國 立 成 功 大 學
工 業 與 資 訊 管 理 研 究 所
碩 士 論 文

考慮目的地組合之船載單或雙無人機旅行推銷員問題研究
A Study of the Carrier Vehicle Traveling Salesman Problem Considering Target Composition with Single or Two UAVs

研 究 生 ：娜莎娣 Maharani Rizki Larasati

指導教授：王逸琳 教授 Prof. I-Lin Wang

中華民國 110 年 6 月 11 日

國立成功大學

碩士論文

考慮目的地組合之船載單或雙無人機旅行推銷員問題研究

A Study of the Carrier Vehicle Traveling Salesman Problem Considering Target Composition with Single or Two UAVs

研究生：娜莎娣

本論文業經審查及口試合格特此證明

論文考試委員：王逸琳

周詩梵

吳浩庠

指導教授：王逸琳

單位主管：[illegible]

（單位主管是否簽章授權由各院、系（所、學位學程）自訂）

中 華 民 國 110 年 6 月 11 日

摘要

本研究探討一個「船載無人機旅行推銷員問題」，旨在指揮一艘船從給定的起點出發，沿途拜訪開放水域中的 *n* 個目的地後，再抵達給定的訖點。船可以在各目的地附近施放其所承載的單或雙架無人機，由速度較快但續航力較短的無人機去目的地執行探勘或收送件任務。依不同的任務需要，無人機或可在續航力足夠的情況下，一次拜訪一或多個目的地，再與船會合，而無人機與船的每次起飛與降落之會合點不見得要相同。本研究重點在於規劃船與無人機的最佳協同任務時程與路線，以使整個任務能在最短的時間內完成。

為了尋找最佳之目的地拜訪順序以及無人機各趟的起降點座標，本研究設計了一個具旅行推銷員問題限制的混整數二階錐規劃模式(SOCP)，並使用最佳化軟體 GUROBI 求解之。依任務類別需要，本研究考慮了單(SV)或雙架無人機(TV)，及其每趟飛行僅拜訪單一(ST)或多個(MT)目的地等，總共四類問題設定。然而，使用數學規劃模式求解十分耗時，僅能處理小規模的問題。倘若給定各架無人機應造訪的目的地指派與其順序組合，本研究之混整數二階錐規劃模式即可較快計算出其對應的最佳無人機起降點座標。因此，針對 ST 類別問題，本研究提出一個二階段啟發式演算法：在第一階段用模擬退火法(ALG-SA) 猜測目的地造訪順序；而第二階段求解 SOCP 以算出其對應的最佳無人機起降座標。針對 MT 類別問題，本研究提出一個四階段啟發式演算法：第一階段先以 ST 模式直接使用 ALG-SA 與 SOCP 計算 ST 模式之各目的地起降座標；第二階段再使用一個分群演算法(CA)以指派單架無人機在 MT 設定下，可造訪的目的地集合；第三階段針對每架無人機負責的個別目的地分群，以一個基於最短路徑而設計的

演算法 (ALG-SP)進一步決定其各趟航程需造訪的目的地集合與其順序；最後，於第四階段求解 SOCP 以得到精確的各趟無人機起降點座標。

針對單趟航程多目的地(MT)以及雙架無人機(TV)的路線規劃問題，本研究所設計的數學模式與演算法皆為相關領域首創。在數值測試實驗中，本研究實作並測試了五種不同的 SOCP 模式，以得出未來研究的數學規劃模式選用建議。而本研究所設計的二階段與四階段啟發式演算法，皆能在數秒內針對中大規模(*n*=70)問題計算出品質不錯的解。

關鍵詞：混整數二階錐規劃；模擬退火；旅行推銷員問題；無人機；最短路徑

Abstract

This research investigates a carrier vehicle traveling salesman problem (CVTSP) to visit *n* targets. Each target will be visited by a smaller vehicle (e.g., unmanned aerial vehicle, UAV) launched from a carrier (e.g., ship). The vehicle will land at the carrier after visiting one or a few targets. The smaller vehicles and the carrier have to synchronize in takeoff and landing. To seek the best target visiting sequence and the associated takeoff and landing coordinates, this research presents a mixed integer second order cone programming model with TSP-like constraints and solves the problem by GUROBI, the state-of-the-art optimization software.

This research considers two settings on the number of targets that a vehicle can visit within one flight: (1) the single-target-per-flight setting (ST), and (2) the multiple-target-per-flight setting (MT), which requires a target composition that records which target to be visited for a specific flight by a specific vehicle. This research also considers two settings on the number of vehicles: (1) single vehicle (SV) and (2) two vehicles (TV). Given a sequence of targets to be visited and the corresponding target-vehicle assignment, our mathematical programming model can calculate optimal coordinates to take off and land each vehicle on a carrier. To speed up the solution process, this research proposes a Simulated Annealing algorithm (ALG-SA) to determine the target visiting sequence for all cases and a Shortest-Path-based algorithm (ALG-SP) to determine the target composition for the MT cases. The results from the computational experiments indicate that our proposed solution methods can calculate a good solution faster than the proposed mixed integer second order cone programming model by GUROBI.

Keywords: mixed-integer second-order cone programming; simulated annealing; vehicle routing; unmanned aerial vehicle; traveling salesman problem

Acknowledgments

First of all, I would like to express my deepest gratitude to my advisor, Prof. I-Lin Wang (王逸琳), for his assistance during my study, for his guidance and dedication to complete this thesis. I also would like to thank all committee members, Prof. Wu (吳浩庠), Prof. Chou (周詩梵), Prof Hsieh (謝中奇), Prof. Wang (王惠嘉). They are willing to take the time to review the thesis and put forward many valuable suggestions to make this thesis better.

In addition, I would like to thank my seniors 岳晏慈, 陳彥瑋, 王宗瀚, and Ari. To my lab mates, 胡承中, 張書桓, and Ocha. To my juniors 沈郁恩, 李蕎伃, and 徐威銓. Thank you for helping me whenever I struggle and let me have an enjoyable life in Taiwan. Furthermore, thank you for making the lab a lively place. I also want to thank all my friends, whom I cannot mention one by one, for the laughter and endless support during my study here.

Finally, I want to express my gratitude to my family for their support. They are my strongest motivation and a reason to keep moving forward despite my difficulties during my study. With many things that they worried about, they still support my decision to pursue my degree in Taiwan. I am feeling happy and grateful for their understanding so that I can focus more on my study.

Maharani Rizki Larasati

Department of Industrial and Information Management

National Cheng Kung University, Tainan City, Taiwan

August 2021

Table of Contents

List of Figure

List of Table

Chapter 1

INTRODUCTION

Unmanned aerial vehicle (UAV) has been widely applied in several industries, such as military and private sectors (Agatz et al., 2018), because of its flexibility. The UAVs are usually smaller and more flexible, thus executing a mission in areas where larger vehicles cannot access them. However, they have limited energy to complete the mission altogether. Therefore, they are often paired with larger vehicles such as ships and trucks to extend their energy limitation. A larger vehicle or carrier vehicle acts as a mobile charging device for the UAV by charging or swapping its battery.

To develop a UAV-carrier system, one of the most critical issues is path planning. The main objective of the overall system is to minimize energy consumption for both the carrier and UAVs. This research expects one or two UAVs to visit all target points from and to a carrier within its range (i.e., before the UAV runs out of energy). The path planning system can be formulated as starting and ending at some prespecified takeoff and landing locations. In this research, the UAV and the carrier are deployed in continuous space.

The UAV-carrier system aims to create an optimal UAV trajectory and carrier where the UAV visits all target points once while considering energy constraints. This type of problem is called a Carrier-Vehicle Traveling Salesman Problem (CVTSP), first introduced by Garone et al. (2011). CVTSP consists of two different vehicles:

1. A carried vehicle or a UAV (hereafter referred to as a *vehicle*) is a small vehicle that can visit all target points with limited energy.
2. A carrier vehicle (hereafter referred to as a *carrier*) is a larger vehicle that deploys and retrieves a vehicle.

This chapter is divided into four sections to elaborate further. Section 1.1 explains the background of the research. Then, in section 1.2, the research objective is described. Section 1.3 defines the research scope. Lastly, section 1.4 shows the outline of the thesis.

1.1. Research Background

This research aims to minimize the travel time of carrier vehicles or the mission makespan considering the UAV energy constraint and movement constraint. The UAV and the carrier vehicle operate in a continuous space where the position's location is distinctive. UAVs can be deployed in an uncertain environment because of their size and flexibility.

Figure 1.1 UAV is deployed from a carrier (i.e., ship)

The coordination between UAV and carrier is found in several applications. However, such path planning for both vehicles is difficult, especially in the continuous space. Determining UAV and carrier's exact position depends on several factors such as UAV energy constraint and the targets' location.

The research aims to create a path planning model for UAV and carrier where all targets are served by the UAV considering UAV's energy constraint and the coordination with a carrier as a recharge. Such operation is conducted in a continuous environment (i.e., open seas). It is applicable in different situations where path planning is necessary. Furthermore, this research can also be used to see the performance of deploying multiple UAVs to execute a mission.

The model created represents the abovementioned problem and illustrates the effectiveness of increasing the number of UAVs for the given situation. It investigates how such an increase can affect an optimal route for UAVs and their carriers while considering their energy constraints. To evaluate the performance for each case, this research conducts a numerical evaluation of the optimization model. However, a better representation or parameter is needed to create a correct model. Thus, the model's setting must be analyzed to understand the model and the problem better.

To create optimal path planning for both UAV and carrier to perform a mission, the efficiency of such path planning is essential. There are several ways to do a path planning algorithm. One of the most prominent algorithms is to treat both UAV and carrier as an integrated vehicle. Then, the path planning problem can be approximated by solving the Traveling Salesman Problem (TSP) for this integrated vehicle. This research uses a TSP solution to plan the initial paths. Furthermore, we consider the realistic UAV range constraint that calculates the UAV energy consumption. Therefore, this research considers energy consumption when the UAV does the mission by viewing each point's distance.

This research is conducted in a continuous space. Therefore, this research's model structure is based on the Second Order Cone Programming (SOCP), a convex optimization problem. Since this study used the basic idea of TSP, SOCP will be implemented together with the Integer Programming (IP) model. Thus, the model structure in this research is Mixed-Integer Second-Order Cone Programming (MISOCP). Such a problem can be solved by state-of-the-art optimization solver such as GUROBI.

However, when the problem search is large and complex, the optimization solver takes time. Thus, this problem is considered an NP-hard problem because its simplified subproblem (e.g., TSP) is already NP-complete. The complexity of calculating the best route increases when more targets are added to the problem. Therefore, efficient heuristic algorithms are necessary to solve the problem in a considerable amount of time. After knowing the order to visit the target points, we solve the problem using SOCP to calculate the exact takeoff and landing points.

1.2. Research Objective

This research aims to develop a mathematical model for two UAVs and their carrier in continuous space using the MISOCP model structure. The objective is to find an efficient model to determine the route by minimizing the total makespan with UAV's energy constraint consideration. These mathematical models are expected to have these capabilities:

1. To determine the route for the UAV and its carrier simultaneously.
2. To determine the exact location for each takeoff and landing for each flight of a UAV.
3. To determine the order of UAV takeoff and landing for the multiple UAVs case.

1.3. Research Scope

This research focuses on the case for the multiple UAVs and their carrier where UAVs and carrier are in the joint operation. This problem is defined as the Carrier-Vehicle System (Garone et al., 2008). A UAV needs to visit all targets within the space. However, due to the vehicle's limited energy, the vehicle is paired with a carrier. In this research, the carrier is assumed to have unlimited energy. The carrier can travel to every point within the space and meet with the vehicle to swap or recharge its battery and continue the mission. Note that this research is conducted in a continuous space. Thus, both vehicle and carrier are able to visit every single point within the continuous space. During the mission, the carrier can launch and retrieve the vehicle at the designated point. Carrier and vehicle can meet at the same time, or the carrier can wait for the vehicle. This Carrier-Vehicle System aims to find an optimal path within the space while considering the vehicle's energy limitation. Due to the model complexity, this research limits the number of carriers to be one only.

This research uses Mixed Integer Programming (MIP) with the basic idea from the traveling salesman problem (TSP) to find the vehicle and carrier route. By using MIP, an optimal route for both vehicle and carrier can be calculated. Furthermore, the route itself is insufficient to solve the carrier-vehicle system; the exact location of vehicle takeoff and landing must be determined within the space. The Second-Order Cone Programming (SOCP) is used to solve this problem. Combining the two models, a good route, exact vehicle flight sequences, and takeoff and landing locations can be solved simultaneously.

1.4. Research Contribution

Several studies related to the Carrier-Vehicle System (CVS) are in discrete and continuous space. This research focuses on CVS in continuous space. Previous research dealing with a similar problem solves the given situation with various methods. This research proposes an alternative method to solve similar problems by conducting a computational experiment to assess the methods' performance.

Another contribution of this research is that we have dealt with cases of two vehicles. Solving the two-vehicle cases in continuous space is a challenge due to the increasing number of possible combinations. There is only one research in literature dealing with two-vehicle cases, to the author's best knowledge. However, the existing research solves the problem using heuristics. Here we propose an exact method to solve the two-vehicle cases.

1.5. Thesis Outline

This research is organized as follows. Chapter 2 provides a literature review for the previous studies related to this research as well as research contribution. Chapter 3 describes the problem definition and formulation together with the mathematical model. Chapter 4 explains the heuristic and algorithm that has been developed for this research. Chapter 5 shows the computational experiments to show the performance of our model and algorithm. Chapter 6 explains future research and challenges, including the expectations of the result of this research.

Chapter 2

LITERATURE REVIEW

The literature review is divided into two main sections; cooperation between vehicles and their carrier also related research to solve the carrier-vehicle system.

2.1. Vehicle and Carrier Operation

Unmanned aerial vehicle (UAV) has been deployed to finish several tasks such as persistent intelligence surveillance and reconnaissance mission (Manyam et al., 2017), goods delivery (Ha et al., 2018), and disaster relief operation (Chowdhury et al., 2017).

UAV’s optimal trajectory has to be considered to ensure that the UAV works efficiently. There is some research related to the routing problem for UAVs. Yakıcı (2016) formulates a UAV routing problem for naval intelligence or surveillance mission called prize collecting location and routing problem (PCLRP). Yakıcı (2016) evaluates the UAV’s performance by collecting points on the visited nodes. Thus, the objective of this method is to maximize the score of interest points. A metaheuristic method called Ant Colony Optimization (ACO) is used to solve the problem. ACO is chosen because it provides a better solution in a short time.

Another research using simulated annealing (SA) is conducted by Dorling et al. (2017) by introducing a cost function that considers the energy consumption of UAVs. Besides considering battery, Dorling et al. (2017) also take payload weight into account. However, Dorling et al. (2017) assume one UAV can fulfill the demand on every given node, and there is only one depot for UAV to land and take off.

Manyam et al. (2017) formulated a routing problem for UAVs in intelligence surveillance & reconnaissance mission (ISR). The objective is to minimize the time to finish all of the given tasks.

Manyam et al. (2017) have a stationary depot where UAVs land and take off. However, to increase its efficiency, the depot should be a mobile depot such as an unmanned ground vehicle (Yu et al., 2018). The utilization of mobile depots also expands the coverage area of UAVs. An example of a mobile depot is a transportation mode such as a ground vehicle (GV). Since UAVs are known for their small coverage, UAVs are combined with a mobile depot so that when UAV moves, the mobile depot also moves. Thus, the implementation of the mobile depot expands the coverage area.

Despite UAV's broad application, UAV has limited energy, which leads to a limited range. Due to the restrictions, recent research combines UAV and ground vehicle (GV) (Luo et al., 2017). Ground vehicle functions as a landing station for UAV; to recharge the battery (Yu et al., 2018) or swap UAV's battery while GV charges another battery (Luo et al., 2017). The complementary features of UAV and carrier are written in Table 2.1.

Table 2.1 UAV and carrier complementary features (Luo et al., 2017)

	Speed	**Weight**	**Capacity**	**Range**
UAV	High	Light	One	Short
GV	Low	Heavy	Many	Long

The combination of UAV and carrier allows UAV to have a wider coverage area, affecting both vehicles' trajectories. UAVs have to operate along with carriers to ensure that UAV land on a carrier at the right time. Another benefit of using the UAV-carrier combination is that completion time can be reduced to 75% compared with a carrier-only task (Wang et al., 2016).

Otto et al. (2018) summarize the most common application for combined operations in Figure 2.1.

