{i,k\}} x_{ij} + x_{lm} \quad \forall [i,k,lmr_2]\epsilon\ \{B|\forall i\ \epsilon\ V'_0, lmr_2\ \epsilon\ R'\}, \alpha\epsilon D \tag{15g\_02}$$

$$2y^{\alpha}_{[ijr_1,k,m]} \le x_{ij} + \sum_{l \epsilon V_0' \setminus \{m,k\}} x_{lm} \qquad \forall [ijr_1,k,m] \epsilon \{B | \forall ijr_1 \epsilon R', m \epsilon V'_{n+1}\}, \alpha \epsilon D \qquad (15g\_03)$$

$$2y^{\alpha}_{[i,k,m]} \le \sum_{j \epsilon V'_{n+1} \setminus \{i,k\}} x_{ij} + \sum_{l \epsilon V_0' \setminus \{m,k\}} x_{lm} \qquad \forall [i,k,m] \epsilon \{B | \forall i \epsilon V_0', m \epsilon V'_{n+1}\}, \alpha \epsilon D \qquad (15g\_04)$$

Constraint (15a) states that each customer must be visited once by either the truck or a drone. Equations (15b), (15c) require the truck to leave and back to the depot only one time respectively. The truck's flow in equals the flow out of a customer is restricted by equation (15d). Constraints (15e), (15f) limit that any drone cannot be launched or received more than one time at any point. The set of inequalities (15g) are set to assure that the truck must traverse an arc if the drone is launched or received at any point on the arc or a node. Launches and receptions of multiple drones on the same arc or the same point are allowed.

I set the time tracking of the truck.

$$\sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} + \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V^*_{n+1}} y^{\alpha}_{[lmr_2,n,pqr_3]} \le 2z^{\alpha}_{lmr_2} \le 2 \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} +$$

$$2 \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V^*_{n+1}} y^{\alpha}_{[lmr_2,n,pqr_3]}) \qquad \forall lmr_2 \epsilon V^*_{0,n+1}, \alpha \epsilon D \qquad (16a)$$

$$\sum_{\alpha \epsilon D} z^{\alpha}_{lmr_2} \le n_d z_{lmr_2} \le n_d \sum_{\alpha \epsilon D} z^{\alpha}_{lmr_2} \qquad \forall lmr_2 \epsilon V^*_{0,n+1}, \alpha \epsilon D \qquad (16b)$$

$$t_{lmr_2} \ge t_{lm(r_2-1)} + \frac{1}{2} t_{ex} z_{lm(r_2-1)} + \tau_{[lm(r_2-1),lmr_2]} + \frac{1}{2} t_{ex} z_{lmr_2} - M_{4[lm(r_2-1),lmr_2]} \cdot (1 - x_{lm})$$
$$\forall lmr_2 \epsilon V^*_{n+1}, lm(r_2-1) \epsilon V_0^* \qquad (16c)$$

$$e_k \left(1 - \sum_{\alpha \epsilon D} \sum_{ijr_1 \epsilon V_0^*} \sum_{lmr_2 \epsilon V^*_{n+1}} y^{\alpha}_{[ijr_1,k,lmr_2]}\right) \le t_k \le l_k \left(1 - \sum_{\alpha \epsilon D} \sum_{ijr_1 \epsilon V_0^*} \sum_{lmr_2 \epsilon V^*_{n+1}} y^{\alpha}_{[ijr_1,k,lmr_2]}\right)$$
$$\forall k \epsilon V_{0,n+1} \qquad (16d)$$

Constraints (16a) determines $z_{lmr_2} = 1$ when an intermediate point is visited by the truck either to launch or receive the α^{th} drone or to do both. (16b) forces the truck to visit the

intermediate point ijr if any drone (e.g. the α^{th} drone) utilized this point. Arrival time instants of the truck are defined by constraint (16c) within which the receiving and launching times and the time of out-of-route are considered. We can set a big enough positive number $M_{4[lm(r_2-1),lmr_2]} = L + \tau_{[lm(r_2-1),lmr_2]} + t_{ex}$. To make the constraints more tightly, we can set $M_{4[lm(r_2-1),lmr_2]} = max(l_l, l_m) - min(e_l, e_m) + \tau_{[lm(r_2-1),lmr_2]} + t_{ex}$. Time window constraints of customers visited by the truck are described by constraint (16d).

I set the time tracking of drones.

$$t'^{\alpha}_{lmr_2} \geq t'^{\alpha}_{ijr_1} + s_L + \tau'_{[ijr_1,k]} + \tau'_{[k,lmr_2]} + s_R + M_{5[ijr_1,k,lmr_2]} \cdot \left(y^{\alpha}_{[ijr_1,k,lmr_2]} - 1\right) \quad \forall [ijr_1, k, lmr_2] \epsilon B, \alpha \in D \qquad (17a)$$

$$t'^{\alpha}_{lmr_2} \geq t_{lm(r_2-1)} + \frac{1}{2} t_{ex} z_{lm(r_2-1)} + \tau_{[lm(r_2-1),lmr_2]} + \frac{1}{2} t_{ex} z_{lmr_2} + s_R \left(\sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} \right)$$
$$- M_{6[lm(r_2-1),lmr_2]} \cdot \left(1 - z^{\alpha}_{lmr_2}\right) \quad \forall lmr_2 \epsilon V_{n+1}^*, lm(r_2 - 1) \epsilon V_0^*, \alpha \in D \qquad (17b)$$

$$t'^{\alpha}_{lmr_2} + s_L \left(\sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]} \right) \leq t_{lmr_2} + L\left(1 - z^{\alpha}_{lmr_2}\right) \quad \forall lmr_2 \epsilon V_{n+1}^*, \alpha \in D \qquad (17c)$$

Arrival times of drones at a point is set in the constraint (17a). We can assign $M_{5[ijr_1,k,lmr_2]} = max\left(l_i, l_j\right) - min(e_l, e_m) + \tau'_{[ijr_1,k]} + \tau'_{[k,lmr_2]} + s_R + s_L$. To be sure drones can be operated only after the truck arrives a point (17b, 17c) are set. We have $M_{6[lm(r_2-1),lmr_2]} = max(l_l, l_m) - min(e_l, e_m) + \tau_{[lm(r_2-1),lmr_2]} + t_{ex}$.

$$t'^{\beta}_{ijr_1} - t'^{\alpha}_{ijr_1} \leq M_{7[ijr_1]} \cdot Z^{\alpha,\beta}_{ijr_1} \quad \forall ijr_1 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (17d)$$

$$t'^{\alpha}_{ijr_1} - t'^{\beta}_{ijr_1} \leq M_{7[ijr_1]} \cdot (1 - Z^{\alpha,\beta}_{ijr_1}) \quad \forall ijr_1 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (17e)$$

$$\left| t'^{\alpha}_{lmr_2} - t'^{\beta}_{lmr_2} \right| \geq Z^{\alpha,\beta}_{lmr_2} \left(s_L \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]} + s_R \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\beta}_{[ijr_1,k,lmr_2]} \right) +$$
$$(1 - Z^{\alpha,\beta}_{lmr_2}) \left(s_L \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\beta}_{[lmr_2,n,pqr_3]} + s_R \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} \right) -$$

$$(s_L + s_R)\left(2 - z^{\alpha}_{lmr_2} - z^{\beta}_{lmr_2}\right) \qquad \forall lmr_2 \epsilon V^*_{n+1}, \alpha, \beta \in D, \alpha \neq \beta \qquad (17f)$$

$$e_k \sum_{ijr_1 \epsilon V^*_0} \sum_{lmr_2 \epsilon V^*_{n+1}} y^{\alpha}_{[ijr_1,k,lmr_2]} \leq t'^{\alpha}_{ijr_1} + s_L + \tau'_{[ijr_1,k]} \leq L + (l_k - L) \sum_{ijr_1 \epsilon V^*_0} \sum_{lmr_2 \epsilon V^*_{n+1}} y^{\alpha}_{[ijr_1,k,lmr_2]}$$

$$\forall ijr_1 \epsilon \{V^*_0 | [ijr_1, k, lmr_2] \epsilon B\}, k \epsilon V_d, \alpha \in D \qquad (17g)$$

$$t'^{\alpha}_{lmr_2} - t'^{\alpha}_{ijr_1} \leq F + (L - F)\left(1 - y^{\alpha}_{[ijr_1,k,lmr_2]}\right) \qquad \forall [ijr_1, k, lmr_2] \epsilon B, \alpha \in D \qquad (17h)$$

(17d) and (17e) are to determine the correct value of $Z^{\alpha,\beta}_{lmr_2}$. $M_{7[ijr_1]}$ is a positive big enough number. To make the constraints more tightly, we can set it equal to L - E. If we want to make it even tighter, its value can be $M_{7[ijr_1]} = l_j - e_i$. Constraint (17f) is assigned to maintain an appropriate relationship of arrival times between each pair of drones at a given point. Please note that (17f) is nonlinear. It can be transformed into a set of linear constraints which are shown in Appendix B. Customers' time windows and flying time limit for the drone are restricted by constraints (17g), (17h) respectively.