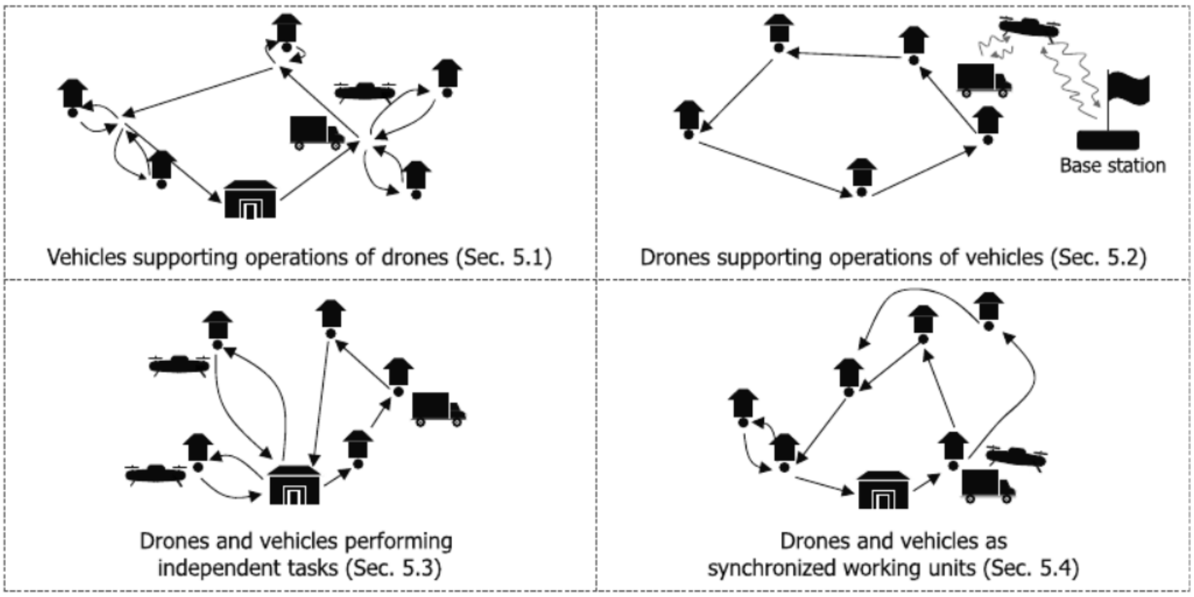

Vehicles supporting operations of drones (Sec. 5.1)
Drones supporting operations of vehicles (Sec. 5.2)
Base station
Drones and vehicles performing independent tasks (Sec. 5.3)
Drones and vehicles as synchronized working units (Sec. 5.4)

Figure 2.1 Illustrative examples of different types of combined operations (Otto et al., 2018)

This research application is most similar to the 'vehicle supporting operation of drones.' The main focus is a smaller vehicle executing a mission. Due to its endurance (i.e., range) constraint, a larger vehicle is deployed to extend the range. The combined operation with vehicle supporting units can be applied in various applications, from agriculture to oceanographic sampling. The support vehicle (carrier) is slower in this operation but travels longer than drones (UAVs). UAVs land on the carrier to recharge or swap batteries is the most common one. Another common model is that a UAV travels with the carrier on the ground without considering any energy loss. For example, in Tokekar et al. (2016), a carrier can retrieve, launch, and travel with a UAV, which has to take aerial images as many points as possible. The authors model the problem as an orienteering problem.

2.2. Solving the Carrier-Vehicle System Problem

There are several types of research that deal with the carrier-vehicle system (CVS), and this section divides the discussion into two topics; solving CVS in the discrete space or the continuous space.

Discrete space is where the points form a discontinuous sequence, or in other words, the points are isolated from each other. Meanwhile, continuous space has an infinite number of points in the space. Figure 2.2 shows the example of discrete space by Otto et al. (2018). Figure 2.3 illustrates a continuous space by Poikonen and Golden (2020). Note that the continuous space can be discretized to make the formulation linear.

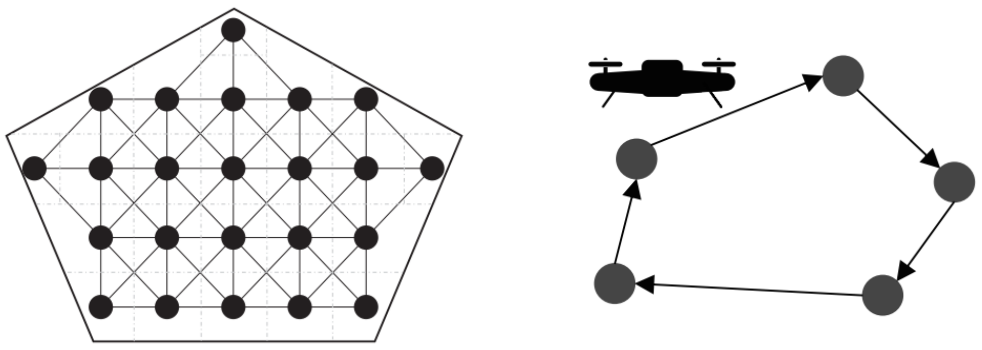


Figure 2.2 Routing in discrete space (Otto et al., 2018)

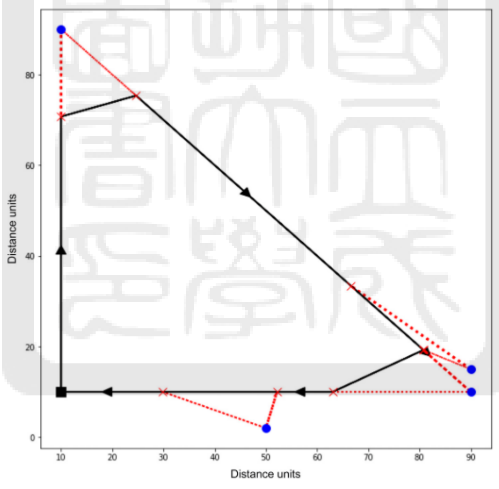


Figure 2.3 Routing in continuous space (Poikonen and Golden, 2020).

2.2.1. Discrete Space

To solve CVS, one of the most common formulations is Integer Programming (IP) (Agatz et al., 2018; Ha et al., 2018; Luo et al., 2017)). Integer programming is used because of the problem's characteristics, which is a two-echelon cooperative routing problem. This formulation represents a decision where it is important to know the route or trajectory used.

Usually, we solve such kinds of problems by heuristics. There are several proposed heuristics.

For instance, Ha et al. (2018) propose two heuristic approaches. The first method is to “cluster first – route second.” The vehicle routes are determined first, and the carrier routes are determined using the given vehicle nodes. The alternative is to “route first – cluster second.” The carrier routes are defined first, and vehicle nodes are selected afterward.

The route first–cluster second method is also used by Agatz et al. (2018). The carrier functions as a mobile depot, and the carrier can also serve demand in the given nodes. The problem is formulated using integer programming and solved using greedy partitioning heuristics. The result of greedy partitioning heuristics is compared to the exact partitioning. Based on the research, exact partitioning is better than greedy partitioning. Figure 2.4 illustrates the expected result of their formulation by Agatz et al. (2018)

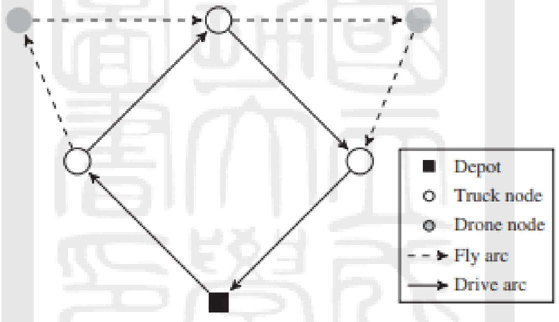


Figure 2.4 An example of TSP-D solution (Agatz et al., 2018)

Despite the better result, exact partitioning requires a long computational time and is applicable for a small-size model. Thus, Luo et al. (2017) develop a two-echelon routing problem for carriers (UAV) and vehicles (UAV) called 2E-GU-RP using genetic algorithms (GA). Figure 2.5 shows the diagram for 2E-GU-RP by Luo et al. (2017). They use two types of heuristics; the vehicle route is determined first, followed by the carrier route, the second heuristics is vice versa. The result shows that both heuristics are effective and efficient.

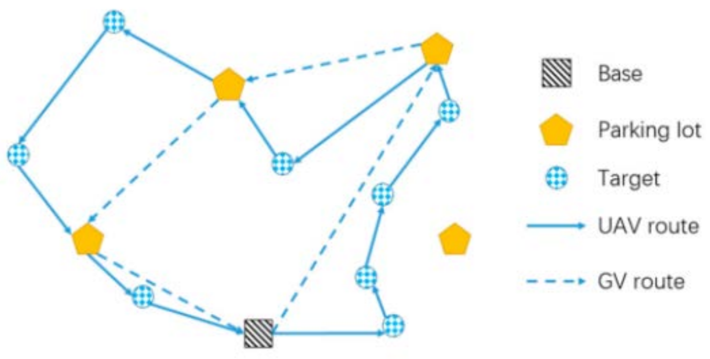


Figure 2.5 Schematic diagram for 2E-GU-RP (Luo et al., 2017)

Another heuristic method called Greedy Randomized Adaptive Search Procedure (GRASP) is used to solve the problem (Ha et al., 2018). GRASP method consists of three steps that make a giant tour; *k*-nearest neighbor, *k*-cheapest insertion, and random insertion.

Besides heuristics, Wang et al. (2016) introduce a vehicle routing problem with vehicle (VRPD). VRPD is close to the conventional vehicle routing problem but takes the vehicle into account.

The objective of a cooperated routing problem for vehicles and carriers is to find the optimal route that satisfies the constraints. Li and Garcia-Luna-Aceves (2006) solved it using the depth-first search method to find a feasible path. The research uses an extended depth-first search (EDFS). EDFS outperforms other methods when dealing with a multi-constrained path. Duhamel et al. (2011) also develop an extended depth-first search and combine a greedy algorithm to solve a routing problem. Branch and bound are also often used in the vehicle and carrier hybrid (Poikonen et al., 2019). Each node of the tree corresponds to a possible order to deliver a subset of packages.

2.2.2. Continuous Space

One of the most common methods to solve the carrier-vehicle system is using the traveling salesman problem in continuous space. Such a problem is called Carrier-Vehicle Traveling Salesman Problem (CVTSP). CVTSP is a problem when a carrier launches a vehicle to visit a target point and return to recharge. The vehicle and carrier move to the following designated takeoff point and continue the mission until all target points are served. CVTSP was first mentioned by Garone et al. (2010). Here, Garone et al. (2010) solved a problem using heuristics, given visitation sequence. This problem's objective is to find the number of takeoffs and points visited for each takeoff and landing.

Furthermore, they also need to find the continuous trajectories of the vehicle and carrier. They named the Euclidean Traveling Salesman Problem (E-TSP) method, where the result from E-TSP will be plugged into the convex optimization formulation. Later, Garone et al. (2012) introduced a general problem of CVTSP, where the order of visits is unknown beforehand. Heuristics is implemented to find the target visiting sequence.

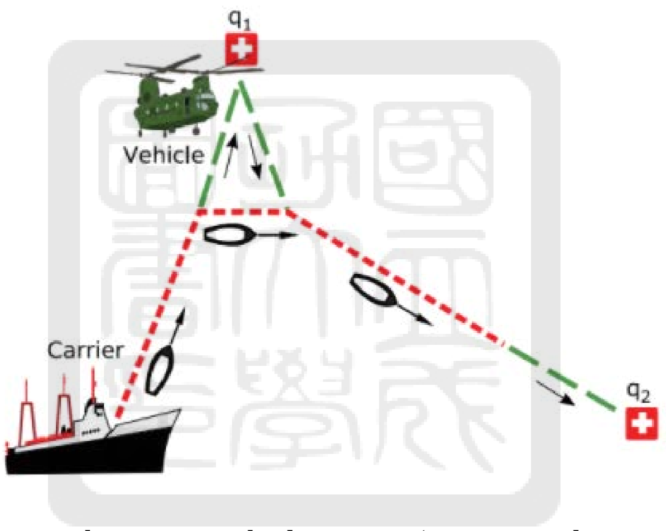


Figure 2.6 The carrier-vehicle system (Garone et al., 2012)

Since Garone et al. (2012) used a Mixed Integer Non-linear Programming (MI-NLP), Klaučo et al. (2014) transform the problem to Mixed Integer Second Order Cone Programming (MISOCP). Furthermore, the formulation in Klaučo et al. (2014) solve multiple target points within one vehicle visit given the order visitation sequence.

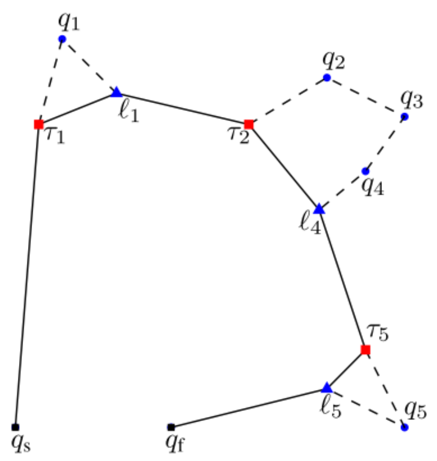


Figure 2.7 Optimal path for multiple target points (Klaučo et al., 2014)

Also solved using MISOCP, Gambella et al. (2018) provide a formulation to solve one target point for one vehicle visit given unordered target points. Their main contribution is proposing a ranking-based algorithm (RBA) to rank the possible path in ascending order. In contrast with Gambella et al. (2018), where they introduce an assignment variable to solve CVTSP, Erdoğan and Yıldırım (2020) emphasize more on the routing variables and subtour elimination constraints.

Another method to solve CVTSP is proposed by Poikonen and Golden (2020), where he use a branch-and-bound algorithm to determine the target visiting sequence. The best order is used as an input for the Second-order Cone Programming (SOCP) to know the exact location of takeoff and landing. Besides providing formulation for one target point per vehicle visit, Poikonen and Golden (2020) also provide the formulation to visit multiple target points per vehicle takeoff and landing. To know the best sequence to visit, they also use a branch-and-bound algorithm.

In this field of research, there is also some extension. There are three directions for extension; increasing the number of vehicles, increasing the number of carriers, and combining multiple vehicles and multiple carriers. Chen et al. (2019) propose a non-linear formulation and a particle swarm optimization (PSO) for two vehicles, given an ordered set of target points for the two-vehicle-single-carrier case. On the formulation, Chen et al. (2019) introduce a binary variable representing each vehicle’s takeoff and landing sequence.

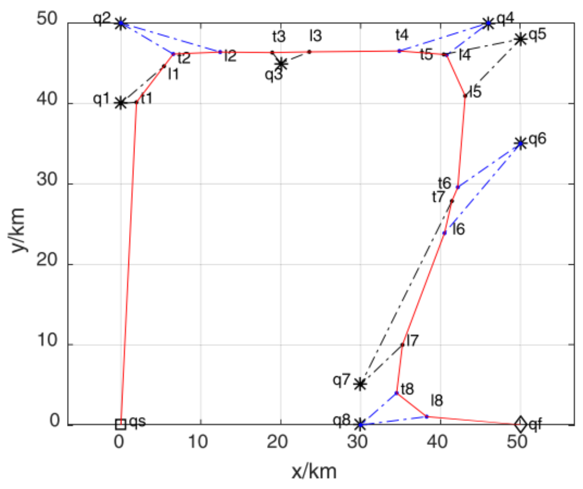


Figure 2.8 Path planning for two vehicles single carrier case (Chen et al., 2019)

Another extension is multiple-carrier and single-vehicle proposed by Fahradyan et al. (2020). They used a MISOCP formulation for the case of a given order of visit. They also provide a heuristic to solve a problem in a decent computational time.

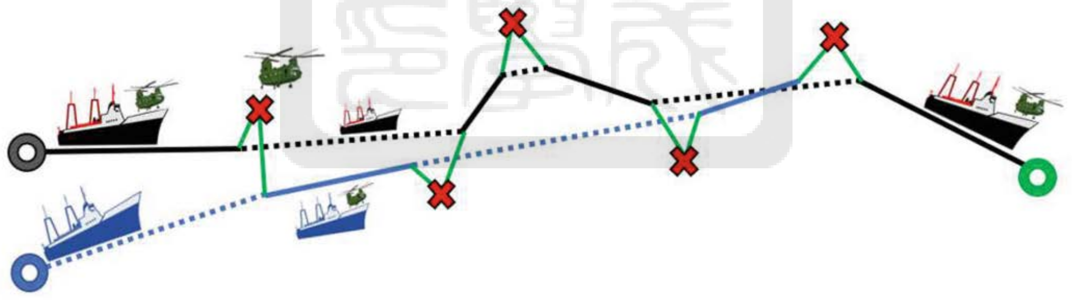


Figure 2.9 Path planning for multiple-carrier single-vehicle (Fahradyan et al., 2020)

Since this research focuses on continuous space, below is the research summary related to the carrier-vehicle system shown in Table 2.2.

Table 2.2 Related research and contribution

Literature	**Method**			
	System	**Math Model**	**Heuristics**	**Heuristics**
Garone et al. (2011)	SVST		V	
Garone et al. (2011)	SVST		V	
Gambella et al. (2018)	SVST	V	V	Ranking-based algorithm
Erdoğan et al. (2020)	SVST	V	V	ILS
Garone et al. (2014)	SVMT		V	Three-phase heuristics
Klaučo et al. (2014)	SVMT	V		
Poikonen et al. (2020)	SVMT	V	V	Branch-and bound
Chen et al. (2020)	TVST		V	PSO
Fahradyan et al. (2020)	SVST (mC)	V	V	SOCP-heuristics
This Research	**TVMT**	**V**	**V**	**Simulated annealing**

Based on Table 2.2, several kinds of research deal with a carrier-vehicle system (CVS). Each research has a specific method to solve the problem. Figure 2.10 illustrates the relationship and the difference between the previous research and this research.

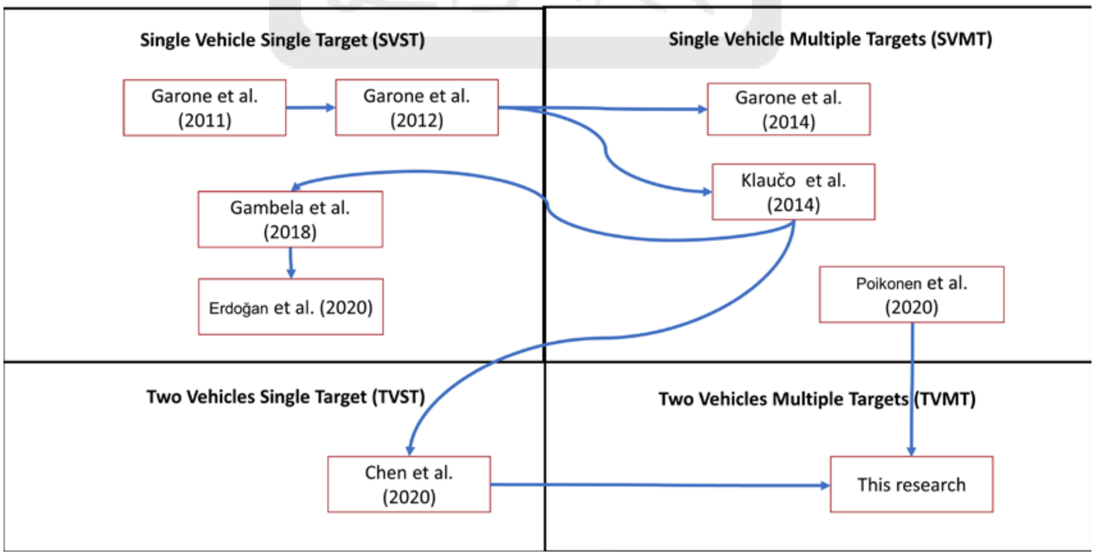


Figure 2.10 Relationship between literature

The early research dealing with the carrier-vehicle system (CVS) in a continuous space solves a single vehicle single target using convex optimization (Garone et al., 2011; Garone et al., 2012). Garone et al., 2014. However, convex optimization is not suitable to solve larger networks due to computational limitations. Thus, Klaučo et al. (2014) transform the formulation into a second order cone programming (SOCP) with a shorter computational time. However, Klaučo et al. (2014) assume that the target visiting sequence is given. Gambella et al. (2018) provide the formulation with an assignment variable for a single target that could solve the target visiting sequence to generalize this problem. In contrast with Gambella et al. (2018), Erdoğan and Yıldırım (2020) propose a formulation for a single target with subtour elimination constraints. This research also proposes a CVTSP formulation with subtour elimination but not limited to a single target. This research extends it further to multiple targets.

Poikonen and Golden (2020) propose an alternative SOCP formulation with a branch-and-bound algorithm to solve the target visiting sequence. Furthermore, Poikonen and Golden (2020) also introduce the term *target composition* to represent targets the vehicle can visit without returning to the carrier. Unlike Poikonen and Golden (2020), this research develops a simulated annealing-based algorithm to solve the target visiting sequence.

Chen et al. (2019) become the first research to solve the two-vehicle CVTSP using particle swarm optimization (PSO) to determine the takeoff and landing coordinate. Unlike Chen et al. (2019), this research uses an exact solution method to determine the takeoff and landing coordinates. In particular, a shortest-path-based algorithm (ALG-SP) is introduced to determine the target composition and vehicle assignment.

Chapter 3

MATHEMATICAL PROGRAMMING MODELS

This chapter explains the problem description and its characteristics to understand the problem settings better.

3.1. Problem Description

This research deals with the carrier-vehicle system by defining the carrier-vehicle traveling salesman problem (CVTSP). In the CVTSP system, there are two different vehicles:

1. A carried vehicle is a smaller and faster vehicle to visit all target points. Vehicles have limited energy. Therefore, vehicles need to be retrieved by a carrier.

2. A carrier vehicle is a larger vehicle to launch and retrieve a carried vehicle. Carriers have unlimited energy. Thus, in the system, the carrier acts as a recharging or swapping station for the vehicle.

Vehicle and carrier begin and finish the mission together from the same location and assume full energy at the beginning of the mission. Target points are known prior and remain the same for the rest of the mission. When a vehicle is separated from the carrier, the vehicle consumes energy. It will be fully charged after the vehicle lands on the carrier.

In this problem, the vehicle and carrier operations in the Euclidean space means the carrier can launch and retrieve the vehicle at any location, which is a continuous problem. Thus, this research formulates this problem using second order cone programming.

Coordinates on the Euclidean plane (represented by (x, y)) represent the exact location. Both vehicles and carriers can do any continuous trajectory within the Euclidean plane $\Re^2$. The vehicle and carrier travel together whenever the vehicle is not flying to visit the target point. A carrier must retrieve the vehicle before the vehicle runs out of energy. The time to charge or swap the battery is assumed negligible. Therefore, by the time the vehicle lands, its energy is immediately restored. It is assumed that the duration to stay over a target is also negligible.

Both vehicles travel at a constant speed where the carrier moves slower than the vehicle. The objective of CVTSP is to create a route for carriers and vehicles where the target points are visited with the least time. Flight durations and times when vehicle takeoff and land are continuous variables (not restricted to discrete-time intervals).

This research considers two settings on the number of vehicles: single vehicle (SV) and two vehicles (TV), paired with a carrier vehicle. Since only one carrier is considered, every case in this thesis is paired with a single carrier. This research investigates four problem types based on the number of vehicles and the number of targets visited per flight. Table 3.1 shows the difference between the four models.

Table 3.1 Four Models Developed

Vehicle \ # target(s) per flight	**Single Target (ST)**	**Multiple Targets(MT)**
Single Vehicle (SV)	SVST	SVMT
Two Vehicles (TV)	TVST	TVMT

Since one of the objectives of this research is to know the target visiting sequence, which is similar to the traveling salesman problem (TSP). Thus, this research starts with the traveling salesman problem and extends it to the four above problem types. Suppose there is a random network illustrated in Figure 3.1. Since the target visiting sequence is one of the objectives, the random network is treated as a traveling salesman problem. The carrier travel time is used as the objective function. The carrier travel time can be considered the makespan since the vehicle will depend highly on the carrier. Figure 3.2 illustrates the result of the traveling salesman problem.

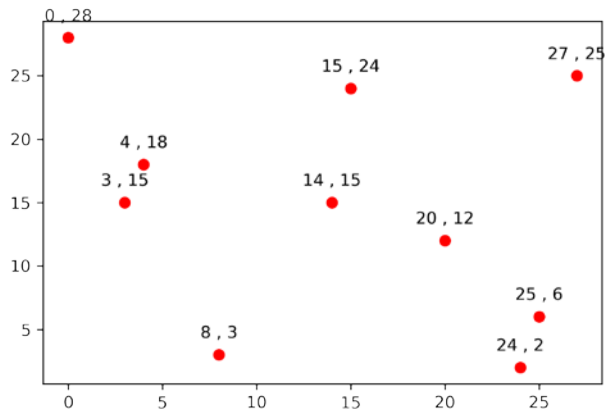


Figure 3.1 Random Network

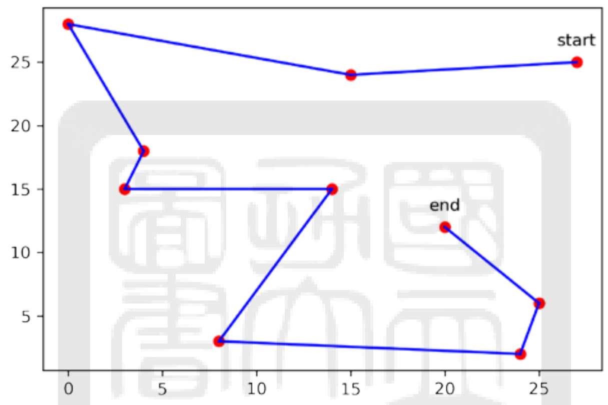


Figure 3.2 Traveling Salesman Problem

The traveling salesman problem can produce a target visiting sequence. However, we assume the carrier does not have to visit the target. The carrier only needs to get close to a target and deploy a vehicle to visit the target. Thus, Figure 3.3 illustrates the carrier's condition close to the target and deploys the vehicle. Since there is only one vehicle, and the vehicle visit one target per flight, this scenario is called *Single Vehicle Single Target* (SVST).

The SVST's decision variable is the exact takeoff and landing point for each target point and the visitation sequence. A TSP-like formulation is implemented to the model to know the order of visitation, an NP-hard problem. Therefore, as the problem becomes large and complex, it becomes too difficult to solve.

Scenario SVST can be extended by allowing the vehicle to visit more than one target as long as the vehicle's endurance is sufficient. The scenario where there is only one vehicle in the system and the vehicle visit multiple targets is called *Single Vehicle Multiple Targets* (SVMT), illustrated in Figure 3.4.

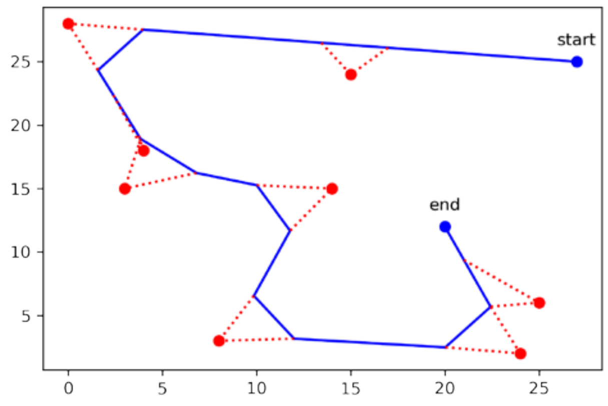


Figure 3.3 Single Vehicle Single Target (SVST)

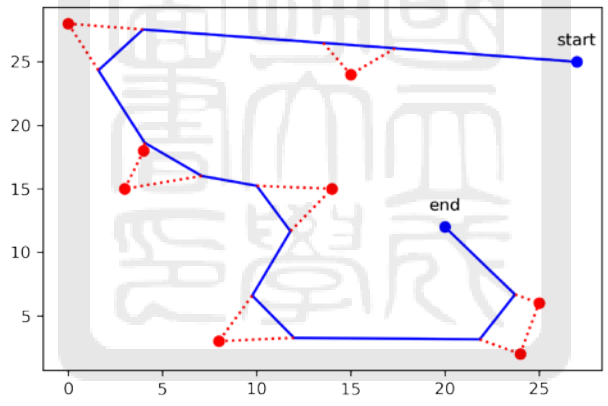


Figure 3.4 Single Vehicle Multiple Targets (SVMT)

Another extension from these two models is to consider an additional vehicle (thus, two vehicles). In several networks, there are some targets close to each other. Since the targets are close to each other, the carrier can deploy multiple vehicles simultaneously, and each vehicle can visit (i.e., serve) a different target. By doing so, the carrier does not need to move closer to each target. The carrier only needs to move to a point sufficiently close to a target and deploy vehicles at a specific point. This research limits the number of vehicles to two due to the computational complexity. This scenario is called *Two Vehicles Single Target* (TVST), as illustrated in Figure 3.5.

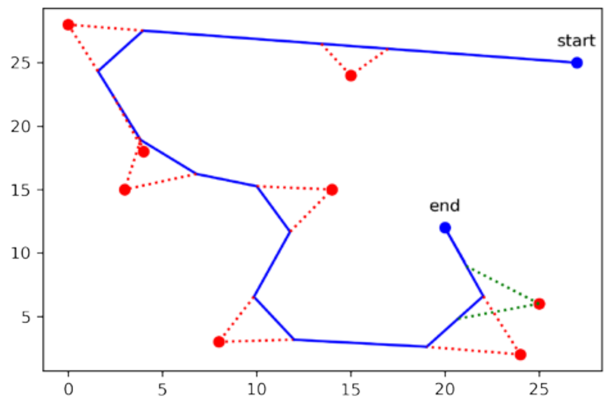


Figure 3.5 Two Vehicles Single Target (TVST)

Like the previous scenario, TVST can be extended further. Each vehicle can visit multiple targets per flight as long as the vehicle's endurance is sufficient. This scenario is called *Two Vehicles Multiple Targets* (TVMT), as illustrated in Figure 3.6.

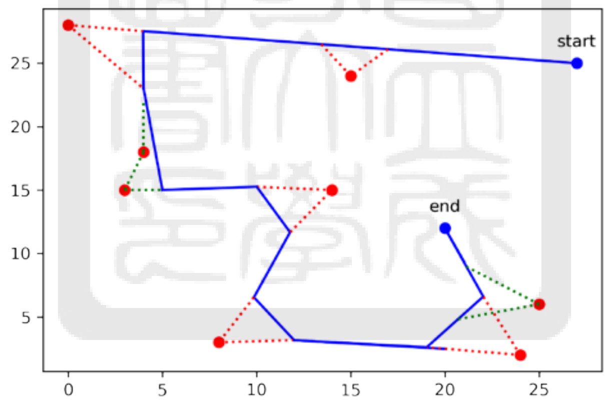


Figure 3.6 Two Vehicles Multiple Targets (TVMT)

Like the single-vehicle scenario, the objective is to know the exact location for a vehicle to take off and land and determine the target visiting sequence. Besides these two decision variables, each vehicle's takeoff and landing sequence must also be determined if the target points are within some proximity.

3.2. Problem Definition

Let S be a set of target points in $\Re^2$ that need to be visited by vehicles. A vehicle and carrier have speed V_v and V_c respectively where $V_v > V_c$. Both the vehicle and carrier start the mission together from the same point.

During the mission, when the target *i* (with coordinate s_i) is within the vehicle's operational range *R* time unit, a vehicle can be launched from its takeoff coordinate to_i to s_i. The distance that the vehicle travels is equal to $dOut_i$. After visiting s_i, the vehicle is retrieved by the carrier on a landing coordinate l_i. The distance traveled by this vehicle is equal to dIn_i. Note that when the vehicle is deployed, the duration of the entire vehicle trip cannot exceed *R*.

To represent the route assignment, this research introduces a binary variable x_{ij} equal to 1 if target *j* is visited after target *i*. This research also introduces a variable u_i for subtour elimination. When the carrier and vehicle travel together, the time spent by the carrier from l_i to the next launching coordinate to_j is tc_{ij}. After reaching the takeoff point and the vehicle is deployed to serve target s_i, the carrier travels to the landing point l_i, and the time spent by the carrier is represented by ts_i. Besides traveling, the carrier is also able to wait for the vehicle to land. The idle/waiting time spent by the carrier to wait for the vehicle (without traveling) is defined as wt_i. The objective function of this model is to minimize the makespan of the mission. The makespan can be represented as the carrier travel time. The following assumptions are introduced to develop our problem into a mathematical programming model:

1. The service times for all target points and time to swap or recharge the battery are negligible. Therefore, when a vehicle lands, its energy is restored immediately.
2. All the target points are given and need to be visited
3. The carrier has unlimited energy.
4. The speeds of vehicle and carrier speed remain constants of different values throughout the mission.

This research proposes two main models; the single-vehicle model and the two vehicles model. Each model will be described further in the next section.

3.3. Single-Vehicle Model (SV)

There are one vehicle and one carrier in this system where the vehicle needs to visit all target points. Both vehicle and carrier are deployed in a continuous space. Due to the vehicle's energy limitation, a vehicle cannot be launched too far away from the target. On the other hand, the carrier also needs to minimize its own traveled distance. Thus, it is essential to determine optimal takeoff and landing points to ensure that the carrier can travel its shortest distance while considering the vehicle's energy constraint.