Crossed branch routes as shown in Figure 6 are prohibited by constraints (18a)~(18c).

$$t'^{\alpha}_{ijr_3} - t'^{\alpha}_{der_1} \leq M_{8a[der_1,ijr_3]} \cdot p^{\alpha}_{[der_1,ijr_3]} \qquad \forall der_1 \epsilon V^*_0, ijr_3 \epsilon \{V^* | ijr_3 \neq der_1\}, \alpha \in D \qquad (18a)$$

$$t'^{\alpha}_{der_1} - t'^{\alpha}_{ijr_3} \leq M_{8b[der_1,ijr_3]} \cdot \left(1 - p^{\alpha}_{[der_1,ijr_3]}\right) \qquad \forall der_1 \epsilon V^*_0, ijr_3 \epsilon \{V^* | ijr_3 \neq der_1\}, \alpha \in D \qquad (18b)$$

$$t'^{\alpha}_{ijr_3} \geq t'^{\alpha}_{ghr_2} - L(3 - \sum_{f \epsilon V_d} y^{\alpha}_{[der_1,f,ghr_2]} - \sum_{k \epsilon V_d} \sum_{lmr_4 \epsilon V^*_{n+1}} y^{\alpha}_{[ijr_3,k,lmr_4]} - p^{\alpha}_{[der_1,ijr_3]})$$

$$\forall der_1 \epsilon V^*_0, ghr_2 \epsilon V^*_{n+1}, ijr_3 \epsilon \{V^* | ijr_3 \neq der_1\}, \alpha \in D \qquad (18c)$$

$M_{8a[der_1,ijr_3]}$ and $M_{8b[der_1,ijr_3]}$ are two positive big enough numbers. To make the constraints more tightly, we can set them equal to L - E. If we want to make them even tighter, their values can be $M_{3a[der_1,ijr_3]} = \max(l_i, l_j) - \min(e_d, e_e)$, $M_{3b[der_1,ijr_3]} = \max(l_d, l_e) - \min(e_i, e_j)$.

Decision variables are defined in (19a)~(19g).

$$t_0 = t'^{\alpha}_0 = 0 \quad \forall \alpha \in D \qquad (19a)$$

$$t_{ijr}, t'^{\alpha}_{ijr} \geq 0 \quad \forall ijr \in V^*_{0,n+1}, \alpha \in D \tag{19b}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in V'_0, j \in V'_{n+1}, i \neq j \tag{19c}$$

$$y^{\alpha}_{[ijr_1,k,lmr_2]} \in \{0,1\} \quad \forall ijr_1 \in V^*_0, k \in V_d, lmr_2 \in V^*_{n+1}, ijr_1 \neq k \neq lmr_2, \alpha \in D \tag{19d\_01}$$

$$y^{\alpha}_{[ijr_1,k,lmr_2]} = 0 \quad if \ \ \tau'_{[ijr_1,k]} + \tau'_{[k,lmr_2]} > F \quad \forall \alpha \in D \tag{19d\_02}$$

$$p^{\alpha}_{[der_1,ijr_3]} \in \{0,1\} \quad \forall der_1 \in V^*_0, ijr_3 \in \{V^* | ijr_3 \neq der_1\}, \alpha \in D \tag{19e}$$

$$z^{\alpha}_{ijr}, z_{ijr} \in \{0,1\} \quad \forall ijr \in V^*_{0,n+1}, \alpha \in D \tag{19f}$$

$$Z^{\alpha,\beta}_{ijr} \in \{0,1\} \quad \forall ijr \in V^*_{0,n+1}, \alpha, \beta \in D, \tag{19g}$$

After optimization, we can calculate t'^{α}_{k}(arrival time to customer k by the α^{th} drone), $w^{\alpha}_{lmr_2}$ (waiting time of the truck for the α^{th} drone at intermediate point lmr_2), $w'^{\alpha}_{lmr_2}$(waiting time of the α^{th} drone for the truck at intermediate point lmr_2) as the followings.

$$t'^{\alpha}_{k} = t'^{\alpha}_{ijr_1} + s_L + \tau'_{[ijr_1,k]} \quad \forall y^{\alpha}_{[ijr_1,k,lmr_2]} = 1, \alpha \in D \tag{20a}$$

$$w^{\alpha}_{lmr_2} = max(t'^{\alpha}_{lmr_2} - s_R - t_{lm(r_2-1)} - \frac{1}{2} t_{ex} z_{lm(r_2-1)} - \tau_{[lm(r_2-1),lmr_2]} - \frac{1}{2} t_{ex} z_{lmr_2}, 0) \quad \forall y^{\alpha}_{[ijr_1,k,lmr_2]} = 1, \alpha \in D \tag{20b}$$

$$w'^{\alpha}_{lmr_2} = t'^{\alpha}_{lmr_2} - t'^{\alpha}_{k} - \tau'_{[k,lmr_2]} - s_R \quad \forall y^{\alpha}_{[ijr_1,k,lmr_2]} = 1, \alpha \in D \tag{20c}$$

According to the experience of essay 2 (Chapter 3) I could not find exact optimal solutions for large sized (more than six customers) TSPTWD problems within an hour. This generalized TSPTWmD would be even more time consuming. Then I decided not to solve optimal solutions but to develop effective metaheuristics to find good truck and drone solutions for large size problems (as many as 100 customers) in the following section.

Development of Metaheuristics

Based on the nature of the models and the heuristics in Chapter 3, some of them can still be used to solve multiple drone problems but some others cannot. The modified Savings Method

still can be used to find good truck solutions of TSP. The naïve method will not be used again since it does not use any improvement strategies and tests showed that for many instances it produced poor quality solutions than those from the Savings Method. SA_01 is a good metaheuristic as shown in Chapter 3, but it is not suitable for this multiple-drones model. SA_01 only uses near neighbors to find launching and receiving points. It cannot generate any overlapping situations as in Figure 16(f, g) which is one of the main reasons that multiple drones can help a driver save more time than operating only one drone. SA_02 from Chapter 3 can be extended to a multiple-drone version in this chapter. A new algorithm SA_03 will be developed to incorporate all the advantages of SA_01 and make it suitable for multiple drones.

Modified Savings Method for Traveling Salesman Problem with Time Windows (TSPTW)

It is the same as this section in Chapter 3.

Simulated Annealing (SA) framework for TSPTWD

Both SA_02, SA_03 are based on Algorithm 2 SA framework for TSPTWD in Chapter 3. We just need to add the number of drones into the algorithm.

SA_02, SA_03 return the same format two-dimension matrix (transformed_TD) to represent a truck and multiple drones solution as shown in Figure 16(h) which is transformed from the one-dimension truck solution. The returned solution of a truck and multiple drones is passed to an algorithm (cost_TmD) to calculate its corresponding cost as shown in Algorithm 6.

Algorithm 6. Calculate the cost a truck and multiple drones solution (cost_TmD)

Input:

transformed_TD #two-dimension matrix, e.g. Figure 16(h) to represent a truck and multiple drones solution

Parameters of the instance, e.g. τ, τ', V_d, s_L, s_R, t_{ex}, F, e_i, l_i, E, L (symbols are defined in Table 1)

Function cost_TmD:

Principle 1- If there are drones to be received at a point, then receive them at first and launch them if necessary. After that, launch the other drones.

Principle 2- If there are multiple drones to be received at a point, then receive the earliest arriving one at first and then launch it immediately if it is required. And then receive and launch the second arriving one and the others.

Principle 3- After all drones have been received and launched, that time is considered as the arrival time of the truck and drones system at a point.

Principle 4- The arrival time back to the depot is the delivery time of the truck and drones system.

Principle 5- *cost* = delivery time + ω_{TW}*violation of time windows + ω_F*violation of maximum flying time of a drone

Return *cost*

End Function

Simulated Annealing 02 (SA_02) for Multiple Drones

This algorithm (SA_02) is a generalized form of the SA_02 in Chapter 3 for one drone. SA_02 is distinguished from SA_03 since it uses some negative numbers, e.g. '-1's, '-2's, to indicate launching and receiving points. After negative numbers are inserted and adjusted properly, this adjusted string of solution can be input into Algorithm 2 SA framework to search for good solutions of a truck and multiple drones. The following paragraph will demonstrate how to insert negative numbers. When we need to transform a truck route into a truck and multiple drones solution, we will use this algorithm SA_02. SA_02 will utilize one of the improvement strategies, i.e. expanding and shrinking strategy to get good quality truck and multiple drones solutions. This process is briefly described in Algorithm 7 SA_02 multiple drones. A detailed example is illustrated after the paragraph about inserting negative numbers.

Figure 16(a) shows an original truck route within which 0, 6 represent the depot, and 1~5 are customers within which 1, 3, 4, 5 are drone eligible customers. To adjust it as a suitable string solution for SA_02, we need to insert $2*|V_d|$, i.e. 8, negative numbers randomly between the start and end depots, Figure 16(b). Assume we have two drones. Then they are represented by '-1's and '-2's. The number of negative numbers of each drone should be assigned approximately evenly. If there are greater than or equal to two '-1's following the depot '0', then one of those '-1's will be moved to the end of the solution string (after the depot 6) to be sure the depot at the end can be a receiving point and the same process for '-2' or other negative numbers. After that we have a well-adjusted truck route string solution (Figure 16(c)) to put into Algorithm 2 SA framework. The SA operator will operate on the customers and negative numbers between the start and end depot. When we need to calculate the truck cost of an adjusted form just described for SA_02, we need to remove all negative numbers temporarily.

Figure 16(c~i) will use an example to illustrate how to transfer a truck route of SA_02 into a truck and multiple drones solution. We iterate each point after 0 the depot in Figure 16(c). First, we should focus on '-1's for the first drone. The first '-1' after 0 indicates that depot 0 can be a launching point (Principle 1 of Algorithm 7). The two '-1's after 4 means customer 4 can be both a launching and a receiving point (Principle 2). Since there is a potential launching point 0 and two potential drone-visit customers 1, 3, we choose 1 randomly as the first-drone-visit customer out of 1, 3 (Principle 4). Then we can find a '-1's after 6 which means depot 6 can be a receiving point. There is one drone eligible customer 5 between 4 and 6. Then customer 5 will be visited by the first drone and 4, 6 are the launching and receiving points. The result is demonstrated in Figure 16(d). According to principle 5, we should remove all '-1's and first-drone-visit customers from the truck route, Figure 16(e). The launching and receiving points of

the first drone cannot be visited by the other drones. Then customer 4 cannot be a drone-visit customer and there is only customer 3 is drone eligible. Depot 0 and customer 4 can be launching and receiving points since there are '-2's following them. The '-2' after customer 2 has to be ignored because there are no drone eligible customers between 0 and 2. Between 0 and 4 there are two customers 2, 3, and 3 is the only one that can be visited by a drone. Then we choose customer 3 as a second-drone-visit customer as shown in Figure 16(f). Until now we only used customers as launching and receiving points. According to my new model, I should explore solutions by using intermediate points. Then the next step is to randomly shift the existing launching and receiving points to their nearby intermediate points but not further than the next customer. Suppose the algorithm randomly searching the nearby neighborhood of the launching point 0 and choosing an intermediate point 7 between depot 0 and customer 2. Then intermediate point 7 becomes the launching point of the first drone to visit customer 1. Similarly, an intermediate point 8 between customers 2, 4 is chosen as the updated receiving point. The next pair of launching and receiving points 4, 6 randomly shifted to intermediate points 8, 9. For the second drone, the launching point of customer 3 shifted from 0 to an intermediate point 7. The truck and multiple drones solution is shown in Figure 16(g). Next, this truck and multiple drones solution will be transformed into a two-dimension matrix as illustrated in Figure 16(h). The first row in that matrix indicates the truck route (7, 8, 9 are intermediate points between depot 0, customers 2 and customers 2, 4 and 4, 6). The '-1's in the second and fourth rows show the launching and receiving points of the drone. In Figure 16(h) the four '-1's indicate that points 7, 8, and 8, 9 are launching and receiving points. The third row shows that customers 1, 5 will be visited by the first drone after it is launched from intermediate points 7 and 8, respectively. Similarly, rows 5, 6, 7 show that the second drone will be launched at an intermediate point 7

then visit customer 3 and be received at customer 4. The last step before returning is to use the expanding and shrinking strategy. We will do shrinking at first and then expanding for the first drone and the second one. If the drone's flying time on 7-1-8 is shorter than the truck's traveling time on 7-2-8, then there is drone waiting time. We can use the shrinking strategy to reduce it. Suppose the algorithm shrinks the receiving point from intermediate point 8 to customer 2 such that the drone waiting time is reduced and there is not truck waiting time. After this shrinking operation, we cannot reduce the truck and drone delivery time directly, but we can reduce the probability of exceeding the maximum flying time and provide more opportunities for the following expanding strategy. If in this example the truck traveling time on 8-4-9 is shorter than the drone's flying time on 8-5-9, then there is truck waiting time which should be avoided as much as possible. Here we need to use the expanding strategy to try to expand the truck's route 8-4-9. The algorithm will search points out of the range of 8-4-9 as far as possible but cannot expand into other launching and receiving ranges of this drone. Suppose the algorithm chooses intermediate point 8 as the new launching point such that the truck traveling time on 2-8-4-9 is greater than or equal to the drone's flying time on 2-5-9. Then there will be no truck waiting time on this segment and the quality of the solution has been improved. Similarly, we shrank the receiving point of the second drone from customers 4 to 2. Finally, Figure 16(i) shows the matrix of the truck and drones solution (*transformed_TD*) after implementing the expanding and shrinking strategy which will be returned and this algorithm SA_02 is done.

Algorithm 7. Simulate Annealing 02 for multiple drones (SA_02. A detailed illustration example can be found in the main body)

Input:

String *sol* (initial solution of the truck route with negative numbers, e.g. Figure 16(c))

Parameters of the instance, e.g. τ, τ', V_d, s_L, s_R, t_{ex}, F (symbols are defined in Table 1)

Function SA_02:

Step 1. Generate a truck and multiple drones solution, e.g. Figure 16(f), based on the following principles.

Principle 1: If a customer or the depot is followed by a negative number (‘-1’, ‘-2’, etc.), then that customer or depot is eligible to be a launching or receiving point.

Principle 2: If a customer or the depot is followed by greater than or equal to two negative numbers, then that customer or depot is eligible to be a launching and a receiving point.

Principle 3: If there are no drone eligible customers between a pair of launching and receiving points, then ignore this potential receiving point. But the launching point is still active.

Principle 4: If there are multiple drone eligible customers between a pair of launching and receiving points, then one of them is randomly chosen as the drone-visit customer.

Principle 5: If there are multiple drones, then implement the above principles to the first drone. The first-drone-visit customers should be removed from the truck route and cannot be visited by other drones. The launching and receiving points of the first drone cannot be reused as launching or receiving points but cannot be visited by other drones. Then repeat principle 5 to the other drones.

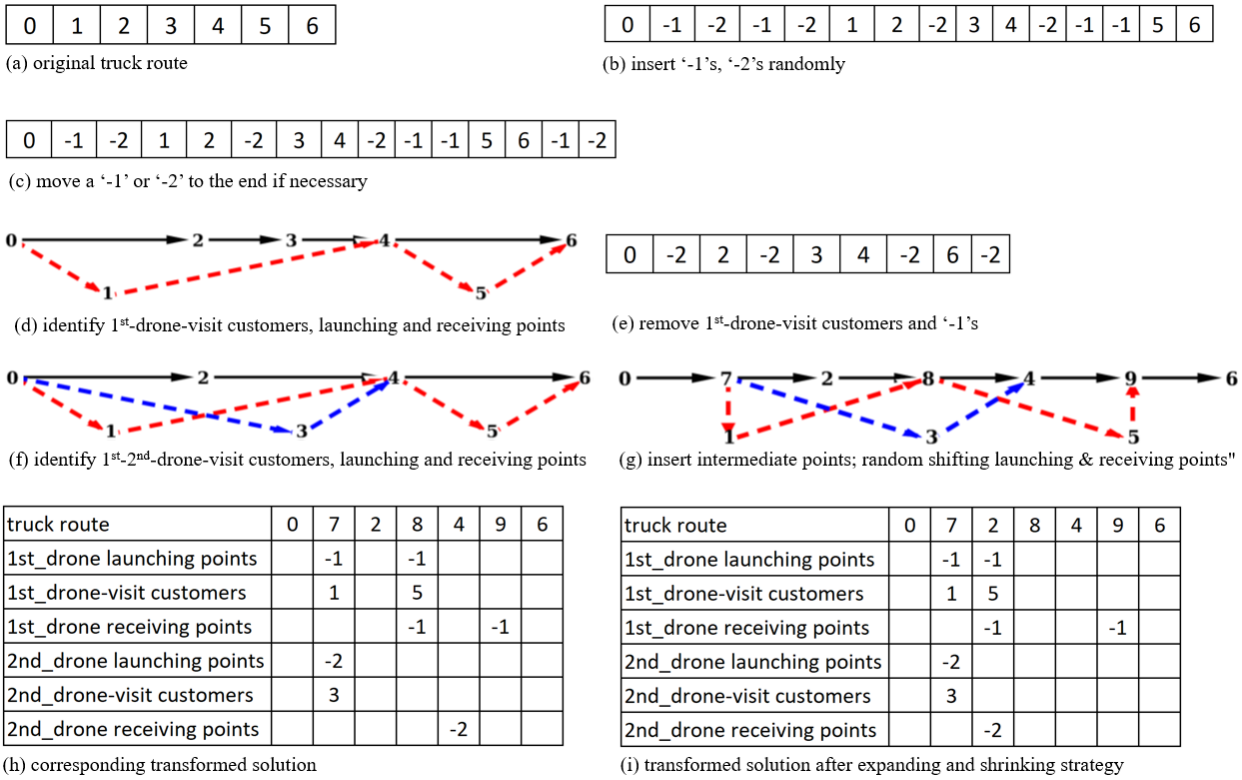
Step 2. Randomly shift the launching and receiving points to nearby intermediate points to explore more solutions (not further than one customer), e.g. Figure 16(g).

Step 3. Generate a two-dimension matrix *transformed_TD* to represent a truck and multiple drones solution, e.g. Figure 16(h).

Step 4. Implement the expanding and shrinking strategy to improve the quality of a solution further, e.g. Figure 16(i).

Return *transformed_TD*

End Function



truck route	0	7	2	8	4	9	6
1st_drone launching points		-1		-1			
1st_drone-visit customers		1		5			
1st_drone receiving points				-1		-1	
2nd_drone launching points		-2					
2nd_drone-visit customers		3					
2nd_drone receiving points					-2		

(h) corresponding transformed solution

truck route	0	7	2	8	4	9	6
1st_drone launching points		-1	-1				
1st_drone-visit customers		1	5				
1st_drone receiving points			-1			-1	
2nd_drone launching points		-2					
2nd_drone-visit customers		3					
2nd_drone receiving points			-2				

(i) transformed solution after expanding and shrinking strategy

Figure 16. An illustrative example of SA_02 (multiple drones)

Simulated Annealing 03 (SA_03) for Multiple Drones

Algorithm SA_01 in Chapter 3 is a good algorithm that utilizes all the improvement strategies and generates better results than SA_02. But SA_01 is not suitable for this multiple drones model since it cannot generate overlapped flights for different drones. This SA_03 will heritage all the advantages of SA_01 and incorporate multiple drones to find good solutions. Similar to SA_01, SA_03, it is also based on the Algorithm 2 SA framework. When we need to transform a truck route into a truck and multiple drones solution, we will use this algorithm. This process is briefly described in Algorithm 8 SA_03. A detailed example is illustrated in the following paragraph and Figure 17.