The vehicle endurance may still be sufficient to visit multiple targets without returning to the carrier in between when doing its mission, hereafter named as a *target composition*. The single-target-per-flight scenario and the multiple-target-per-flight scenario have different mathematical models. Therefore, this research divides the discussion into two subsections: single-target-per-flight scenario and multiple-target-per-flight scenario (target composition).

3.3.1. Single-target-per-flight scenario (SVST)

In this case, the vehicle is only able to visit one target per flight. This means that after visiting a target, the vehicle will immediately return to the carrier.

1. Notation

The notations used for the mathematic model are listed below:

Parameter

n	:	Number of target points
V_v	:	Vehicle speed
V_c	:	Carrier speed $(V_v > V_c)$

R	:	Vehicle operational range (time unit)
s_i	:	Target point coordinates in $\Re^2$ $i = 1, \ldots, n$

Decision Variable

to_i	:	Takeoff coordinate in $\Re^2$ for each target i
l_i	:	Land coordinate in $\Re^2$ for each target i
x_{ij}	:	1, if target j is visited after target i 0, if otherwise
u_i	:	variable to represent the ordering of visit for subtour elimination
ts_i	:	time spent by the carrier to travel from to_i to l_i or when vehicle and carrier separated
wt_i	:	time spent by the carrier to wait when vehicle travel from to_i to l_i
tc_{ij}	:	time spent by the carrier to travel from l_i to to_i or when vehicle and carrier combined
$dOut_i$	:	distance traveled by vehicle to move from to_i to s_i
dIn_i	:	distance traveled by vehicle to move from t_i to s_i

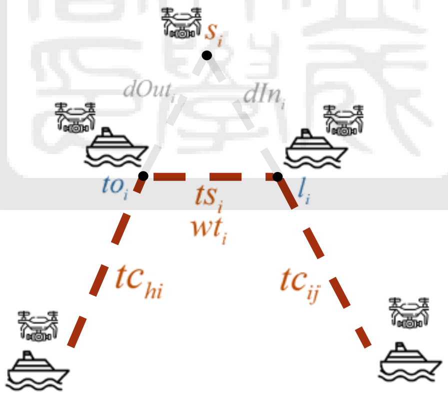


Figure 3.7 Notation illustration

2. *Mathematics model*

Given a set of targets, one vehicle is deployed to visit (serve) all the target points. In this case, the vehicle only visits one target point per takeoff and landing. The objective function minimizes the makespan or carrier travel time to finish the entire mission, as shown in equation (3.1).

$$\text{Minimize} \sum_{i=0}^{n+1} \sum_{j=0, j\neq i}^{n+1} tc_{ij} x_{ij} + \sum_{i=1}^{n} ts_i + \sum_{i=1}^{n} wt_i \tag{3.1}$$

Equations (3.2) – (3.3) are the flow balance for each target point. This ensures that all targets are visited only once. A binary variable x_{ij} represents this condition.

$$\sum_{i=0, i\neq j}^{n+1} x_{ij} = 1 \quad \forall\, j = 0, \ldots, n+1 \tag{3.2}$$

$$\sum_{j=0, i\neq j}^{n+1} x_{ij} = 1 \quad \forall\, i = 0, \ldots, n+1 \tag{3.3}$$

Equation (3.4) – (3.5) related to the subtour elimination. This research implements the Miller-Tucker-Zemlin or MTZ Formulation by introducing variable u_i as the subtour elimination constraints.

$$u_i - u_j + (n+2) x_{ij} \leq n+1 \quad \forall\ 1 \leq i, j \leq n+1,\ i \neq j \tag{3.4}$$

$$u_i \leq n+1 \quad \forall\ 1 \leq i \leq n+1 \tag{3.5}$$

Equation (3.6) ensures that tc_{ij} is at least the carrier travel time from l_i to to_j or when the carrier travels with the vehicle.

$$\| l_i - to_j \| \leq tc_{ij} \quad \forall i, j = 1, \ldots, n,\ i \neq j \tag{3.6}$$

Equation (3.7) ensures that ts_i is at least the carrier travel time from to_i to l_i or when the vehicle is deployed to visit target s_i.

$$\| to_i - l_i \| \leq ts_i \quad \forall i = 1, \ldots, n \tag{3.7}$$

Equations (3.8) – (3.9) show the relationship between distance traveled by vehicle and distance from takeoff to_i and landing l_i to the target s_i.

$$\| s_i - to_i \| \leq dOut_i \quad \forall i = 1, \ldots, n \tag{3.8}$$

$$\| s_i - l_i \| \leq dIn_i \quad \forall i = 1, \ldots, n \tag{3.9}$$

Equation (3.10) ensures that the carrier travel time ts_i and waiting time wt_i are at least the vehicle travel time when deployed.

$$(dOut_i + dIn_i) / V_v \leq ts_i + wt_i \quad \forall i = 1, \ldots, n \tag{3.10}$$

Equation (3.11) ensures that each flight time for a vehicle cannot exceed the vehicle operational range *R*.

$$ts_i + wt_i \leq R \quad \forall i = 1,...,n \tag{3.11}$$

Equation (3.12) bounds the variable for the subtour elimination.

$$u_i \in \mathrm{Z} \quad \forall i = 2,...,n \tag{3.12}$$

Finally, equation (3.13) expresses the requirement for the variable to be binary.

$$x_{ij} \in \{0,1\} \quad \forall i, j = 1,...,n \tag{3.13}$$

This research uses state-of-the-art optimization solvers as an exact solution for the model to solve the problem above. However, based on the performance obtained by optimization solvers, the model resulted in a quadratic objective function. Therefore, a transformation is needed to make a linear objective function. This research introduces a new continuous variable $\overset{\wedge}{tc_{ij}}$ where $\overset{\wedge}{tc_{ij}} = tc_{ij}x_{ij}$. This research needs to specify the upper and lower bound to make the function holds.

$$Lx_{ij} \leq \overset{\wedge}{tc_{ij}} \leq Ux_{ij} \tag{3.14}$$

$$L\left(1 - x_{ij}\right) \leq tc_{ij} - \overset{\wedge}{tc_{ij}} \leq U\left(1 - x_{ij}\right) \tag{3.15}$$

Since there is a new variable, the objective function needs to be changed to accommodate the new variable.

$$\text{Minimize} \sum_{i=0}^{n+1} \sum_{j=0, j \neq i}^{n+1} \overset{\wedge}{tc_{ij}} + \sum_{i=1}^{n} ts_i + \sum_{i=1}^{n} wt_i \tag{3.16}$$

3.3.2. Multiple-target-per-flight scenario (SVMT)

For this case, since the vehicle can visit multiple targets without returning to the carrier, calculating the traveled distance between targets is also needed. This formulation requires the information about which target is combined as an input. This subproblem will be solved using an algorithm, which will be discussed later in the chapter.

A preprocessing is needed to calculate the distance between each target which be used later in the formulation. The main concept to calculate vehicle traveled distance in between each target is adopted from (Poikonen and Golden, 2020). *Vehicle subtour* $ST_i = \left(st_{i_1}, st_{i_2}, \ldots, st_{i_z}\right)$ is an ordered set of target locations that the vehicle visits consecutively without returning to the mothership in between. The distance between targets is calculated with this formulation:

$$len\left(ST_i\right) = \sum_{j=1}^{z-1} \left\| st_{i_{j+1}} - st_{i_j} \right\| \tag{3.17}$$

The difference between the target composition case and the single-vehicle case is that the target composition case introduces another variable to represent the distance between the target for composition *i*. Furthermore, since this research uses target composition, ensuring that it corresponds correctly with which target is important for every constraint related to takeoff and landing. For example, suppose there is a composition $ST_1 = \left(st_{1_1}, st_{1_2}, st_{1_3}\right)$. To represent the distance between takeoff point and target composition ST_1, st_{1_1} must be used as a constraint since st_{1_1} is the first target point of the subtour. The same goes with the landing point, where st_{1_3} must be used as the last target point of the subtour. To deal with this problem, let $first\left(ST_i\right) = st_{i_1}$ denote the first target location for ST_i and $last\left(ST_i\right) = st_{i_z}$ denote the last target location for ST_i. The complete notation and mathematical model are written as follows:

1. *Notation*

Parameter

n	:	Number of target points
V_v	:	Vehicle speed
V_c	:	Carrier speed $(V_v > V_c)$
R	:	Vehicle operational range (time unit)
s_i	:	Target point coordinates in $\Re^2$ $i = 1, \ldots,$ n

Decision Variable

to_i	:	Takeoff coordinate in $\Re^2$ for each target *i*

l_i	:	Landing coordinate in $\Re^2$ for each target *i*
x_{ij}	:	1, if target *j* is visited right after target *i* 0, if otherwise
u_i	:	variable to represent the node ordering of visit for subtour elimination
ts_i	:	time spent by the carrier to travel from to_i to l_i or when vehicle and carrier separate each other
wt_i	:	time spent by the carrier to wait when vehicle travel from to_i to l_i
tc_{ij}	:	time spent by the carrier to travel from l_i to to_i or when vehicle and carrier meet each other
$\overset{\wedge}{tc}_{ij}$	:	variable to represent the product value between tc_{ij} and x_{ij}
$dOut_i$	:	distance traveled by vehicle to move from to_i to s_i
dIn_i	:	distance traveled by vehicle to move from t_i to s_i
$dIntra_i$	:	distance traveled by vehicle to move in between target composition *i*

2. *Mathematics model*

This formulation represents the mathematical programming model for this case:

$$\text{preprocess} \quad len(ST_i) = \sum_{j=1}^{z-1} \left\| st_{i_{j+1}} - st_{i_j} \right\| \tag{3.18}$$

$$\text{Minimize} \sum_{i=0}^{n+1} \sum_{j=0,\, j\neq i}^{n+1} \overset{\wedge}{tc}_{ij} + \sum_{i=1}^{n} ts_i + \sum_{i=1}^{n} wt_i \tag{3.19}$$

$$\sum_{i=0, i\neq j}^{n+1} x_{ij} = 1 \quad \forall\, j = 0,\ldots,n+1 \tag{3.20}$$

$$\sum_{j=0, i\neq j}^{n+1} x_{ij} = 1 \quad \forall\, i = 0,\ldots,n+1 \tag{3.21}$$

$$u_i - u_j + (n+2)x_{ij} \le n+1 \quad \forall\, 1 \le i, j \le n+1,\; i \neq j \tag{3.22}$$

$$u_i \le n+1 \quad \forall\, 1 \le i \le n+1 \tag{3.23}$$

$$Lx_{ij} \le \overset{\wedge}{tc}_{ij} \le Ux_{ij} \tag{3.24}$$

$$L(1 - x_{ij}) \le tc_{ij} - \overset{\wedge}{tc}_{ij} \le U(1 - x_{ij}) \tag{3.25}$$

$$\| l_i - to_j \| \le tc_{ij} \quad \forall i, j = 1,\ldots,n,\; i \neq j \tag{3.26}$$

$$\| to_i - l_i \| \le ts_i \quad \forall i = 1,\ldots,n \tag{3.27}$$

$$\| first(s_i) - to_i \| \le dOut_i \quad \forall i = 1,\ldots,n \tag{3.28}$$

$$\| last(s_i) - l_i \| \le dIn_i \quad \forall i = 1,\ldots,n \tag{3.29}$$

$$\left(dOut_i + dIntra_i + dIn_i\right)/V_v \le ts_i + wt_i \quad \forall i = 1,\ldots,n \tag{3.30}$$

$$len\left(ST_i\right) \le dIntra_i \quad \forall i = 1,\ldots,n \tag{3.31}$$

$$ts_i + wt_i \le R \quad \forall i = 1,\ldots,n \tag{3.32}$$

$$u_i \in \mathrm{Z} \quad \forall i = 2,\ldots,n \tag{3.33}$$

$$x_{ij} \in \{0,1\} \quad \forall i, j = 1,\ldots,n \tag{3.34}$$

Equation (3.18) is to calculate the constant for the length between targets. The objective function (3.19) is the mission completion time (hereafter referred to as the makespan), the precise time required for the carrier to finish the entire mission. Constraints (3.20) and (3.21) are the flow balance constraints for each target point, respectively. Constraints (3.22) and (3.23) are related to the subtour elimination. Constraint (3.24) and (3.25) are related to the new variable to ensure the model's objective function is linear. Constraint (3.26) ensures that tc_{ij} is at least the carrier travel time from l_i to to_j. Constraint (3.27) ensures at least the carrier travel time from to_i to l_i when the vehicle is deployed. Constraints (3.28) and (3.29) show the relationship between the distance traveled by the vehicle and the distance from takeoff to_i and landing l_i to the target s_i. Constraint (3.30) ensures that the carrier travel time ts_i and waiting time wt_i are at least the vehicle travel time when deployed. Constraint (3.31) ensures that the distance between vehicle targets does not violate the preprocessed constant. Constraint (3.32) ensures that the vehicle travel time is not beyond the vehicle operational range when deployed. Constraint (3.33) bounds the variable for the subtour elimination. Finally, constraint (3.34) expresses the requirement for the variable to be binary. To solve the problem given above, this research uses GUROBI as the solver to find an exact solution for the model.

3.4. Two Vehicles Model (TV)

This research also introduces an optimization method for two vehicles and a single carrier. The main idea to design the formulation for two vehicles is based on the single-vehicle formulation. The difference is deploying two vehicles to serve the targets to minimize the total travel time. Each vehicle serves one or more target points and then lands at the carrier.

Note that when deploying two vehicles to serve targets, the targets need to be within proximity. This condition must be met to ensure that the carrier can retrieve both vehicles without violating their operational range constraint. Since the targets need to be within proximity, the latter's formulation only applies to targets that meet the condition.

Increasing the number of vehicles will increase the problem complexity. For multiple-vehicle cases, there are several combinations to deal with the takeoff and landing sequence. Suppose each vehicle serves one target per flight; equation (3.35) estimates the number of possible combinations, where *n* is the number of targets within the vehicle's operation range.

$$c = \frac{(2n)!}{2^n n!} \tag{3.35}$$

Note that when the number of target points increases, the total combinations also increase exponentially. Figure 3.8. illustrates the exponential growth in the number of possible combinations with respect to the number of targets. Due to the problem's complexity, this research focuses only on two vehicles. Note that the two targets are within the vehicle's operational range, or the two targets are within one group.

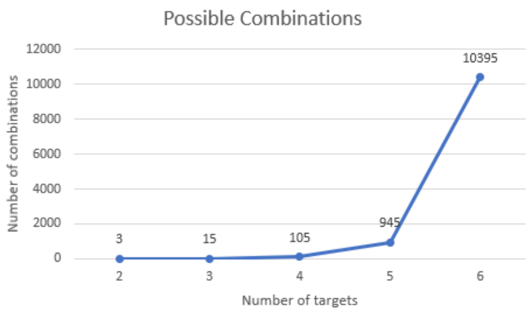


Figure 3.8 Possible combinations as a function of *n*

3.4.1. Single-target-per-flight scenario (TVST)

To model the case of two vehicles, we consider all possible takeoff and landing combinations for both vehicles. Suppose there are two vehicles A and B, and there are two targets to be visited, one target for each vehicle. There are three possible takeoff and landing sequences, as shown in Figure 3.9.

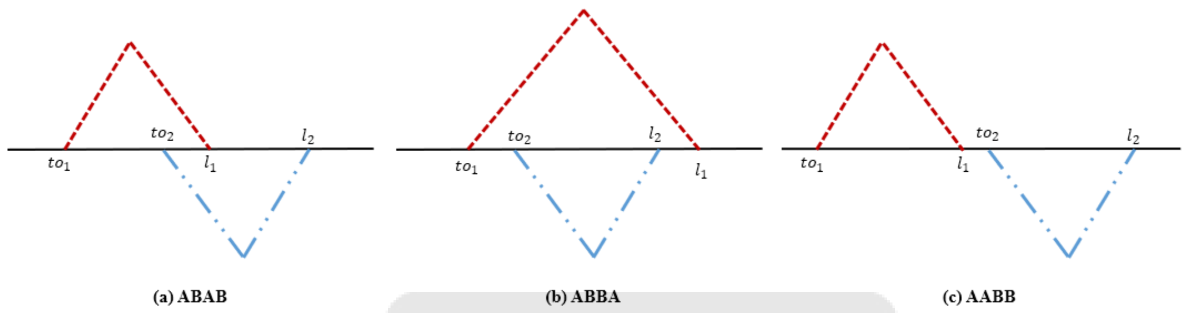


Figure 3.9 Three possible scenarios

The first scenario is when vehicle A takeoffs, followed by vehicle B. Then, vehicle A is retrieved, followed by vehicle B. This scenario is shown in Figure 3.9 (a), and we call it the ABAB case. The second scenario is when vehicle A takeoffs, followed by vehicle B. Then, the carrier retrieved vehicle B, followed by vehicle A. This scenario is shown in Figure 3.9 (b), and we call it the ABBA case. The last scenario is when both vehicles are taking off and then retrieved, respectively. Since the vehicles are identical, remember that this scenario is similar to a single vehicle with a single-target-per-flight scenario. Please note that this scenario only happens when we assume that the two vehicles are identical. If the number of vehicles increases, the number of combinations also increases. The model here requires a target visiting sequence and the vehicle's takeoff and landing order. The algorithm will be used to know precisely which scenario applies for calculation.