I will use the following example to demonstrate Algorithm SA_03. Suppose we have two drones in this example. Figure 17(a) shows part of an original tuck route that is passed into this

algorithm. 0 is the depot and 1 to 6 are customers. According to the picking and dropping strategy, all drone eligible customers will be picked up at first, i.e. assume customers 1,2,3,4, 5 in Figure 17 (b). An intermediate point 7 is on the arc (0, 6). Customers 1 to 5 are considered as a group of consecutive customers between the range 0-6. Each of such a group of customers must find their launching and receiving points within their ranges. Any of the two reasons would lead to a random dropping of one of the customers in a group, i.e. this group is infeasible or unpromising. The group of customers 1,2,3,4, 5 in Figure 17(b) is an example of an infeasible group since customer 1,3,5 and 2,4 will be visited by the first and second drone respectively but three points 0,7,6 in this range 0-6 cannot handle three customers of the first drone. Then we will drop one of the customers in this group (1,2,3,4,5) randomly. Suppose customer 4 is chosen and dropped back to the truck route. Then we can insert intermediate points 8, 9 into their corresponding arcs 0-4, 4-6, Figure 17(c). Now we have two groups of drone eligible customers, i.e. customers 1,2,3 between the range of 0-4 and 5 between the range of 4-6 which are all feasible since we can find enough launching and receiving points. Now we should check whether those groups are cost promising based on the idea of picking and dropping strategy, i.e. truck saving time is larger than drones' serving time. For example for the group of 1,2,3 in the range 0-4, we want to check whether $t_{01} + t_{12} + t_{23} + t_{34} - t_{04} > 3*(s_L + s_R + 2*t_{ex})$. If the answer is no, then this is not a promising picking we will drop back randomly one of customers 1,2,3. Suppose in this example it is promising for the two groups 1,2,3 and 5 and we will maintain to find their launching and receiving points. Since there are two drones, we assign customer 1,3 to the first drone and 2 for the second. In the second group, the only customer 5 is assigned to the first drone. Launching and receiving points for each drone and each group of customers will found randomly within their ranges. The result is shown in Figure 17(c). Next, this truck and drones

solution will be transformed into a two-dimension matrix as illustrated in Figure 17(d). The first row in that matrix indicates the truck route. The '-1's in the second and fourth rows show the launching and receiving points of the first drone. In Figure 17(d) the six '-1's indicate points 0,8,4 and 8,4,6 are launching and receiving points. The third row shows that customers 1,3,5 will be visited by the first drone. Similarly, the fifth to seventh rows show that the second drone is launched at point 8 to visit customer 2 and will be received at customer 4. The last step before returning is to use the expanding and shrinking strategy. We will do shrinking at first and then expanding for the first drone and the second one. If the drone's flying time on 4-5-6 is shorter than the truck's traveling time on 4-9-6, then there is drone waiting time. We can use the shrinking strategy to reduce it. Suppose the algorithm shrinks the launching point from customer 4 to intermediate point 8 such that the drone waiting time is reduced and there is not truck waiting time. Then we will use the expanding strategy. If in this example the truck traveling time on 8-4 is shorter than the drone's flying time on 8-3-4, then there is truck waiting time which should be avoided as much as possible. Here we need to use the expanding strategy to try to expand the truck's route 8-4. The algorithm will search points out of the range of 8-4 as far as possible, i.e. cannot expand into other launching and receiving ranges. Suppose the algorithm choose intermediate point 9 as the new receiving point such that the truck traveling time on 8-4-9 is greater than or equal to the drone's flying time on 8-3-9. Then there will be no truck waiting time on this segment and the quality of the solution has been improved. Similarly, the flight of the second drone is also expanded from 8-2-4 to 8-2-9. Finally, the matrix of the truck and drones solution (*transformed_TD*) after implementing the expanding and shrinking strategy will be returned and this algorithm is done.

Due to the nature of SA_02, it does not use the picking and dropping strategy. So, I am expecting that SA_03 will generate better solutions than SA_02. The following computational experiments will test it.

Algorithm 8. Simulate Annealing 03 for multiple drones (SA_03. A detailed illustration example can be found in the main body)

Input:

String *sol* (initial solution of the truck route of a traveling salesman problem, e.g. Figure 17 (a))

Parameters of the instance, e.g. τ, τ', V_d, s_L, s_R, t_{ex}, F (symbols are defined in Table 1)

Function SA_03:

Step 1. Pick up all customers in V_d out of the truck route as potential drone-visit customers, e.g. Figure 17 (a, b).

Step 2. Infeasible or unpromising customers will be dropped back, e.g. Figure 17 (b, c).

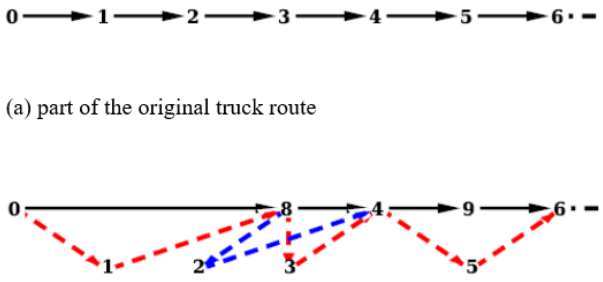
Step 3. Intermediate points are inserted and launching and receiving points of each drone-visit customers are chosen randomly within their ranges, e.g. Figure 17 (c).

Step 4. Generate a two-dimension matrix *transformed_TD* to represent a truck and multiple drones solution, e.g. Figure 17(d).

Step 5. Implement the expanding and shrinking strategy to improve the quality of a solution further, e.g. Figure 17 (e).

Return *transformed_TD*

End Function



(a) part of the original truck route

(b) will drop back infeasible or unpromising customers

(c) truck and multiple drones solution before expanding and shrinking strategy

truck route	0	8	4	9	6
1st_drone launching points	-1	-1	-1		
1st_drone-visit customers	1	3	5		
1st_drone receiving points		-1	-1		-1
2nd_drone launching points		-2			
2nd_drone-visit customers		2			
2nd_drone receiving points			-2		

(d) corresponding transformed solution

truck route	0	8	4	9	6
1st_drone launching points	-1	-1		-1	
1st_drone-visit customers	1	3		5	
1st_drone receiving points		-1		-1	-1
2nd_drone launching points		-2			
2nd_drone-visit customers		2			
2nd_drone receiving points				-2	

(e) transformed solution after expanding and shrinking strategy

Figure 17. An illustrative example of SA_03 (multiple drones)

Computational Experiments:

I arranged the following computational experiments to test the effectiveness of those several heuristics and how my new model can provide further benefits to the practice of a truck and multiple drones delivery mode.

Experimental Factors and Parameters

The same as this section in Chapter 3 and Table 6. The difference is that the number of drones is a new factor to test the effect of my new model on the delivery time savings.

Experimental Design

To compare those developed heuristics and effectiveness of my new model, I employed full factorial experiments with four factors, i.e. length of sides of a square area (three values),

number of customers (three values), number intermediate points on each arc (four values) and the number of drones (three values). I call each combination of the values of the four factors as a scenario. For each of such a scenario, I generated five instances randomly. Each of the instances would be solved by three algorithms, i.e. Savings Method, SA_02, and SA_03. Due to the randomness of the three algorithms, I have set each of them to take ten runs based on the same instance. First, the Savings Method started from the initial solution to run ten times. The average cost of the feasible solutions would be saved as the result of this algorithm of this instance. The best solution of the Savings Method would be the initial solution for all other algorithms. Each of them will take ten runs and save the average cost of feasible solutions as the result of this algorithm of this instance. The average cost of the five instances is the result of a scenario of an algorithm. Those results will be saved as the final results of these experiments. Then totally the experiments will take 3*3*4*3*5*3*10 = 16,200 runs.

I implemented the algorithms and experiments mainly in Julia (1.0.4) which is a high-performance language designed for scientific computation (Bezanson et al., 2017). The program was run on a desktop PC (Core i7 2.4GHz quad-core PC with 32GB of memory, Windows system).

Results

The results of the average delivery times of the computational experiments can be found in Appendix C. The impacts of the factors are reported as follows.

The comparison of algorithms is summarized in Table 10. The feasibility percentage shows how many feasible solutions were found within all the 450 runs. It shows that SA_03 has the highest percentage (99.8%) of feasibility and the Savings Method has the lowest one (98.9%). The average delivery time (of feasible solutions) demonstrates how good the quality is

for the solutions found by an algorithm. The result of the Savings Method for TSP can be considered as the benchmark. SA_03 found solutions with the lowest average delivery time (198.51 minutes) which dominates the results from SA_02 (217.28 minutes). It is obvious that SA_03 almost always finds feasible solutions and with higher quality. Then in the following part of this chapter, I will report data from this algorithm to show the impacts of factors.

Table 10. Comparison of algorithms

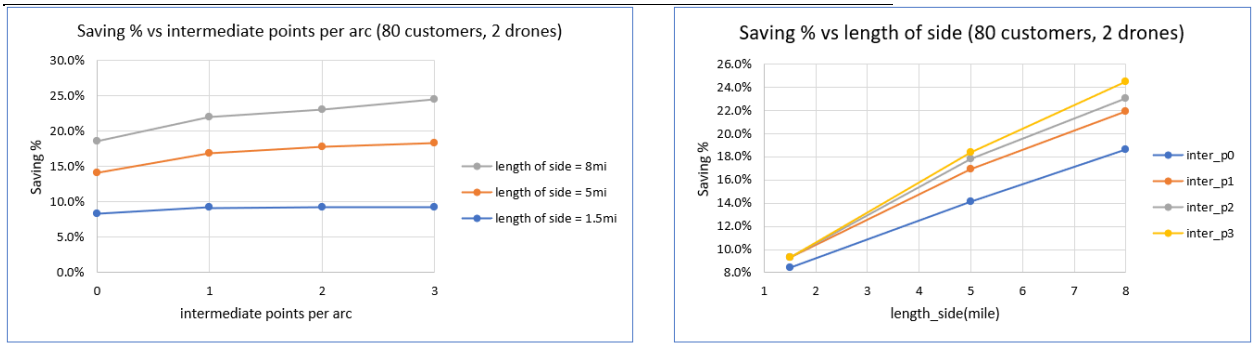
	Savings_Method	SA_02	SA_03
feasibility %	98.9%	99.0%	99.8%
average delivery time (minute)	246.55	217.28	198.51

The data of the impacts of the number of intermediate points on each arc is shown in Table 11. For a given number of customers and drones, normally the delivery time saving percentage will increase when more intermediate points are provided. We can also observe that for a given number of customers, an increased size of area will lead to more savings. It implies that lower customer density provides more opportunities for delivery time saving by deploying drones. Figure 18(a) shows a chart of one of the typical situations.

Table 11 also shows the impact of the length of side (or size of area) on delivery time saving percentage. For a given number of customers and drones, if we increase the length of side (or size of area) then we will find more delivery time savings. That is consistent with my analysis of the CG strategy in Chapter 3. This trend is also demonstrated in Figure 18(b). We can also observe that for given customer density and number of drones, more intermediate points on an arc will lead to more savings and with diminishing margins. It implies that too many intermediate points may not continuously bring us more delivery time savings.