1. *Mathematics model*

To solve the first stage of CVTSP for two vehicles, we first treat the problem as a single-vehicle model. Thus, the notation is the same. Below is the formulation to determine the vehicle's exact location to takeoff and land.

The objective function minimizes the makespan or carrier travel time to finish the entire mission, as shown in equation (3.36).

$$\text{Minimize} \sum_{i=0}^{n+1} \sum_{j=0,\, j\neq i}^{n+1} tc_{ij} x_{ij} + \sum_{i=1}^{n} ts_i + \sum_{i=1}^{n} wt_i \qquad (3.36)$$

Equation (3.37) is related to the carrier's time when the vehicle and carrier meet each other.

$$\| l_i - to_j \| \leq tc_{ij} \quad \forall i, j = 1,\dots,n,\ i \neq j \qquad (3.37)$$

Equation (3.38) relates to carrier time when vehicle and carrier are separated (when the vehicle is deployed).

$$\| to_i - l_i \| \leq ts_i \quad \forall i = 1,\dots,n \qquad (3.38)$$

Equations (3.39) – (3.40) show the relationship between distance traveled by vehicle and distance from takeoff to_i and landing l_i to the target s_i.

$$\| s_i - to_i \| \leq dOut_i \quad \forall i = 1,\dots,n \qquad (3.39)$$

$$\| s_i - l_i \| \leq dIn_i \quad \forall i = 1,\dots,n \qquad (3.40)$$

Equations (3.41) – (3.42) ensure no violation of the vehicle's operational range when deployed.

$$\left(dOut_i + dIn_i \right) / V_v \leq ts_i + wt_i \quad \forall i = 1,\dots,n \qquad (3.41)$$

$$ts_i + wt_i \leq R \quad \forall i = 1,\dots,n \qquad (3.42)$$

2. *The formulation for each scenario*

Since the takeoff and land sequence are different for each scenario, the formulation is distinct. This research divides the carrier route into segments to build the formulation, and we will explain further in each scenario. The notations and parameters are like the ones provided in Section 3.3.2. However, there are some adjustments in variable ts_i and wt_i. This case ts_i means time spent by a carrier to travel in segment *i* and wt_i means time spent by a carrier to wait in segment *i*. The waiting time is associated with the landing segment. Then, the target sequence is also fixed.

All of the scenarios have the same objective function and several constraints, which will be written below.

The objective function (3.43) is the carrier's time to finish the entire mission or makespan.

$$\text{Minimize } \sum_{i=1}^{3} ts_i + tc_0 + tc_{n+2} + \sum_{i=1}^{3} wt_i \tag{3.43}$$

Equations (3.44) – (3.45) show the relationship between distance traveled by vehicle and distance from takeoff to_i and landing l_i to target s_i.

$$\| s_i - to_i \| \le dOut_i \quad \forall i = 1,\dots,n \tag{3.44}$$

$$\| s_i - l_i \| \le dIn_i \quad \forall i = 1,\dots,n \tag{3.45}$$

Equation (3.46) represents the segment where vehicles and carriers travel together before the first vehicle takeoff. Other constraints related to the segment for each scenario are written in the latter part.

$$\| l_0 - to_1 \| \le tc_{01} \tag{3.46}$$

a. Scenario ABAB

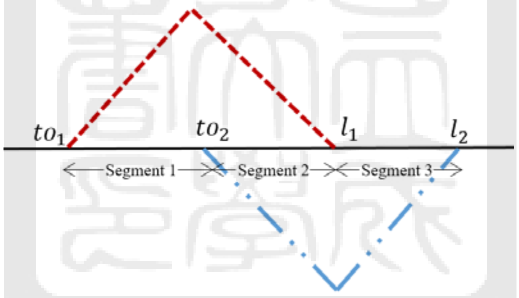


Figure 3.10 Segments for scenario ABAB

$$\| to_1 - to_2 \| \le ts_1 \tag{3.47}$$

$$\| to_2 - l_1 \| \le ts_2 \tag{3.48}$$

$$\| l_1 - l_2 \| \le ts_3 \tag{3.49}$$

$$\| l_2 - to_3 \| \le tc_{23} \tag{3.50}$$

$$\left(dOut_1 + dIn_1 \right) / V_v \le ts_1 + ts_2 + wt_2 \tag{3.51}$$

$$\left(dOut_2 + dIn_2 \right) / V_v \le ts_2 + ts_3 + wt_3 \tag{3.52}$$

$$ts_1 + ts_2 + wt_2 \leq R \tag{3.53}$$

$$ts_2 + ts_3 + wt_3 \leq R \tag{3.54}$$

b. Scenario ABBA

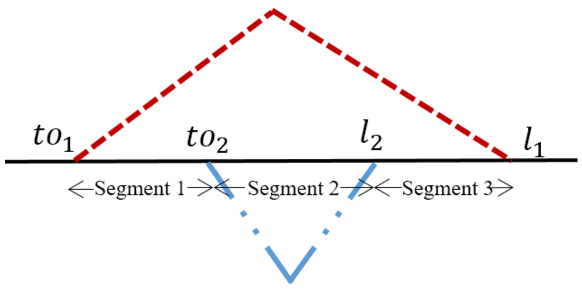


Figure 3.11 Segments for scenario ABBA

$$\| to_1 - to_2 \| \leq ts_1 \tag{3.55}$$

$$\| to_2 - l_2 \| \leq ts_2 \tag{3.56}$$

$$\| l_2 - l_1 \| \leq ts_3 \tag{3.57}$$

$$\| l_1 - to_3 \| \leq tc_{13} \tag{3.58}$$

$$\left(dOut_1 + dIn_1\right)/V_v \leq ts_1 + ts_2 + ts_3 + wt_2 \tag{3.59}$$

$$\left(dOut_2 + dIn_2\right)/V_v \leq ts_2 + wt_3 \tag{3.60}$$

$$ts_1 + ts_2 + ts_3 + wt_2 \leq R \tag{3.61}$$

$$ts_2 + wt_3 \leq R \tag{3.62}$$

c. Scenario AABB

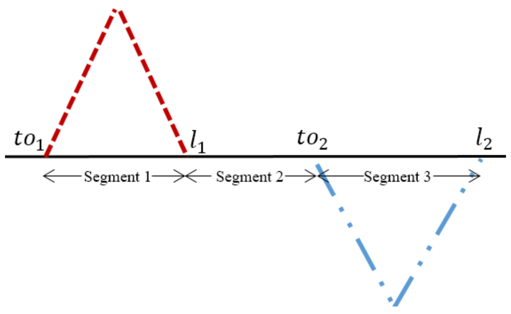


Figure 3.12 Segments for scenario AABB

$$\| to_1 - to_2 \| \le ts_1 \qquad (3.63)$$

$$\| to_2 - l_2 \| \le ts_2 \qquad (3.64)$$

$$\| l_2 - l_1 \| \le ts_3 \qquad (3.65)$$

$$\| l_1 - to_3 \| \le tc_{13} \qquad (3.66)$$

$$\left(dOut_1 + dIn_1\right)/V_v \le ts_1 + ts_2 + ts_3 + wt_2 \qquad (3.67)$$

$$\left(dOut_2 + dIn_2\right)/V_v \le ts_2 + wt_3 \qquad (3.68)$$

$$ts_1 + ts_2 + ts_3 + wt_2 \le R \qquad (3.69)$$

$$ts_2 + wt_3 \le R \qquad (3.70)$$

All the additional constraints for each scenario have the same function. Equation (3.47)-(3.49), (3.55)-(3.57), and (3.63)-(3.65) are related to the segments between each vehicle's takeoff and landing. Equations (3.50), (3.58), and (3.66) deal with the last segment after the second vehicle is retrieved. Equation (3.51)-(3.52), (3.59)-(3.60), and (3.67)-(3.68) ensure that the carrier travel time and waiting time is at least the vehicle travel time when deployed. Equations (3.53)-(3.54), (3.61)-(3.62), and (3.69)-(3.70) ensure the operational vehicle range is satisfied when the vehicle is deployed. The best alternative is obtained by enumerating all scenarios and determine the best alternative. The best scenario will be plugged in to the primary CVTSP model.

3.4.2. Multiple-target-per-flight scenario (TVMT)

For the mathematical programming model in the target compositions case, there are some limitations. Our model requires to know a target visiting sequence, takeoff and landing sequence for each vehicle, and target assignment before it starts. We estimate these prerequisites using some heuristics.

Like subsection 3.3.2, the mathematical programming model for this case, a variable named $dIntra_i$ needs to be added to represent the distance between targets. Before running the model, preprocessing the length of each subtour needs to be conducted where it will be used as a constraint. The complete mathematical programming model is written as follows.

$$\text{Minimize } \sum_{i=1}^{3} ts_i + tc_0 + tc_{n+2} + \sum_{i=1}^{3} wt_i \quad (3.71)$$

$$\| s_i - to_i \| \le dOut_i \quad \forall i = 1,\dots,n \quad (3.72)$$

$$\| s_i - l_i \| \le dIn_i \quad \forall i = 1,\dots,n \quad (3.73)$$

$$\| l_0 - to_1 \| \le tc_{01} \quad (3.74)$$

$$len(ST_i) \le dIntra_i \quad \forall i = 1,2 \quad (3.75)$$

Scenario ABAB

$$\| to_1 - to_2 \| \le ts_1 \quad (3.76)$$

$$\| to_2 - l_1 \| \le ts_2 \quad (3.77)$$

$$\| l_1 - l_2 \| \le ts_3 \quad (3.78)$$

$$\| l_2 - to_3 \| \le tc_{23} \quad (3.79)$$

$$(dOut_1 + dIntra_1 + dIn_1)/V_v \le ts_1 + ts_2 + wt_2 \quad (3.80)$$

$$(dOut_2 + dIntra_2 + dIn_2)/V_v \le ts_2 + ts_3 + wt_3 \quad (3.81)$$

$$ts_1 + ts_2 + wt_2 \le R \quad (3.82)$$

$$ts_2 + ts_3 + wt_3 \le R \quad (3.83)$$

Scenario ABBA

$$\| to_1 - to_2 \| \le ts_1 \quad (3.84)$$

$$\| to_2 - l_2 \| \le ts_2 \quad (3.85)$$

$$\| l_2 - l_1 \| \le ts_3 \quad (3.86)$$

$$\| l_1 - to_3 \| \le tc_{13} \tag{3.87}$$

$$\left(dOut_1 + dIntra_1 + dIn_1\right)/V_v \le ts_1 + ts_2 + ts_3 + wt_2 \tag{3.88}$$

$$\left(dOut_2 + dIntra_2 + dIn_2\right)/V_v \le ts_2 + wt_3 \tag{3.89}$$

$$ts_1 + ts_2 + ts_3 + wt_2 \le R \tag{3.90}$$

$$ts_2 + wt_3 \le R \tag{3.91}$$

Scenario AABB

$$\| to_1 - to_2 \| \le ts_1 \tag{3.92}$$

$$\| to_2 - l_2 \| \le ts_2 \tag{3.93}$$

$$\| l_2 - l_1 \| \le ts_3 \tag{3.94}$$

$$\| l_1 - to_3 \| \le tc_{13} \tag{3.95}$$

$$\left(dOut_1 + dIntra_1 + dIn_1\right)/V_v \le ts_1 + ts_2 + ts_3 + wt_2 \tag{3.96}$$

$$\left(dOut_2 + dIntra_2 + dIn_2\right)/V_v \le ts_2 + wt_3 \tag{3.97}$$

$$ts_1 + ts_2 + ts_3 + wt_2 \le R \tag{3.98}$$

$$ts_2 + wt_3 \le R \tag{3.99}$$

The objective function (3.71) is makespan, the carrier's time to finish the entire mission. Equations (3.72) – (3.73) show the relationship between distance traveled by vehicle and distance from takeoff to_i and landing l_i to the target s_i. Equation (3.74) represents the distance traveled to move from starting point to the first takeoff point. New equation (3.75) ensures that when a vehicle flies between targets, it will not violate the total distance between targets. Equation (3.76) – (3.83), (3.84) – (3.91), and (3.92) – (3.99) constraints the vehicle and ship movement for scenario ABAB, ABBA, and AABB, respectively.

Chapter 4

DESIGN OF EFFICIENT HEURISTIC ALGORITHMS

Even though a commercial solver can generate an optimal solution, it takes a long time to solve larger instances. Furthermore, it is not guaranteed that the commercial solver can find an optimal or feasible solution within a considerable amount of time. Therefore, developing a heuristic and algorithm to shorten the computational time is necessary, especially to deal with large instances.

This chapter explains the procedures for the heuristics and algorithms developed to solve the CVTSP problem. Section 4.1 describes the heuristic for single-target-per-flight cases and Section 4.2. describes the heuristic for multi-target-per-flight cases.

4.1. A 2-stage Heuristic for the Single-Target-per-Flight Cases (ST)

To solve the single-target-per-flight cases for visiting *n* targets, determining the best target visiting sequence is the first step. The mathematical model mentioned in Chapter 3 can find the best target visiting sequence by implementing a TSP-like formulation using subtour elimination constraints. However, as the number of targets increases, the computational time also increases significantly. Thus, a simulated annealing algorithm is implemented to determine the best target visiting sequence.

The simulated annealing algorithm produces the target visiting sequence used as an input to calculate the exact takeoff and landing position. To determine the exact takeoff and landing coordinates, SOCP will be applied.

4.1.1. Heuristic Framework

The steps to solve the 2-stage heuristics are illustrated in Figure 4.1.

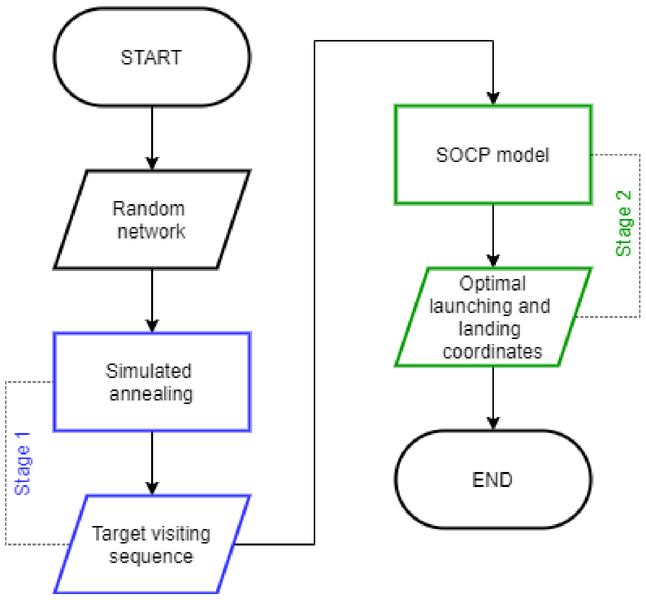


Figure 4.1 2-stage Heuristic Framework

4.1.2. A Simulated Annealing Heuristic Algorithm for Target Sequence

The simulated annealing (SA) algorithm is a local search heuristic that can escape a local optima by accepting a worse solution with a specific probability function. This subsection discusses the proposed SA in detail to determine the target visiting sequence, including the solution representation, parameter setting, and the proposed SA procedure.

1. *Solution representation*

Since the simulated annealing in this research determines the target visiting sequence, the solution representation is relatively straightforward. This step aims at finding a sequence that results in the least travel time possible to pass target locations according to the specified ordering. The simulated annealing algorithm will try to find the best sequence. An example of the SA solution is illustrated in Figure 4.2. The SA solution is represented by a permutation of n targets denoted by the set $\{1, 2, 3, \ldots, n\}$. Figure 4.3 illustrates the SA solution representation of the example.

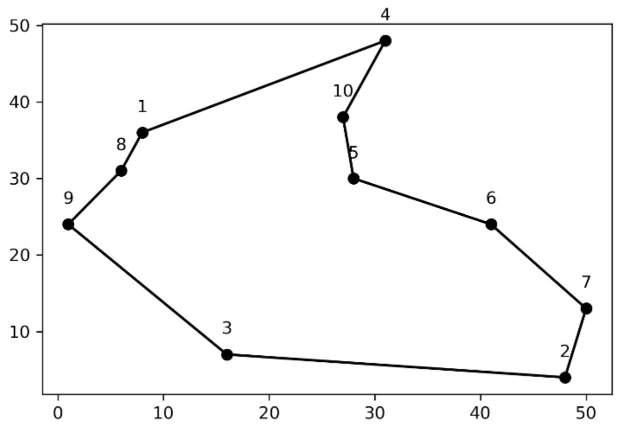


Figure 4.2 An example of SA solution

1	**4**	**10**	**5**	**6**	**7**	**2**	**3**	**9**	**8**

Figure 4.3 SA solution representation

2. Parameter used

The ALG-SA uses three parameters, T_{init}, T_F , and $S_{\max}$. T_{init} represents the initial temperature and T_F represents the final temperature, indicating when the SA procedure is stopped. This SA algorithm implements a linear cooling schedule to explore most of the solution space. $S_{\max}$ denotes the maximum step for the ALG-SA to control the linear cooling schedule.

3. SA Procedure

The algorithm starts by setting the parameters and then randomly generates an initial solution x_0. The best objective function is denoted by $f\left(x_0\right)$. For every iteration, a new solution is generated based on the current solution. The objective values of both solutions are evaluated. If $f\left(x_0+\Delta x\right)< f\left(x_0\right)$, the new solution will replace the current best solution. Otherwise, the new solution will replace the current best solution with a probability that satisfies $r > \exp(\Delta f \, / \, kT)$ for a random number *r*. The current temperature T_k is decreased based on linear cooling, where *k* indicates the current step.

$$T_k = T_F + \left(T_{init} - T_F\right)\left(\frac{S_{\max} - k}{S_{\max}}\right) \qquad (4.1)$$

This algorithm will terminate if one of the two terminating conditions is reached: (1) when the current temperature T_k is equal to T_F and (2) when current step *k* equals $S_{\max}$. The ALG-SA pseudocode is shown in Table 4.1.

Table 4.1 Simulated Annealing Algorithm Procedure

Simulated Annealing Algorithm (ALG-SA)

Input Data: target nodes
Initialization: Initial temperature T_{init}, target temperature T_F, maximum step $S_{\max}$

```
1: while $T_k > T_F$ or $k < S_{\max}$ :
2:     for number of new solution
3:         select a new solution: $x_0 + \Delta x$
4:         if $f\left(x_0 + \Delta x\right) < f\left(x_0\right)$ then
5:             $f_{new} = f\left(x_0 + \Delta x\right); x_0 = x_0 + \Delta x$
6:         else
7:             $\Delta f = f\left(x_0 + \Delta x\right) - f\left(x_0\right)$
8:             random $r$ (0,1)
9:             if $r > \exp(\Delta f / T)$ then
10:                $f_{new} = f\left(x_0 + \Delta x\right); x_0 = x_0 + \Delta x$
11:            else
12:                $f_{new} = f\left(x_0\right)$
13.:           end if
14:        end if
15:    end for
16:    $f = f_{new}$; $k = k + 1$
17:    Decrease temperature $T_k$ based on linear cooling schedule
18: end while
```

Result: Optimal target visiting sequence

The result obtained from the simulated annealing algorithm will be used as an input in SOCP (as mentioned in Chapter 3) for a further calculation to determine the exact takeoff and landing point.

4.1.3. An Illustrative Example

This subsection provides an illustrative example of how the ALG-SA and SOCP determine the individual takeoff and landing coordinates.

1. *Input and parameter*

In this illustrative example, the inputs are:

- Targets: (27, 30), (13, 29), (31, 17), (41, 10), (2, 33), (31, 20)
- Starting point: (36, 0)
- Ending point: (36, 2)

There are three parameters for ALG-SA:

- T_{init} (initial temperature) = 15
- T_F (final temperature) = 0.0000001
- $S_{\max}$ (maximum step) = 5,000

2. *ALG-SA result*

To see the improvement from each iteration in simulated annealing, *current energy,* and *best energy* are recorded in every step. The *energy* here means the total distance to pass through those targets following a certain visiting order. The lower the energy, the better since the objective is to minimize the distance traveled.

From ALG-SA, the following result is obtained:

- Target visiting sequence: (2, 33), (13, 29), (27, 30), (31, 20), (31, 17), (41, 10)
- Best energy (distance): 122.53

Figure 4.4. represent the recorded current energy, recorded the best energy, and the illustration of simulated annealing result.

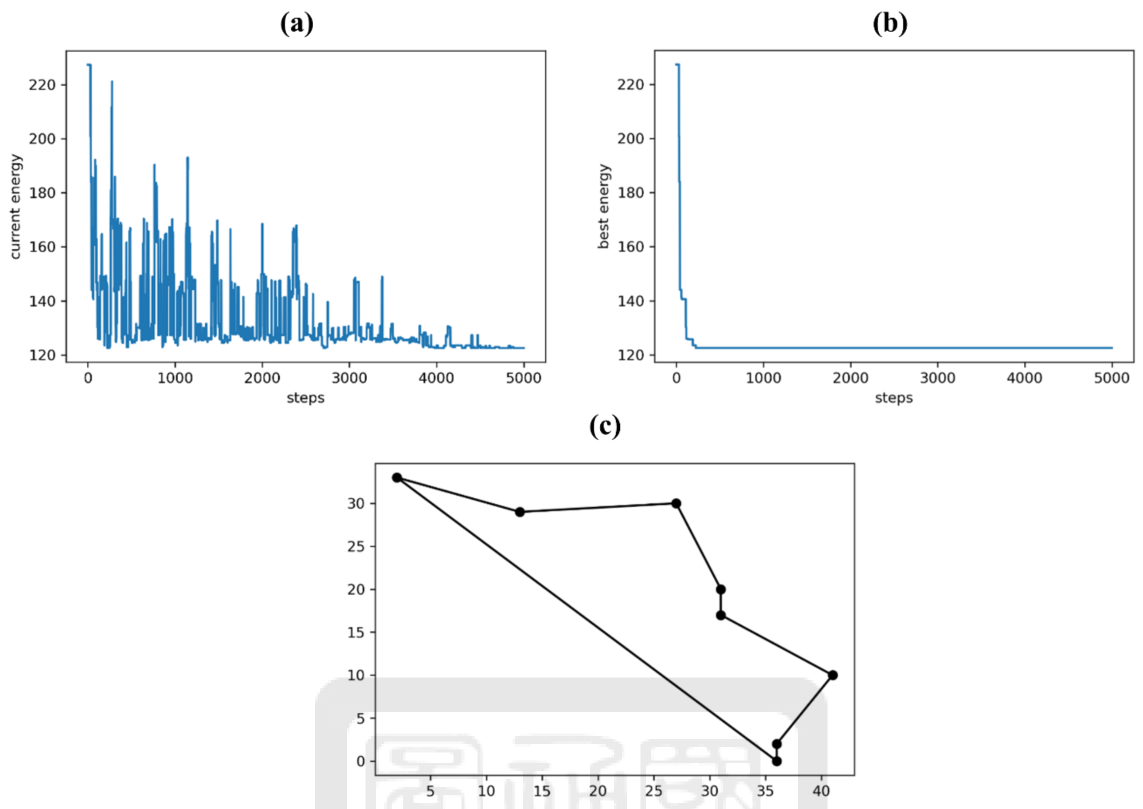


Figure 4.4 (a) SA current energy, (b) SA best energy, and (c) SA result

3. *SOCP Result*

After the result is obtained from ALG-SA, the converged target visiting sequence is used to solve the SOCP further. In this step, the goal is to find the individual takeoff and landing coordinates. In this example, the vehicle endurance (time unit) is set to be R = 10. After solving SOCP, the following result is obtained:

- Makespan: 85.77 time unit
- Individual takeoff and landing coordinates:
  - Takeoff and Landing Point 1: (8.77, 26.43), (11.93, 29.39)
  - Takeoff and Landing Point 2: (15.31, 29.53), (21.64, 29.78)
  - Takeoff and Landing Point 3: (23.60, 27.74), (28.33, 22.80)
  - Takeoff and Landing Point 4: (29.11, 21.11), (30.45, 18.20)
  - Takeoff and Landing Point 5: (32.12, 16.90), (36.26, 13.68)

- Takeoff and Landing Point 6: (36.21, 11.46), (36.06, 4.95)

Figure 4.5 illustrates the result that was obtained from this stage.

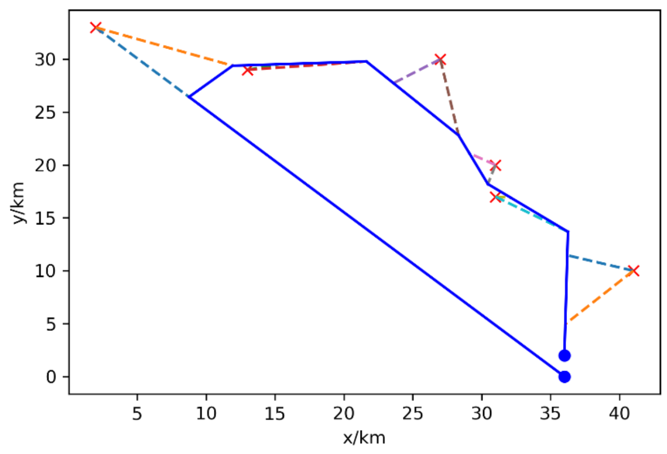


Figure 4.5 SOCP result

4.2. A 4-stage Heuristic for the Multiple-Targets-per-Flight Cases (MT)

When the targets are close to each other, and the vehicle's endurance is sufficient, the vehicle may visit multiple targets consecutively without returning the carrier unless necessary. This research defines this problem as a multiple-targets-per-flight case. However, determining which target belong to which cluster is another set of problem. To solve this problem, a clustering mechanism is used based on the vehicle's endurance.

The clustering mechanism needs the target visiting sequence as an input obtained from simulated annealing. This research implements the Shortest-path-based algorithm for every cluster formed with multiple targets to determine the target composition.

It is essential to solve the multi-target-per-flight (hereafter referred to as a target composition), ensuring that the vehicle can travel through the targets without violating its range constraint. Therefore, it is necessary to have an algorithm to determine the target composition considering the vehicle's range constraint.

4.2.1. Heuristic Framework

To determine the target composition and its exact takeoff and landing position, we first treat it as a single-target-per-flight problem to find its individual takeoff and landing coordinates. Thus, this research uses the ALG-SA algorithm that has been described in the previous section. The result from ALG-SA will be further used to determine the target composition. The steps to solve the target composition problem is summarized by the following steps:

1. ***Stage 1 – Determine target visiting sequence and its takeoff/landing coordinates***

   The given network will be treated as a single-target-per-flight case. A simulated annealing algorithm (ALG-SA) is implemented to determine the target visiting sequence. ALG-SA results are used as an input to know the exact takeoff and landing coordinates using SOCP. Furthermore, the target visiting sequence is also used as an input to assign each target to its corresponding cluster.

2. ***Stage 2 – Determine cluster***

   The targets close to each other will be considered a cluster and later will be served with either one vehicle or two vehicles. A vehicle's endurance is used to differentiate between targets classified as one cluster and isolated from the other.

3. ***Stage 3 – Determine target composition***

   Knowing the target visiting sequence and the clusters are insufficient for the target composition scenario. Each cluster must know which targets can be visited by a vehicle in one flight (without violating the range constraint). In this stage, a procedure is developed based on the shortest path algorithm to determine the target composition. This stage uses stage 1 (the individual takeoff and landing coordinates) and stage 2 (target assignment to its corresponding cluster) as an input.

4. ***Stage 4 – Adjust takeoff/landing coordinates for each group of multiple targets***

   The result from stage 3 is based on the individual takeoff and landing coordinates. However, it is more suitable to use takeoff and landing coordinates for each target composition instead of individual takeoff and landing coordinates in this stage.

The flowchart for this step is illustrated in Figure 4.6.

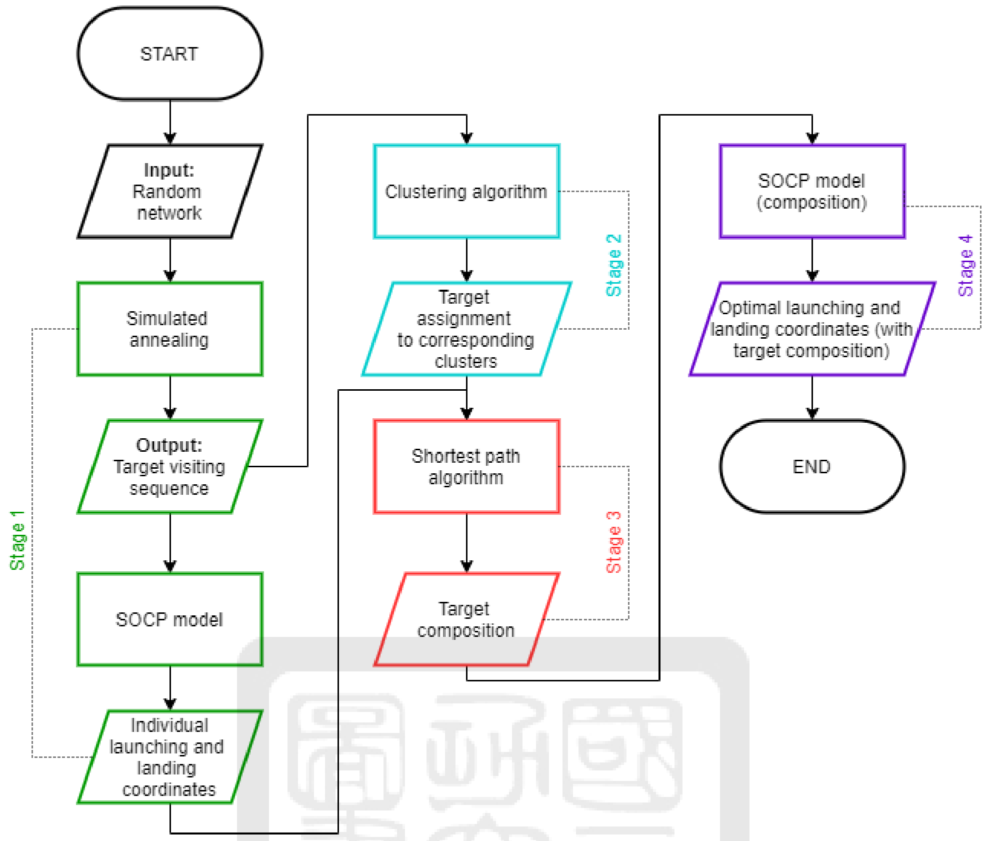

START
Input: Random network
Simulated annealing
Output: Target visiting sequence
SOCP model
Individual launching and landing coordinates
Stage 1
Clustering algorithm
Target assignment to corresponding clusters
Stage 2
Shortest path algorithm
Target composition
Stage 3
SOCP model (composition)
Optimal launching and landing coordinates (with target composition)
Stage 4
END

Figure 4.6 Target composition flowchart

4.2.2. Clustering algorithm

The input of the multiple-target-per-flight scenario is the random network. Since this is a multiple-target case, each target needs to be assigned to a corresponding cluster. A mechanism based on the vehicle’s endurance is implemented to assign each target.

Based on the target visiting sequence obtained from simulated annealing, every target and its consecutive target are assessed to see whether it violates the vehicle's endurance or not. Suppose the time needed to travel from a target and its consecutive target obeys the vehicle's range constraint. In that case, those targets will be considered as a cluster. Table 4.2 shows the procedure for the clustering algorithm.

Table 4.2 Clustering Algorithm Procedure

Clustering Algorithm Procedure

```
Input Data: target nodes, target visiting sequence, vehicle endurance
1: while $i$ < target length:
2:         append target[$i$] to cluster_list
3:         remaining endurance = vehicle endurance – distance(target[$i$], target[$i+1$])
4:         if remaining endurance ≥ 0:
5:                 append target[$i+1$] to cluster_list
6:                 $i = i + 1$
7:         end if
8:         $i = i + 1$
9: end while
```

Result: Target assignment to its corresponding cluster

Based on the flight network obtained from the previous section, the following target visiting sequence in the illustrative example in Section 4.1 can be calculated as follows:

- Starting point: Starting point: (36, 0)
- Ending point: (36, 2)
- Target visiting sequence: (2, 33), (13, 29), (27, 30), (31, 20), (31, 17), (41, 10)

The travel distance between consecutive target visits is evaluated to ensure the vehicle's range constraint is satisfied. For example, the vehicle range is five time units. Since the speed of the vehicle is 2, then the vehicle range is ten distance units. The calculation for every target and its consecutive target is presented in Table 4.3.