Table 11. Delivery time saving % with one truck two drone, SA_03 metaheuristic

#_cus	length_side(mile)	inter_p0	inter_p1	inter_p2	inter_p3
60	1.5	5.9%	6.5%	6.8%	6.8%
60	5	18.0%	21.3%	22.3%	22.5%
60	8	24.6%	27.3%	28.4%	28.9%
80	1.5	8.4%	9.3%	9.3%	9.3%
80	5	14.1%	16.9%	17.8%	18.4%
80	8	18.6%	22.0%	23.1%	24.5%
100	1.5	6.2%	6.7%	6.7%	6.7%
100	5	12.6%	16.4%	17.5%	18.3%
100	8	24.6%	28.2%	30.3%	30.6%



(a) (b)

Figure 18. Delivery time saving % vs # of intermediate points or length of side, 2 drones

The data of the impacts of the number of customers is also contained in Table 11. For a given area and number of drones, increasing numbers of customers did not lead to a monotonic trend of saving percentages. That was a little bit confusing to me. A typical chart is shown in Figure 19(a). After analyzing the nature of problems, I think the reason is that random instances were generated with different numbers of customers, random time windows, random locations. Those would bring unpredictable randomness to the problems. Literature also supports my analysis that sometimes algorithms generate unexpected solutions (with longer traveling time) due to the constraints of some customers' time windows (Holland et al., 2017). Then I run the experiment again with no time windows. Some typical results are shown in Figure 19(b) with a monotonic decreasing trend of delivery time saving percentages versus the number of customers

in a given area and number of drones. It is generally consistent with my analysis that higher customer density leads to lower savings. Please note that the saving percentages are reduced when time windows are removed. The possible reason is that time windows make it more difficult to find good solutions, e.g. UPS stated that their truck may visit an area twice due to time windows (Holland et al., 2017). Such that there might be more chances to save time when there are time windows and vice versa.

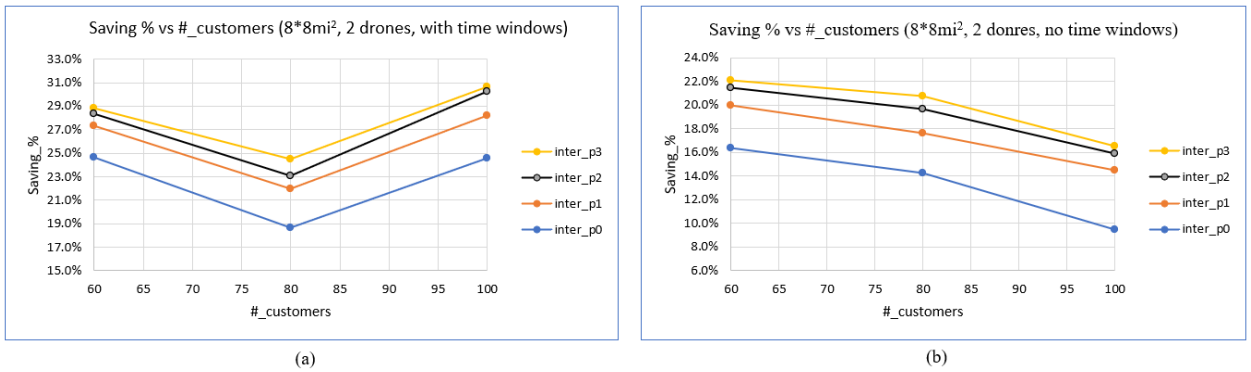
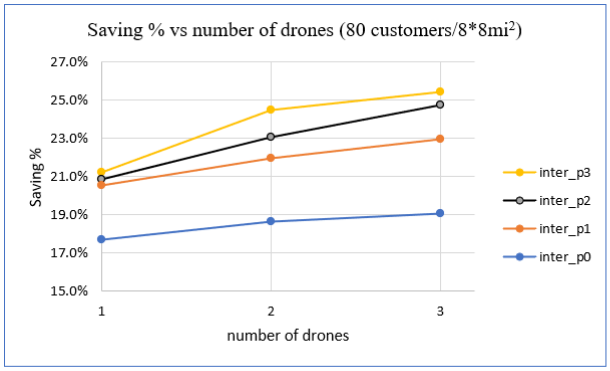


Figure 19. Typical charts of delivery time saving % vs number of customers

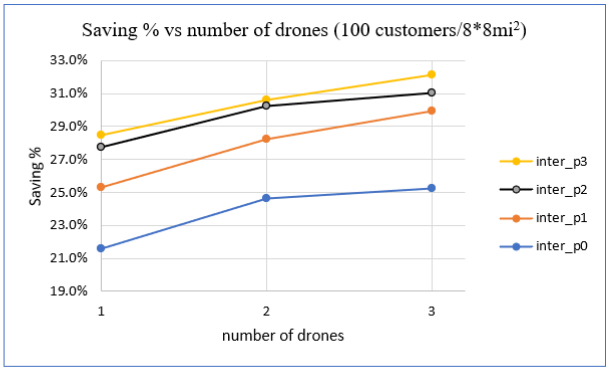
The data of the impacts of number of drones is shown in Table 11. For a given customer density, normally the delivery time saving percentage will increase when more drones are provided. Figure 20 shows the charts of two typical situations. This figure also infers that the marginal saving percentages are decreasing when there is a larger number of drones. It implies that too large number of drones may not bring us a large percentage of delivery time savings. A possible reason is that too many drones require the truck driver to collaborate them and must wait for the last drone being received at a point then he or she can continue to visit the next customer. Those collaborations and waiting may increase the total delivery time which diminishes the savings from a larger number of drones.

Table 12. Delivery time saving %, area=8*8mi^2, SA_03 metaheuristic

length_side(mile)	#_cus	#_drones	inter_p0	inter_p1	inter_p2	inter_p3
8	60	1	20.7%	23.7%	25.2%	25.8%
8	60	2	24.6%	27.3%	28.4%	28.9%
8	60	3	26.0%	28.8%	29.6%	29.8%
8	80	1	17.7%	20.5%	20.9%	21.2%
8	80	2	18.6%	22.0%	23.1%	24.5%
8	80	3	19.0%	23.0%	24.7%	25.5%
8	100	1	21.6%	25.3%	27.8%	28.5%
8	100	2	24.6%	28.2%	30.3%	30.6%
8	100	3	25.2%	29.9%	31.1%	32.1%



(a)



(b)

Figure 20. Typical charts of delivery time saving % vs number of drones

Conclusion

In Chapter 4 I developed a new mathematical model of TSPTWmD and two metaheuristics (SA_02, SA_03) to solve TSPTWmD problems having as many as 100 customers which is similar to the size of problems in the real world.

A series of improvement strategies had been developed to improve the quality of solutions. Those strategies were developed based on the specific nature of this novel problem TSPTWmD. When those strategies were implemented the metaheuristics could generate much higher quality solutions.

Computational experiments were designed and implemented to test the developed metaheuristics and the impact of factors. The results showed that SA_03 is outperformed to SA_02. The idea of my research to provide multiple drones did generate more delivery time savings than only employing one drone. The results show that the delivery time could be further reduced as large as more than 5% and the actual time saving could be as large as 32.1%. And intermediate points also help to increase those savings. UPS says for its 55,000 drivers even one mile reduction per driver per day will lead to about $50 million annual saving (UPS, 2016), i.e. about $2.5 per mile. The typical operating cost (including purchasing, electricity, maintenance, monitoring costs) of a drone is about $0.044 per mile (Keeney, 2015). Based on the experiments, it may be assumed to save one mile for the truck the drone flies two miles. Then the one-mile cost reduction with a drone is $2.4 or $48 million per year. One mile is about 1% of a typical route of UPS. Then the 32.1% saving from my model is approximately equivalent to an annual saving of $1.54 billion. This huge saving would be attractive to practitioners and customers.

As I analyzed in the part of the center of gravity (CG) improvement strategy, lower customer density may provide more opportunities for delivery time saving. The impact of the size of areas supports this analysis. But the impact of the number of customers is not so predictable. The reason might be the randomness of time windows. When all time windows were removed the results are consistent with what I analyzed.

The computational results also showed that too many intermediate points on an arc or too many drones may not provide directly proportional time savings. That means a limited number of intermediate points on arcs or drones would be enough to provide the most savings. It is consistent with the situation in the real world that due to some constraints, e.g. gas stations, crossings, bus stops, etc., we can only provide a limited number of intermediate points on arcs.

And the too large number of drones could increase the complexity of the operation and diminish further time savings.

CHAPTER 5. GENERAL CONCLUSION AND FUTURE RESEARCH

In this dissertation, I study a relatively novel variant form of traveling salesman problem (TSP), i.e. traveling salesman problem with time windows and a drone (TSPTWD) or multiple drones (TSPTWmD) with intermediate points. Mathematical models and metaheuristics were successfully developed, and numerical experiments showed promising savings of delivery time based on existing modes.

In Chapter 2 (essay 1) I developed a new model to solve a novel variant of TSP, i.e. traveling salesman problem with time windows and a drone (TSPTWD) with intermediate points.

Computational experiments have been implemented to test my new model and several LP heuristics. The results of small size problems (less than seven customers) showed that my new model can save delivery time as large as 42.8% than the traditional pure truck TSP model. Comparing with the existing model of a truck and a drone operated only on customers' sites, my new model further increased the saving as large as 2.85%. And that saving could be even larger if we implement my model to larger size problems (with more customers).

Due to the NP-hardness, only small size problems (less than seven customers) can be solved exactly in a practical period (less than one hour). Some LP heuristics can provide relatively good solutions within a shorter time. But that computing time will also increase too large to accept. Then I developed other heuristics or metaheuristics to solve large size problems within an acceptable period.