Table 4.3 Distance between each target and its consecutive target

Target *i*	**Target *i*+1**	**Distance**
(2, 33)	(13, 29)	11.70
(13, 29)	(27, 30)	14.04
(27, 30)	(31, 20)	10.77
(31, 20)	(31, 17)	3.00
(31, 17)	(41, 10)	12.21

Based on Table 4.3, the only pair of targets that have a distance within the vehicle endurance are located at (31, 20) and (31, 17). This means these targets can be considered as a cluster. Therefore, in total, there are five clusters. Cluster 1 is (2,33), cluster 2 is (13,29), cluster 3 is (27, 30), cluster 4 are (31, 20) and (31, 17), and cluster 5 is (41, 10).

4.2.3. A Shortest-path-based Target Composition Heuristic Algorithm (ALG-SP)

To determine the target composition, this research develops an algorithm based on the Dijkstra algorithm. Dijkstra algorithm was first proposed in 1956 by Edsger Dijkstra and published in 1959. Dijkstra algorithm is recognized as one of the best algorithms when all the weight number is $W_{ij} \geq 0$. The proposed shortest-path algorithm (ALG-SP) is discussed to determine the target composition, including the solution representation, parameter setting, and the proposed SP procedure.

The ALG-SP objective is to minimize the distance traveled by a carrier from a certain point to another. Since the objective is the carrier's travel distance, each target's individual takeoff and landing are used as an input. Every possible path is assessed to see whether the target node is feasible or not for the vehicle to travel.

Algorithm adjustment is conducted to consider the vehicle endurance for each possible path. During the initialization, we remove the connections that violate vehicle endurance constraints. Therefore, the algorithm only considers feasible paths for both vehicle and carrier when determining the shortest path's target composition.

1. *Graph initialization*

As an input for the Dijkstra algorithm, removing infeasible edges from the graph is necessary. This subsection explains the basic idea of removing infeasible edges based on vehicle endurance.

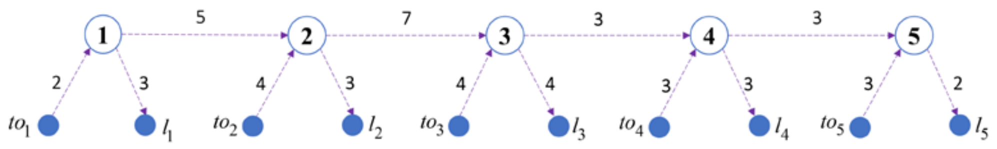


Figure 4.7 Graph initialization

Figure 4.7 illustrates the graph initialization. Suppose there are five targets with their corresponding takeoff and landing coordinates obtained from the ALG-SA. A white circle represents a target, and a blue circle represents its corresponding takeoff or landing coordinate. The dashed line represents a possible flight edge for a vehicle, and its length is displayed besides. Since a vehicle would be carried by the carrier when its endurance is insufficient, the carrier's (solid) arc also needs to be considered. Figure 4.8 illustrates the entire network for both vehicle and carrier.

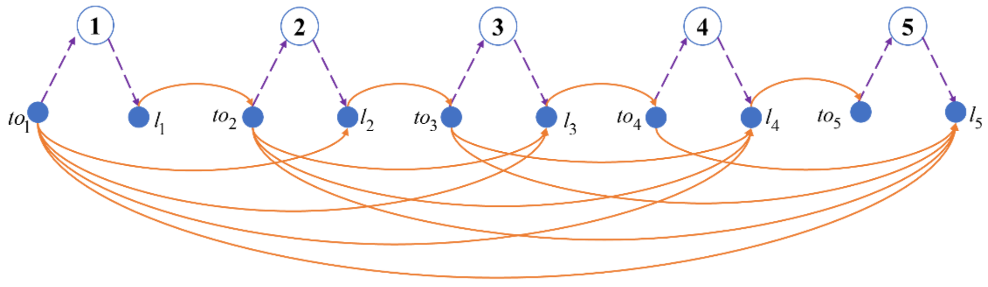


Figure 4.8 Network for vehicle and carrier

For each arc in the network, we preprocess the arcs by removing all of the infeasible arcs. Suppose the vehicle endurance (in distance unit) is 12. Start the calculation from target 1 takeoff point (to_1). After being launched from to_1, a vehicle will fly to target 1 t_1 with a distance equal to 2. We then check whether it is possible for target composition and continue to target 2 (t_2). In this case, the distance becomes 2+5 = 7, which is still less than the vehicle endurance. Despite knowing that the vehicle can fly from target 1 to target 2, it is important to ensure that the carrier retrieves the vehicle at landing point 2 (l_2). Then, we calculate the distance from t_2 to l_2, the distance becomes 2+5+3 = 10, which still satisfies the vehicle endurance constraint. Thus, we add target [1,2] as the feasible connection.

Adding more targets in the composition means adding the next target since it has to be sequential. Next, we consider target 3 (t_3). Adding target 3, we consider the total distance to its landing coordinates (l_3). However, by including target 3 into the composition, the total distance will be infeasible. Since continuing to t_3 is not possible, the calculation should not continue to t_4 as well. Thus, the subsequent composition would start from t_3. Continuing the same procedure, there are three total connections: [1, 2], [3, 4], and [3, 5]. In this case, [3, 5] means that the carrier will directly move from to_3 to l_5 implying the vehicle visits target 3 – 4 – 5. The final network after preprocessing is illustrated in Figure 4.9.

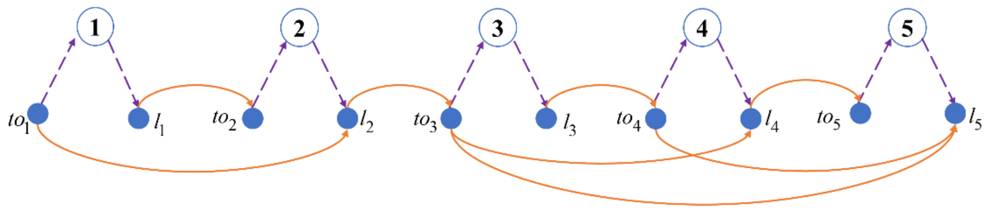


Figure 4.9 Network after preprocessing

To understand how to create a graph, Table 4.4 explains the procedure to initialize the graph.

Table 4.4 Graph Initialization Procedure

Graph Initialization Procedure

```
Input Data: target nodes, individual takeoff and landing coordinates, vehicle endurance
1: for each $i$ in target length:
2:         if $i$ < target length:
3:                 $j = i + 1$
4:         end if
5:         while $j \leq$ target length:
6:                 $k = 0$
7:                 while $k$ < target length – 1:
8:                         between target distance += distance(target[k], target[k+1])
9:                         $k = k + 1$
10:                end while
11:                total distance = distance(takeoff[i], target[i]) + between target distance +
                   distance(target[j], land[j])
12:                vehicle remains = vehicle endurance – total distance
13:                if vehicle remains $\geq 0$:
14:                        add connection [$i$, $j$] // composition
15:                end if
15:        end while
16:        total distance = distance(takeoff[i], target[i]) + distance(target[j], land[j])
17:        vehicle remains = vehicle endurance – total distance
18:        if vehicle remains $\geq 0$:
19:                add connection [$i$, $i$] // single target
20:        end if
21: end for
Result: Graph with feasible connections
```

2. *Proposed SP procedure*

After obtaining the graph consist of feasible connections, we continue the procedure to find a shortest path. Table 4.5 illustrates the procedure of creating a graph to determine the Dijkstra algorithm's target composition.

Table 4.5 Shortest-path-based Algorithm Procedure

Shortest Path Algorithm (ALG-SP)

```
Input Data: Graph, origin, destination
1: for each vertex v in Graph:
2:         dist[v] = infinity
3:         previous[v] = undefined
4: end for
5: dist[origin] = 0
6: Q = set of all nodes in Graph
7: while Q is not empty:
8: u = node is Q with the smallest distance
9: remove u from Q
10:        for each neighbor v of u:
11:                alt = dist[u] + distance between(u,v)
12:                if alt < dist[v]:
13:                        dist[v] = alt
14:                        previous[v] = u
15:        end for
16: end while
Result: Shortest path for target composition
```

The result from the algorithm will be used as an input to calculate the exact takeoff and landing coordinates using SOCP.

4.2.4. An Illustrative Example

Since this problem is the continuation of the previous stage, the previous example is used. From the previous example, the following result is obtained.

- Sequence: (36, 0), (2, 33), (13, 29), (27, 30), (31, 20), (31, 17), (41, 10), (36, 2)
- Makespan: 85.77 time unit

- Individual takeoff and landing coordinates:
  - Takeoff and Landing Point 1: (8.77, 26.43), (11.93, 29.39)
  - Takeoff and Landing Point 2: (15.31, 29.53), (21.64, 29.78)
  - Takeoff and Landing Point 3: (23.60, 27.74), (28.33, 22.80)
  - Takeoff and Landing Point 4: (29.11, 21.11), (30.45, 18.20)
  - Takeoff and Landing Point 5: (32.12, 16.90), (36.26, 13.68)
  - Takeoff and Landing Point 6: (36.21, 11.46), (36.06, 4.95)

With vehicle endurance *R* equal to 10, the connection are: [(1, 1), (2, 2), (3, 3), (3, 4), (4, 4), (4, 5), (4, 6), (5, 5), (5, 6), (6, 6)]. By solving a shortest path using the Dijkstra algorithm, the composition is: [1, 2, 3, 4-6], or if viewed as a target point: [(36, 0), (2, 33), (13, 29), (27, 30), **[(31, 20), (31, 17), (41, 10)]**, (36, 2)].

The target composition will be used as an input to determine the exact takeoff and landing points for the composition using the SOCP formulation. After solving the SOCP, the following result is obtained.

- Makespan: 78.80 time unit
- Individual takeoff and landing coordinates:
  - Takeoff and Landing Point 1 – (2, 33): (22.38, 13.21), (12.90, 29.04)
  - Takeoff and Landing Point 2 – (13, 29): (14.51, 29.15), (17.95, 29.38)
  - Takeoff and Landing Point 3 – (27, 30): (19.43, 28.32), (26.46 , 23.27)
  - Takeoff and Landing Point 4, 5, 6 (composition) – (31, 20), (31, 17), (41, 10): (27.61, 21.68), (36.21, 4.88)

Chapter 5
COMPUTATIONAL EXPERIMENTS

This chapter explains the computational experiments to analyze the performance of the proposed models and algorithms. In section 5.1, we present the technical specifications for computational testing. Section 5.2 describes the randomly generated cases to assess our models and algorithms. The experiment results are analyzed in section 5.3 and summarized in section 5.4.

5.1. Technical Specifications

The computational experiments are conducted on a personal computer. The specification for the PC and the software is shown in Table 5.1.

Table 5.1 PC and software specifications

PC specifications	
Operating System	Microsoft Windows 10 1x64 based
Processor	Intel i7-8700, 3.2GHz
RAM	8.00 GB
Software specifications	
Python	Version 3.6.5
Anaconda	Version 4.9.2
Gurobi	Version 9.0.0

5.2. Settings for Experimental Cases

Randomly generated scenarios are used to test the models and the heuristics for evaluating their performance. The discussion is divided into single-vehicle and two-vehicle cases. For every subsection, the experiments are conducted in a different problem setting. Each case has a different number of targets, ranging from small instances to large instances. For each case, there are five subcases with varying configurations of network. The computational time for the mathematical model is limited to be 7,200 and 1,800 seconds. Thus, if the computational time exceeds the limit, the calculation is terminated. The current solution was determined to be feasible but not optimal.

This research compares our model and algorithm with other literature with similar problem settings to assess the model's performance and algorithm. However, since the objective function is different, other literature's objective function is modified by adding a variable wt_i that represents the time spent by the carrier to wait when the vehicle travels from takeoff point to_i to landing point l_i.

5.2.1. Waiting Time Variable Implications

Most research related to the carrier-vehicle traveling salesman problem (CVTSP) assumes that the carrier will always move from point to point. However, this research assumes that the carrier can wait to visit the target when the vehicle is deployed. By adding waiting time to the model, this research allows the carrier to wait for the vehicle.

By taking the waiting time into account, the carrier's total travel distance decreases. This happens because the carrier will take a shorter route and wait until the vehicle returns. Thus, the carrier can reduce the unnecessary movement when moving around and when the vehicle is deployed. This research conducts several experiments on the two-vehicle scenario to show the implication of waiting time on the total carrier travel distance. Table 5.2 shows the implications of the waiting time on three selected scenarios. An illustration shows how the route changes when introducing the waiting time to the model for each scenario. Figure 5.1, Figure 5.2, and Figure 5.3 illustrate scenarios 1, 2, and 3, respectively. Based on the result, it shows that the waiting time does shorten the carrier travel distance.

Table 5.2 Waiting Time Implications on Model

Scenario	**Carrier travel distance (Without waiting time)**	**Carrier travel distance (With waiting time)**
1	33.03	27.56
2	23.39	21.58
3	19.61	18.51

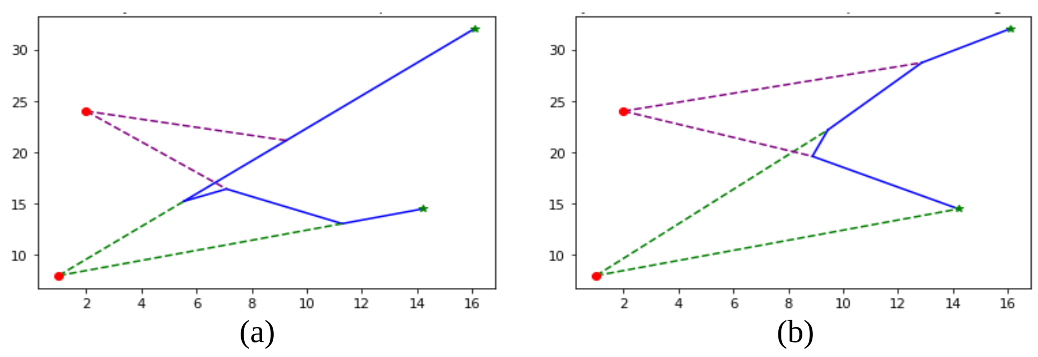


Figure 5.1 Scenario 1 (a) Without waiting time (b) With waiting time

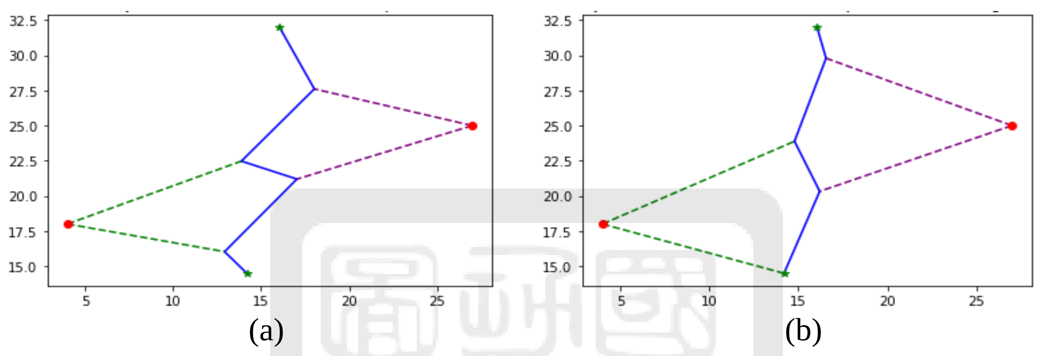


Figure 5.2 Scenario 2 (a) Without waiting time (b) With waiting time

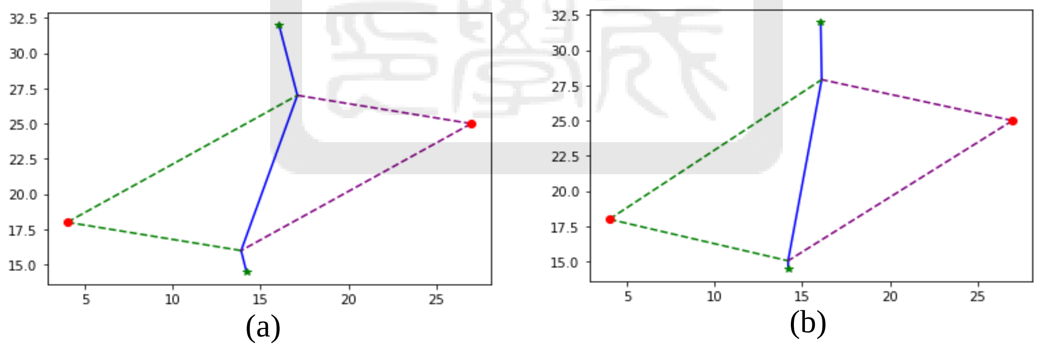


Figure 5.3 Scenario 3 (a) Without waiting time (b) With waiting time

5.2.2. Settings for Single Vehicle Case

For the single-vehicle case, this research aims at comparing the formulation proposed by other researchers. For the carrier-vehicle traveling salesman problem, this research's model is compared to Gambella et al. (2018), and Klaučo et al. (2014) since their study uses an exact model to determine the takeoff landing coordinate. For further comparison, this research adopts the traveling salesman problem (TSP) formulation proposed by de Andrade (2016) to compare the performance of the target visiting sequence.