In Chapter 3 (essay 2) I developed multiple heuristics and metaheuristics to solve TSPTWD problems as large as 100 customers which is like the size of problems in the real world.

The problem of poor-quality solutions had been met and a series of improvement strategies were initiated. Those strategies were developed based on the specific nature of this novel problem TSPTWD. When those strategies were implemented the metaheuristics could generate solutions with much higher quality.

Numerical experiments were designed and implemented to test the developed metaheuristics and the impact of factors. The results showed that the algorithm SA_01 outperformed SA_02 and the naïve method of TSPTWD. The idea of my research to provide intermediate points on arc did generate more delivery time savings than only operating the drone on customers' sites. The results showed that the delivery time could be further reduced as much as 4% and the actual time saving could be as much as 26%.

As I analyzed in the part of the center of gravity (CG) improvement strategy, lower customer density may provide more opportunities for delivery time saving. The computational results also showed that too many intermediate points on an arc do not provide directly proportional time savings. That means a limited number of intermediate points on arcs would be enough to provide the most savings. It is consistent with the situation in the real world that due to some constraints, e.g. gas stations, crossings, bus stops, etc., we can only provide a limited number of intermediate points on arcs.

In Chapter 4 (essay 3) I developed a new mathematical model of TSPTWmD and two metaheuristics (SA_02, SA_03) to solve TSPTWmD problems as large as 100 customers which is like the size of problems in the real world.

A series of improvement strategies had been implemented to improve the quality of solutions. Those strategies were developed based on the specific nature of this novel problem TSPTWmD. When those strategies were utilized the metaheuristics generated much higher quality solutions.

Computational experiments were designed and implemented to test the developed metaheuristics and the impact of factors. The results showed that the algorithm SA_03 outperformed SA_02. The idea of my research to provide multiple drones did generate more delivery time savings than only employing one drone. The results show that the delivery time could be further reduced as large as more than 5% and the actual time saving could be as large as 32.1%. And intermediate points also help to increase those savings. According to literature (UPS, 2016, Keeney, 2015) 32.1% is about $1.54 billion annual saving for a parcel delivery firm like UPS.

As similar to the results in Chapter 3, lower customer density provides more opportunities for delivery time saving. The computational results also showed that too many intermediate points on an arc or too many drones may not provide directly proportional time savings. That means a limited number of intermediate points on arcs or drones would be enough to provide the most savings. It is consistent with the situation in the real world that due to some constraints, e.g. gas stations, crossings, bus stops, etc., we can only provide a limited number of intermediate points on arcs. And a too large number of drones could increase the complexity of the operation and diminish further time savings.

There are also some limitations to my studies. First, although most parameters of the computational experiments are based on actual data, hypothetical instances were generated to be tested. Perhaps some real addresses can be utilized in the future to get more insights into the

topic. Second, my algorithms can solve problems as large as 100 customers which are almost the same as some real problems. But practical reports also show that sometimes a driver may visit as many as 200 customers per day. That means in the future more efficient algorithms are required to find better solutions with even shorter computing time.

There are also some potential future research directions for this new mode of parcel delivery. The delivery mode may be generalized as a drone does not only visit one customer but multiple customers in one flight to further save the delivery time. How this new mode affects pollution and operating cost are also interesting topics for researchers and practitioners. The drones assisted delivery model is still a relatively new approach to transportation. Research and practice on this topic mush bring more benefits to maintain sustainable logistic activities.

REFERENCES

AGATZ, N., BOUMAN, P. & SCHMIDT, M. J. T. S. 2018. Optimization approaches for the traveling salesman problem with drone.

APPLEGATE, D. L. 2006. *The traveling salesman problem: a computational study*, Princeton university press.

BALAS, E. & CHRISTOFIDES, N. 1981. A restricted Lagrangean approach to the traveling salesman problem. *Mathematical Programming,* 21**,** 19-46.

BALLOU, R. H. 2004. Business logistics/supply chain management 5th edn. Pearson Education, Inc, Upper Saddle River, New Jersey.

BALLOU, R. H. & AGARWAL, Y. K. 1988. A Performance Comparison Of Several Popular Alglorithms For. *Journal of Business Logistics,* 9**,** 51.

BARNITT, R. 2011. *FedEx Express Gasoline Hybrid Electric Delivery Truck Evaluation: 12-Month Report* [Online]. @nrel. Available: https://www.nrel.gov/docs/fy11osti/48896.pdf [Accessed September 20, 2017].

BEZANSON, J., EDELMAN, A., KARPINSKI, S. & SHAH, V. B. 2017. Julia: A fresh approach to numerical computing. *SIAM review,* 59**,** 65-98.

BUTLIN, J. 1989. Our common future. By World commission on environment and development.(London, Oxford University Press, 1987, pp. 383£ 5.95.). Wiley Online Library.

CARPANETO, G. & TOTH, P. 1980. Some new branching and bounding criteria for the asymmetric travelling salesman problem. *Management Science,* 26**,** 736-743.

CASTILLO, V. E., BELL, J. E., ROSE, W. J. & RODRIGUES, A. M. J. J. O. B. L. 2018. Crowdsourcing Last Mile Delivery: Strategic Implications and Future Research Directions. 39**,** 7-25.

CERNY, V. 1985. Thermo dynamical approach to the traveling salesman problem, An efficient.

CHRISTOFIDES, N. 1970. The shortest Hamiltonian chain of a graph. *SIAM Journal on Applied Mathematics,* 19**,** 689-696.

CLARKE, G. & WRIGHT, J. W. J. O. R. 1964. Scheduling of vehicles from a central depot to a number of delivery points. 12**,** 568-581.

DON, M. 2013. *UPS Gettysburg Orion* [Online]. Available: https://www.youtube.com/watch?v=CsJRsToDI8w [Accessed 01/28/2020 2020].

DONATH, M. 2014. World cities, home to most people, to add 2.5 billion more by 2050: U.N. *Reuters*.

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**,** 70-85.

DURAND, B., MAHJOUB, S. & SENKEL, M.-P. Delivering to Urban Online Shoppers: The Gains from "Last-Mile" Pooling. Supply Chain Forum: An International Journal, 2013. Taylor & Francis, 22-31.

ERDOĞAN, S. & MILLER-HOOKS, E. 2012. A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review,* 48**,** 100-114.

FIGLIOZZI, M. A. 2011. The impacts of congestion on time-definitive urban freight distribution networks CO2 emission levels: Results from a case study in Portland, Oregon. *Transportation Research Part C: Emerging Technologies,* 19**,** 766-778.

FU, J. & JENELIUS, E. 2017. Transport efficiency of off-peak urban goods deliveries: a Stockholm pilot study. *Submitted to Case Studies on Transport Policy*.

GENDREAU, M. & POTVIN, J.-Y. 2010. *Handbook of metaheuristics*, Springer.

GEVAERS, R., VAN DE VOORDE, E. & VANELSLANDER, T. 2011. Characteristics and typology of last-mile logistics from an innovation perspective in an urban context. *City Distribution and Urban Freight Transport: Multiple Perspectives, Edward Elgar Publishing***,** 56-71.

GLOVER, F. 1977. Heuristics for integer programming using surrogate constraints. *Decision sciences,* 8**,** 156-166.

GOODCHILD, A., TOY, J. J. T. R. P. D. T. & ENVIRONMENT 2018. Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing CO2 emissions in the delivery service industry. 61**,** 58-67.

GUGLIELMO, C. 2017. *Turns Out Amazon, Touting Drone Delivery, Does Sell Lots of Products That Weigh Less Than 5 Pounds* [Online]. @forbes. Available: https://www.forbes.com/sites/connieguglielmo/2013/12/02/turns-out-amazon-touting-drone-delivery-does-sell-lots-of-products-that-weigh-less-than-5-pounds/ [Accessed Sep. 29th 2017].

HELD, M. & KARP, R. M. 1971. The traveling-salesman problem and minimum spanning trees: Part II. *Mathematical programming,* 1**,** 6-25.

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

HERN, A. 2014. DHL launches first commercial drone 'parcelcopter' delivery service. *the Guardian*.

HOLGUÍN-VERAS, J., OZBAY, K., KORNHAUSER, A., BROM, M., IYER, S., YUSHIMITO, W., UKKUSURI, S., ALLEN, B. & SILAS, M. 2011. Overall impacts of off-hour delivery programs in New York City Metropolitan Area. *Transportation Research Record: Journal of the Transportation Research Board***,** 68-76.

HOLLAND, C., LEVIS, J., NUGGEHALLI, R., SANTILLI, B. & WINTERS, J. J. I. 2017. UPS optimizes delivery routes. 47**,** 8-23.

KARP, R. M. 1972. Reducibility among combinatorial problems. *Complexity of computer computations.* Springer.

KEENEY, T. 2015. *How Can Amazon Charge $1 for Drone Delivery?* [Online]. ARK invest. Available: https://ark-invest.com/research/drone-delivery-amazon [Accessed 01/29/2019 2019].

KIRKPATRICK, S., GELATT, C. D. & VECCHI, M. P. 1983. Optimization by simulated annealing. *science,* 220**,** 671-680.

LAMMERT, M. & WALKOWICZ, K. 2012. Eighteen-month final evaluation of UPS second generation diesel hybrid-electric delivery vans. National Renewable Energy Lab.(NREL), Golden, CO (United States).

LAPORTE, G. 1992. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research,* 59**,** 231-247.

LAPORTE, G. 2009. Fifty years of vehicle routing. *Transportation Science,* 43**,** 408-416.

LEE, H. L. & WHANG, S. 2001. Winning the last mile of e-commerce. *MIT Sloan management review,* 42**,** 54-62.

LIN, S. & KERNIGHAN, B. W. 1973. An effective heuristic algorithm for the traveling-salesman problem. *Operations research,* 21**,** 498-516.

MARTIN, C. 2020. *Robotic Deliveries To Hit 20 Billion By 2030: Study* [Online]. mediapost. Available: https://www.mediapost.com/publications/article/348607/robotic-deliveries-to-hit-20-billion-by-2030-stud.html [Accessed 06/30 2020].

MCNABB, M. 2016. *Amazon's New Patent: Drones Could "Hitch Rides" on Trucks - DRONELIFE* [Online]. Available: https://dronelife.com/2016/09/12/amazons-new-patent-drones-hitch-rides-trucks/ [Accessed 05/29 2020].

MILLER, D. L. & PEKNY, J. F. 1991. Exact solution of large asymmetric traveling salesman problems. *Science,* 251**,** 754-761.

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.

OECD 2015. *Data-Driven Innovation: Big Data for Growth and Well-Being*, OECD Publishing.

PEREZ, S. & KOLODNY, L. 2017. *UPS tests show delivery drones still need work* [Online]. @techcrunch. Available: http://social.techcrunch.com/2017/02/21/ups-tests-show-delivery-drones-still-need-work/ [Accessed September 20, 2017].

PRINCETON. 2020. *Cities and the Future Metropolis | School of Engineering and Applied Science* [Online]. Available: https://engineering.princeton.edu/research/cities-and-the-future-metropolis [Accessed 06/30 2020].

REINELT, G. 1994. *The traveling salesman: computational solutions for TSP applications*, Springer-Verlag.

ROSE, W. J., BELL, J. E., AUTRY, C. W. & CHERRY, C. R. 2017a. Urban Logistics: Establishing Key Concepts and Building a Conceptual Framework for Future Research. *Transportation Journal,* 56**,** 357-394.

ROSE, W. J., BELL, J. E., AUTRY, C. W. & CHERRY, C. R. J. T. J. 2017b. Urban Logistics: Establishing Key Concepts and Building a Conceptual Framework for Future Research. 56**,** 357-394.

SAVELSBERGH, M. & VAN WOENSEL, T. J. T. S. 2016. 50th anniversary invited article—city logistics: Challenges and opportunities. 50**,** 579-590.

SOLOMON, M. M. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research,* 35**,** 254-265.

TOTH, P. & VIGO, D. 2002. Vehicle Routing Problem SIAM Monographs on Discrete Mathematics and Applications, Vol. 9. *SIAM: Philadelphia, PA*.

TOTH, P. & VIGO, D. 2003. The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing,* 15**,** 333-346.

TRANSPORT TOPICS 2017. UPS tests drone that uses delivery truck as its home base. *Transport Topics.*

UPS. 2016. *ORION: The algorithm proving that left isn't right | UPS - United States* [Online]. Available: https://www.ups.com/us/en/services/knowledge-center/article.page?kid=aa3710c2 [Accessed 06/22 2020].

UPS 2017. UPS Tests Residential Delivery Via Drone.

UPS. 2019. *UPS Flight Forward Attains FAA's First Full Approval For Drone Airline* [Online]. UPS Pressroom. Available: https://pressroom.ups.com/pressroom/ContentDetailsViewer.page?ConceptType=PressReleases&id=1569933965476-404 [Accessed 05/29 2020].

WANG, X., POIKONEN, S. & GOLDEN, B. J. O. L. 2017. The vehicle routing problem with drones: several worst-case results. 11**,** 679-697.

WEISE, E. 2016. Amazon delivered its first customer package by drone. *USA TODAY*.

WEISE, E. 2017. *UPS Tests Truck-Launched Drone Delivery System in Florida | BCNN1 - Black Christian News Network* [Online]. USA Today. Available: https://blackchristiannews.com/2017/02/ups-tests-truck-launched-drone-delivery-system-in-florida/ [Accessed 05/29 2020].

WOLSEY, L. A. 1998. *Integer programming*, Wiley.

XIAO, Y., ZHAO, Q., KAKU, I. & XU, Y. 2012. Development of a fuel consumption optimization model for the capacitated vehicle routing problem. *Computers & Operations Research,* 39**,** 1419-1431.

APPENDIX A. THE EXTRA STOPPING TIME (t_{ex}) OF A TRUCK

The extra stopping time in the truck's out-of-route period will be derived in this section. Fig. A1 shows a segment of an arc on which a truck travels from left to right. Normally the truck will travel from A to B with a speed v, traveling time is t_{AB}. When the truck will go out of the route from A to C to stop and operate the drone (either launching or receiving) then starts again and go back to the arc at B. Assume it is a symmetric process, i.e. decelerating on AC and later accelerating on CB with time $t_{AC} = t_{CB}$, $v_A = v_B = v$, $v_C = 0$, the absolute value of deceleration and acceleration is a. The extra stopping time on this out-of-route period $t_{ex} = t_{AC} + t_{CB} - t_{AB}$.

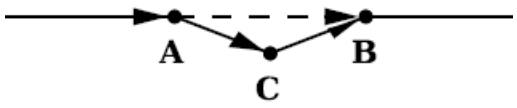


Figure 21. A truck travels out-of-route at point C to operate the drone

According to kinematics of physics, we have

$$v = 0 + a * t_{CB} \quad then \;\; t_{CB} = \frac{v}{a}$$

Let s_{AB}, s_{AC}, s_{CB} denote the distance between AB, AB, and CB. Note that Fig. A1 is an exaggerated one within which the perpendicular distance from C to AB is as short as 3~5meters (UPS (2017)). Comparing with the length of AB (more than 100 meters) that short distance can be ignored. So, we have

$$s_{AB} \cong s_{AC} + s_{CB} = 2 * s_{CB} = 2 * (average\ v_{CB}) * t_{CB} = 2 * \frac{0 + v}{2} * t_{CB} = v * \frac{v}{a} = v^2/a$$

$$Then\ t_{AB} = \frac{s_{AB}}{v} = \frac{\frac{v^2}{a}}{v} = v/a$$

We have the extra stopping time on this out-of-route period

$$t_{ex} = t_{AC} + t_{CB} - t_{AB} = \frac{v}{a} + \frac{v}{a} - \frac{v}{a} = v/a$$

According to Don (2013), the average speed v is about 12 mph (serving time included), acceleration a≅0.61m/s^2 (Lammert and Walkowicz, 2012). If we use v = 12mph, then the extra stopping time will be

$$t_{ex} = \frac{v}{a} = \frac{12mph}{0.61m/_{s^2}} = 8.79423\ seconds \cong 0.1466\ minute$$

In this study, I will use 0.1466 minute as the standard extra stopping time for each out-of-route. The formula of t_{ex} can give specific values if speed v is changed under different situations. Furthermore, we can calculate the total deceleration and acceleration distance

$$s_{AC} + s_{CB} = \frac{v^2}{a} = \frac{(12mph)^2}{0.61m/_{s^2}} \cong 47.18\ meters$$

It is much larger than 3~5 meters. That means our approximation is quite accurate.

APPENDIX B. THE LINEARIZED FORM OF CONSTRAINT (17f)

Constraint (17f) is to maintain an appropriate relationship of arrival times of each pair of drones at a given point. Since it is nonlinear, this appendix shows its equivalent linear form.

$$t'^{\beta}_{lmr_2} - t'^{\alpha}_{lmr_2} \geq s_R - \left(3 - Z^{\alpha,\beta}_{lmr_2} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\beta}_{[ijr_1,k,lmr_2]}\right)(L + s_R)$$
$$\forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_01a)$$

$$t'^{\alpha}_{lmr_2} - t'^{\beta}_{lmr_2} \geq s_R - \left(2 + Z^{\alpha,\beta}_{lmr_2} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\beta}_{[ijr_1,k,lmr_2]}\right)(L + s_R)$$
$$\forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_01b)$$

$$t'^{\beta}_{lmr_2} - t'^{\alpha}_{lmr_2} \geq s_L - \left(3 - Z^{\alpha,\beta}_{lmr_2} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\beta}_{[lmr_2,n,pqr_3]}\right) \cdot (L + s_L)$$
$$\forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_02a)$$

$$t'^{\alpha}_{lmr_2} - t'^{\beta}_{lmr_2} \geq s_L - \left(2 + Z^{\alpha,\beta}_{lmr_2} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\beta}_{[lmr_2,n,pqr_3]}\right) \cdot (L + s_L)$$
$$\forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_02b)$$

$$t'^{\beta}_{lmr_2} - t'^{\alpha}_{lmr_2} \geq s_L + s_R - \left(3 - Z^{\alpha,\beta}_{lmr_2} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\beta}_{[ijr_1,k,lmr_2]}\right) \cdot (L + s_L + s_R)$$
$$\forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_03a)$$

$$t'^{\alpha}_{lmr_2} - t'^{\beta}_{lmr_2} \geq -\left(2 + Z^{\alpha,\beta}_{lmr_2} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\beta}_{[ijr_1,k,lmr_2]}\right) \cdot L$$
$$\forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_03b)$$

$$t'^{\beta}_{lmr_2} - t'^{\alpha}_{lmr_2} \geq -\left(3 - Z^{\alpha,\beta}_{lmr_2} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\beta}_{[lmr_2,n,pqr_3]}\right) \cdot L$$
$$\forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_04a)$$

$$t'^{\alpha}_{lmr_2} - t'^{\beta}_{lmr_2} \geq s_L + s_R - \left(2 + Z^{\alpha,\beta}_{lmr_2} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\beta}_{[lmr_2,n,pqr_3]}\right) \cdot (L + s_L + s_R)$$
$$\forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_04b)$$

$$t'^{\beta}_{lmr_2} - t'^{\alpha}_{lmr_2} \geq s_L + s_R - (4 - Z^{\alpha,\beta}_{lmr_2} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]}$$
$$- \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\beta}_{[ijr_1,k,lmr_2]})(L + s_L + s_R) \qquad \forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_05a)$$

$$t'^{\alpha}_{lmr_2} - t'^{\beta}_{lmr_2} \geq s_R - (3 + Z^{\alpha,\beta}_{lmr_2} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\beta}_{[ijr_1,k,lmr_2]} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]}$$
$$- \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]})(L + s_R) \qquad \forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_05b)$$

$$t'^{\beta}_{lmr_2} - t'^{\alpha}_{lmr_2} \geq s_L - (4 - Z^{\alpha,\beta}_{lmr_2} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]}$$
$$- \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\beta}_{[lmr_2,n,pqr_3]})(L + s_L) \qquad \forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_06a)$$

$$t'^{\alpha}_{lmr_2} - t'^{\beta}_{lmr_2} \geq s_L + s_R - (3 + Z^{\alpha,\beta}_{lmr_2} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\beta}_{[lmr_2,n,pqr_3]} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]}$$
$$- \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]})(L + s_L + s_R) \qquad \forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_06b)$$

$$t'^{\beta}_{lmr_2} - t'^{\alpha}_{lmr_2} \geq s_R - (4 - Z^{\alpha,\beta}_{lmr_2} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\beta}_{[ijr_1,k,lmr_2]}$$
$$- \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\beta}_{[lmr_2,n,pqr_3]})(L + s_R) \qquad \forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_07a)$$

$$t'^{\alpha}_{lmr_2} - t'^{\beta}_{lmr_2} \geq s_L + s_R - (3 + Z^{\alpha,\beta}_{lmr_2} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\beta}_{[ijr_1,k,lmr_2]} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\beta}_{[lmr_2,n,pqr_3]}$$
$$- \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]})(L + s_L + s_R) \qquad \forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_07b)$$

$$t'^{\beta}_{lmr_2} - t'^{\alpha}_{lmr_2} \geq s_L + s_R - (4 - Z^{\alpha,\beta}_{lmr_2} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\beta}_{[ijr_1,k,lmr_2]}$$
$$- \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\beta}_{[lmr_2,n,pqr_3]})(L + s_L + s_R) \qquad \forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_08a)$$

$$t'^{\alpha}_{lmr_2} - t'^{\beta}_{lmr_2} \geq s_L - (3 + Z^{\alpha,\beta}_{lmr_2} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\beta}_{[ijr_1,k,lmr_2]} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\beta}_{[lmr_2,n,pqr_3]}$$
$$- \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]})(L + s_L) \qquad \forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_08b)$$

$$t'^{\beta}_{lmr_2} - t'^{\alpha}_{lmr_2} \geq s_L + s_R - (5 - Z^{\alpha,\beta}_{lmr_2} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]}$$
$$- \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\beta}_{[ijr_1,k,lmr_2]} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\beta}_{[lmr_2,n,pqr_3]})(L + s_L + s_R)$$
$$\forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_09a)$$

$$t'^{\alpha}_{lmr_2} - t'^{\beta}_{lmr_2} \geq s_L + s_R - (4 + Z^{\alpha,\beta}_{lmr_2} - \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\beta}_{[ijr_1,k,lmr_2]} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\beta}_{[lmr_2,n,pqr_3]}$$
$$- \sum_{ijr_1 \epsilon V_0^*} \sum_{k \epsilon V_d} y^{\alpha}_{[ijr_1,k,lmr_2]} - \sum_{n \epsilon V_d} \sum_{pqr_3 \epsilon V_{n+1}^*} y^{\alpha}_{[lmr_2,n,pqr_3]})(L + s_L + s_R)$$
$$\forall lmr_2 \epsilon V_{n+1}^*, \alpha, \beta \in D, \alpha \neq \beta \qquad (B\_09b)$$

For example, when a pair of drones are only received and without launching on a point then the time interval between their arrival times should be at least the receiving time s_R no matter which one is received earlier than the other one. Those relationships are reflected in (B_01a) and (B_01b). Similarly, (B_02a) and (B_02b) are for two drones only launched without receiving on a point, (B_05a) and (B_05b) are for one drone received and launched and then another one is only received. The other constraints in (B) can be explained in a similar logic.

APPENDIX C. AVERAGE RESULTS OF EXPERIMENTS IN CHAPTERS 3, 4

Part I: SA_01

length of side (mile)	1.5	1.5	1.5	5	5	5	8	8	8
# cus	60	80	100	60	80	100	60	80	100
Improved Savings TSP (minute)	67.9	84.4	93.8	232.7	269.3	308.6	377.0	428.0	530.1
naïve TSPTWD (minute)	99.7	140.1	151.7	253.0	293.0	332.0	412.0	439.6	551.9
SA_01,1_drone,inter_p0 (minute)	63.9	76.6	87.8	190.5	236.1	272.1	299.3	356.1	414.9
SA_01,1_drone,inter_p1 (minute)	63.9	76.6	87.8	184.0	228.9	263.8	291.0	345.9	394.6
SA_01,1_drone,inter_p2 (minute)	63.9	76.6	87.7	182.9	227.2	262.7	291.0	340.8	392.4
SA_01,1_drone,inter_p3 (minute)	63.9	76.6	87.7	182.9	226.9	260.5	290.9	338.9	392.0
SA_01,2_drone,inter_p0 (minute)	63.9	76.6	87.8	191.2	236.8	271.6	299.2	355.4	418.0
SA_01,2_drone,inter_p1 (minute)	63.9	76.6	87.8	183.4	228.9	264.0	287.9	346.8	396.6
SA_01,2_drone,inter_p2 (minute)	63.9	76.6	87.8	183.4	227.3	262.2	287.9	343.0	392.9
SA_01,2_drone,inter_p3 (minute)	63.9	76.6	87.8	183.1	225.6	259.8	287.9	342.6	392.9
SA_01,3_drone,inter_p0 (minute)	63.9	76.6	87.8	192.1	234.7	272.4	299.5	353.2	415.2
SA_01,3_drone,inter_p1 (minute)	63.9	76.6	87.4	184.5	224.8	266.1	288.4	338.9	395.3
SA_01,3_drone,inter_p2 (minute)	63.9	76.6	87.4	182.7	224.8	265.4	288.4	338.9	394.4
SA_01,3_drone,inter_p3 (minute)	63.9	76.6	87.4	182.1	224.6	262.4	288.2	338.9	394.4

Part II: SA_02

length of side (mile)	1.5	1.5	1.5	5	5	5	8	8	8
# cus	60	80	100	60	80	100	60	80	100
Improved Savings TSP (minute)	67.9	84.4	93.8	232.7	269.3	308.6	377.0	428.0	530.1
naïve TSPTWD (minute)	99.7	140.1	151.7	253.0	293.0	332.0	412.0	439.6	551.9
SA_02,1_drone,inter_p0 (minute)	64.0	78.4	88.6	209.4	251.6	288.5	322.3	380.3	453.4
SA_02,1_drone,inter_p1 (minute)	64.0	78.4	88.6	201.2	245.1	288.0	311.8	367.8	429.1
SA_02,1_drone,inter_p2 (minute)	64.0	78.4	88.6	200.3	245.1	288.0	311.1	365.9	428.1
SA_02,1_drone,inter_p3 (minute)	64.0	78.4	88.6	199.8	244.9	287.5	308.9	364.3	427.7
SA_02,2_drone,inter_p0 (minute)	64.0	78.4	88.6	210.7	253.5	288.5	314.0	378.6	447.3
SA_02,2_drone,inter_p1 (minute)	64.0	78.4	88.6	202.5	250.3	287.3	299.7	363.8	424.7
SA_02,2_drone,inter_p2 (minute)	64.0	78.4	88.6	199.4	249.4	287.3	299.7	362.6	412.5
SA_02,2_drone,inter_p3 (minute)	64.0	78.4	88.6	199.4	249.3	287.3	299.7	362.0	412.3
SA_02,3_drone,inter_p0 (minute)	64.0	78.4	88.6	208.3	253.1	288.3	312.6	379.1	447.8
SA_02,3_drone,inter_p1 (minute)	64.0	78.4	88.6	198.9	248.8	286.6	293.1	366.5	423.8
SA_02,3_drone,inter_p2 (minute)	64.0	78.4	88.6	198.5	247.6	286.6	293.1	362.9	421.0
SA_02,3_drone,inter_p3 (minute)	64.0	78.4	88.6	198.3	247.6	286.6	293.1	362.6	424.5

Part III: SA_03

length of side (mile)	1.5	1.5	1.5	5	5	5	8	8	8
# cus	60	80	100	60	80	100	60	80	100
Improved Savings TSP (minute)	67.9	84.4	93.8	232.7	269.3	308.6	377.0	428.0	530.1
naïve TSPTWD (minute)	99.7	140.1	151.7	253.0	293.0	332.0	412.0	439.6	551.9
SA_03,1_drone,inter_p0 (minute)	63.9	76.8	87.8	193.9	235.4	272.9	298.9	352.1	415.5
SA_03,1_drone,inter_p1 (minute)	63.7	76.5	87.5	185.7	225.8	260.7	287.7	340.1	395.8
SA_03,1_drone,inter_p2 (minute)	63.4	76.5	87.5	183.1	223.8	258.1	282.0	338.7	382.9
SA_03,1_drone,inter_p3 (minute)	63.4	76.5	87.5	182.3	220.8	256.5	279.8	337.2	379.2
SA_03,2_drone,inter_p0 (minute)	63.9	77.3	88.0	190.8	231.2	269.9	284.1	348.3	399.6
SA_03,2_drone,inter_p1 (minute)	63.5	76.5	87.5	183.0	223.7	258.1	274.0	334.0	380.4
SA_03,2_drone,inter_p2 (minute)	63.3	76.5	87.5	180.7	221.4	254.6	270.0	329.3	369.7
SA_03,2_drone,inter_p3 (minute)	63.3	76.5	87.5	180.2	219.9	252.2	268.1	323.2	367.8
SA_03,3_drone,inter_p0 (minute)	63.9	77.4	88.0	188.9	231.8	267.3	279.1	346.5	396.3
SA_03,3_drone,inter_p1 (minute)	63.5	77.3	87.5	180.1	224.5	259.0	268.4	329.6	371.5
SA_03,3_drone,inter_p2 (minute)	63.5	77.3	87.5	178.9	221.0	256.1	265.3	322.1	365.4
SA_03,3_drone,inter_p3 (minute)	63.4	77.3	87.5	178.9	219.6	253.5	264.8	319.0	359.8