For the single-vehicle single target (SVST) scenario, we compare with Gambella et al. (2018), which used an assignment variable to solve CVTSP. On the other hand, this research uses subtour elimination techniques. The different approaches between Gambella et al. (2018) and this research will be assessed in the objective value and the computational time.

However, there are some modifications required to compare the two models. The model provided by Gambella et al. (2018) requires the specification of starting and ending points of the tour. Thus, this research modifies the model by stating the starting and ending point to make it comparable. Furthermore, this research adds a new waiting time variable to the Gambella et al. (2018).

For the single-vehicle-multiple-target cases (SVMT), we will compare with Klaučo et al. (2014). However, Klaučo et al. (2014) assume that the target visiting sequence is known. Thus, this research modifies the input and the model to determine the target visiting sequence before the calculation. Similar to SVST, a new waiting time variable will be added to Klaučo et al. (2014).

This research limits the experiment to a few targets to ensure that the model always gives an optimal solution. This research will have several subcases with the same number of targets for each case but different network configurations.

5.2.3. Settings for Two Vehicles Case

To the best of our knowledge, Chen et al. (2020) is the only literature related to the carrier-vehicle traveling salesman problem (CVTSP) that deals with two vehicles.

However, Chen et al. (2020) solve the takeoff and landing coordinates using particle swarm optimization (PSO). Thus, it is unknown whether the result is optimal or not. To ensure that the benchmark is optimal, this research uses a total enumeration on a small case network as a comparison.

The total enumeration will be conducted for both single-target and multiple-target cases. Each test case will be calculated on three different takeoffs and landing sequences (ABAB, ABBA, and AABB). Based on the result, it can be determined which scenario will provide the best objective value. Then, the result from total enumeration can be compared with this research's algorithm.

For the multiple-target cases, the total enumeration will be conducted to determine the best target composition. The result from the total enumeration will be compared with the algorithm to see whether they result in the same target composition.

5.3. Experiment Results

The experiments were conducted with the designed settings over random networks. The experiment results from the experiment are written in the following subsections.

5.3.1. Single-vehicle model (SV)

For the single-vehicle single-carrier scenario, the test cases are the same for the single-target-per-flight scenario and the multiple-target-per-flight scenario or target composition. The difference only lies in the performance of different solution methods: algorithms and the second-order cone programming (SOCP) model.

1. Single-target-per-flight scenario (SVST)

This research's model and algorithm are compared several mathematical models to verify and assess the performance. In total, there are four models used for comparison:

1) Gambella et al. (2018) formulation

   To solve the carrier-vehicle traveling salesman problem (CVTSP), they use an assignment variable. Furthermore, they also introduce a ranking-based algorithm to improve computational performance.

2) This research with the MTZ formulation (with and without objective function linearization)

As mentioned in Chapter 3, the original formulation proposed by this research is MTZ formulation with the quadratic objective function. This chapter will compare MTZ with quadratic objective function and linear objective function by introducing a new variable.

3) The primal-dual-based formulation by de Andrade (2016)

de Andrade (2016) proposed three methods regarding the traveling salesman problem: a spanning tree polytope-based formulation, primal-dual-based formulation, and a Steiner TSP formulation. Based on the paper, the primal-dual-based formulation and the STSP-based formulation perform better.

For primal-dual-based formulation, the basic idea of this formulation is dual associate variables π with some constraints. Thus, a feasible solution can be obtained by linking the primal and dual variables in the same constraints and avoiding cycles.

4) de Andrade (2016) Steiner TSP (STSP) based formulation

This formulation is a flow-based model for the elementary paths of minimum cost. The main idea is sending n + 1 units of flow from source node s to the remaining nodes that must be visited, including the destination node.

The mathematical model for each formulation will be written here to explain the model better and used as a comparison.

1) Gambella et al. (2018) formulation

$$\text{Minimize} \sum_{i=1}^{n} t_i + \sum_{i=1}^{n+1} T_i \tag{5.1}$$

$$\| Q_i - p_{to,i} \| \leq V_v t_{i,1} \quad \forall i = 1,\dots,n \tag{5.2}$$

$$\| Q_i - p_{l,i} \| \leq V_v t_{i,2} \quad \forall i = 1,\dots,n \tag{5.3}$$

$$\| p_{to,i} - p_{l,i} \| \leq V_c t_i \quad \forall i = 1,\dots,n \tag{5.4}$$

$$\| p_o - p_{to,1} \| \leq V_c T_1 \tag{5.5}$$

$$\| p_{l,i-1} - p_{to,i} \| \leq V_c T_i \quad \forall i = 2,...,n \tag{5. 6}$$

$$\| p_f - p_{l,n} \| \leq V_c T_{n+1} \tag{5. 7}$$

$$t_{i,1} + t_{i,2} \leq t_i \quad \forall\ i = 1,...,n \tag{5. 8}$$

$$Q_i = \sum_{j=1}^{n} w_{i,j} q_j \quad \forall\ i = 1,...,n \tag{5. 9}$$

$$\sum_{j=1}^{n} w_{i,j} = 1 \quad \forall\ i = 1,...,n \tag{5. 10}$$

$$\sum_{i=1}^{n} w_{i,j} = 1 \quad \forall\ j = 1,...,n \tag{5. 11}$$

$$t_{i,1} \geq 0 \quad \forall\ i = 1,...,n \tag{5. 12}$$

$$t_{i,2} \geq 0 \quad \forall\ i = 1,...,n \tag{5. 13}$$

$$0 \leq t_i \leq a \quad \forall\ i = 1,...,n \tag{5. 14}$$

$$T_i \geq 0 \quad \forall\ i = 1,...,n+1 \tag{5. 15}$$

$$q_{\min} \leq Q_i \leq q_{\max} \quad \forall\ i = 1,...,n \tag{5. 16}$$

$$w_{ij} \in \{0,1\} \quad \forall i, j = 1,...,n \tag{5. 17}$$

The starting point and ending point are represented by p_o and p_f respectively, and their values are given. In Gambella et al. (2018), one of the important decisions is the target visiting sequence, expressed by the assignment variables. The objective function (5.1) is the mission completion time for the carrier to move from the starting point to the ending point. Constraints (5.2) – (5.8) are related to the second-order cone programming to determine the exact takeoff and landing point. The assignment variables are implied in constraints (5.9) – (5.11). Constraints (5.12) – (5.17) are the bounds for the variables.

2) This research using the MTZ formulation (without objective function linearization)

$$\text{Minimize} \sum_{i=0}^{n+1} \sum_{j=0,j\neq i}^{n+1} tc_{ij} x_{ij} + \sum_{i=1}^{n} ts_i + \sum_{i=1}^{n} wt_i \tag{5. 18}$$

$$\sum_{i=0,i\neq j}^{n+1} x_{ij} = 1 \quad \forall\ j = 0,...,n+1 \tag{5. 19}$$

$$\sum_{j=0, i\neq j}^{n+1} x_{ij} = 1 \quad \forall\ i = 0, \ldots, n+1 \tag{5. 20}$$

$$u_i - u_j + (n+2) x_{ij} \leq n+1 \quad \forall\ 1 \leq i, j \leq n+1,\ i \neq j \tag{5. 21}$$

$$u_i \leq n+1 \quad \forall\ 1 \leq i \leq n+1 \tag{5. 22}$$

$$\| s_i - to_i \| \ \leq\ dOut_i \quad \forall i = 1, \ldots, n \tag{5. 23}$$

$$\| s_i - l_i \| \ \leq\ dIn_i \quad \forall i = 1, \ldots, n \tag{5. 24}$$

$$\| to_i - l_i \| \ \leq\ ts_i \quad \forall i = 1, \ldots, n \tag{5. 25}$$

$$\| p_0 - to_i \| \ \leq tc_{1i} \quad \forall i = 1, \ldots, n \tag{5. 26}$$

$$\| l_i - to_j \| \ \leq\ tc_{ij} \quad \forall i, j = 1, \ldots, n,\ i \neq j \tag{5. 27}$$

$$\| p_f - l_i \| \ \leq tc_{i,f} \quad \forall i = 1, \ldots, n \tag{5. 28}$$

$$\left(dOut_i + dIn_i \right) / V_v \ \leq\ ts_i + wt_i \quad \forall i = 1, \ldots, n \tag{5. 29}$$

$$x_{ij} \in \{0,1\} \quad \forall i, j = 0, \ldots, n+1 \tag{5. 30}$$

$$u_i \in Z \quad \forall i, j = 1, \ldots, n+1 \tag{5. 31}$$

The starting point and the ending point are represented by p_o and p_f, respectively. The objective function (5.18) represents the travel time of the mission. The objective function here is quadratic. This research has transformed the objective function to be linear in Chapter 3 by introducing a new variable. Constraints (5.19) – (5.20) related to the target visiting sequence. Constraints (5.21) – (5.22) are to avoid subtour within the path. Constraints (5.23) – (5.29) are the second-order cone constraints to determine the exact takeoff and landing coordinates. Lastly, constraints (5.30) – (5.31) are the bounds for the variables.

3) de Andrade (2016) primal-dual-based formulation

$$\text{Minimize} \sum_{i=0}^{n+1} \sum_{j=0, j\neq i}^{n+1} \hat{tc}_{ij} \tag{5. 32}$$

$$\sum_{i|iv\in A} x_{iv} - \sum_{j|vj\in A} x_{vj} = \begin{matrix} -1, \text{ if } v = o \\ 1, \text{ if } v = f \\ 0, \text{ otherwise} \end{matrix} \quad \forall v \in V \tag{5. 33}$$

$$x_{oj} + \sum_{i \in P} x_{ij} = 1, \ \forall \ j \in P \tag{5. 34}$$

$$x_{if} + \sum_{k \in P} x_{ik} = 1, \ \forall \ i \in P \tag{5. 35}$$

$$\pi_j - \pi_i \le tc_{ij} + M\left(1 - x_{ij}\right), \forall ij \in A \tag{5. 36}$$

$$\pi_j - \pi_i \ge tc_{ij} - M\left(1 - x_{ij}\right), \forall ij \in A \tag{5. 37}$$

$$\pi_s = 0 \tag{5. 38}$$

$$Lx_{ij} \le \hat{tc}_{ij} \le Ux_{ij} \tag{5. 39}$$

$$L\left(1 - x_{ij}\right) \le tc_{ij} - \hat{tc}_{ij} \le U\left(1 - x_{ij}\right) \tag{5. 40}$$

$$\| s_i - to_i \| \ \le \ dOut_i \quad \forall i = 1, ..., n \tag{5. 41}$$

$$\| s_i - l_i \| \ \le \ dIn_i \quad \forall i = 1, ..., n \tag{5. 42}$$

$$\| to_i - l_i \| \ \le \ ts_i \quad \forall i = 1, ..., n \tag{5. 43}$$

$$\| p_0 - to_i \| \ \le tc_{1i} \quad \forall i = 1, ..., n \tag{5. 44}$$

$$\| l_i - to_j \| \ \le \ tc_{ij} \quad \forall i, j = 1, ..., n, \ i \ne j \tag{5. 45}$$

$$\| p_f - l_i \| \ \le tc_{i,f} \quad \forall i = 1, ..., n \tag{5. 46}$$

$$\left(dOut_i + dIn_i\right) / V_v \ \le \ ts_i + wt_i \ \ \forall i = 1, ..., n \tag{5. 47}$$

$$x_{ij} \in \{0,1\} \quad \forall i, j = 0, ..., n+1 \tag{5. 48}$$

$$\pi \ge 0 \tag{5. 49}$$

The starting point and the ending point are represented by p_o and p_f, respectively. The dual variable π is introduced since the base of the formulation is primal-dual. De Andrade (2016) uses a quadratic objective function, but this research transforms it to be linear by introducing a new variable. Equation (5.32) shows the linearized objective function. Constraints (5.33) – (5.35) are related to the target visiting sequence. The dual variable is included in constraints (5.36) – (5.38). Constraints (5.39) – (5.40) are related to the new variable to linearize the objective function. Constraints (5.41) – (5.47) are the second-order cone constraints to determine the exact takeoff and landing coordinates. Finally, constraints (5.48) – (5.49) are the bounds for the variables.

4) de Andrade (2016) Steiner TSP (STSP) based formulation

$$\text{Minimize} \sum_{i=0}^{n+1} \sum_{j=0,\, j\neq i}^{n+1} \overset{\wedge}{tc}_{ij} \tag{5. 50}$$

$$\sum_{i|iv\in A} x_{iv} - \sum_{j|vj\in A} x_{vj} = \begin{matrix} -1, \text{ if } v=o \\ 1, \text{ if } v=f \\ 0, \text{ otherwise} \end{matrix} \quad \forall v\in V \tag{5. 51}$$

$$x_{oj} + \sum_{i\in P} x_{ij} = 1, \ \forall\ j\in P \tag{5. 52}$$

$$x_{if} + \sum_{k\in P} x_{ik} = 1, \ \forall\ i\in P \tag{5. 53}$$

$$\sum_{i|io\in A} x_{is} = 0 \tag{5. 54}$$

$$\sum_{j|fj\in A} x_{fj} = 0 \tag{5. 55}$$

$$\sum_{i|iv\in A} f_{iv} - \sum_{j|vj\in A} f_{vj} = \begin{matrix} -|P|-1, \text{ if } v=o \\ 1, \text{ if } v\in P+\{f\} \end{matrix} \tag{5. 56}$$

$$x_{ij} \le f_{ij} \le \left(|P|+1\right) x_{ij}, \forall ij\in A \tag{5. 57}$$

$$Lx_{ij} \le \overset{\wedge}{tc}_{ij} \le Ux_{ij} \tag{5. 58}$$

$$L\left(1-x_{ij}\right) \le tc_{ij} - \overset{\wedge}{tc}_{ij} \le U\left(1-x_{ij}\right) \tag{5. 59}$$

$$\| s_i - to_i \| \ \le\ dOut_i \quad \forall i=1,\dots,n \tag{5. 60}$$

$$\| s_i - l_i \| \ \le\ dIn_i \quad \forall i=1,\dots,n \tag{5. 61}$$

$$\| to_i - l_i \| \ \le\ ts_i \quad \forall i=1,\dots,n \tag{5. 62}$$

$$\| p_0 - to_i \| \ \le tc_{1i} \quad \forall i=1,\dots,n \tag{5. 63}$$

$$\| l_i - to_j \| \ \le\ tc_{ij} \quad \forall i,j=1,\dots,n,\ i\neq j \tag{5. 64}$$

$$\| p_f - l_i \| \ \le tc_{i,f} \quad \forall i=1,\dots,n \tag{5. 65}$$

$$\left(dOut_i + dIn_i\right)/V_v \ \le\ ts_i + wt_i \quad \forall i=1,\dots,n \tag{5. 66}$$

$$x_{ij} \in \{0,1\} \quad \forall i,j=0,\dots,n+1 \tag{5. 67}$$

$f \geq 0$ (5. 68)

For the STSP-based formulation, the main concept is sending 'flow' f from the starting point to all the target nodes and the ending point. The starting point and ending point are represented by p_o and p_f, respectively. This research linearizes the objective function (5.50) by introducing a new variable similar to the previous primal-dual formulation. Constraints (5.51) – (5.53) are related to the target visiting sequence. Constraints (5.54) – (5.55) ensure no flow coming to the starting point and leaving from the ending point. Constraints (5.56) – (5.57) are related to the flow variable. Constraints (5.58) – (5.59) are related to the new variable to linearize the objective function. Constraints (5.60) – (5.66) are the second-order cone constraints to determine the exact takeoff and landing coordinates. Finally, constraints (5.67) – (5.68) are the bounds for the variables.

The experiments are conducted on random networks of different sizes (n equal to 4, 8, 12, and 18 targets). For each network size, there are ten random subcases of different network configurations. Table 5.3 shows the computational time, the gap provided by the commercial solver, and the optimality gap. The optimality gap is the difference between the best objective value calculated by the model (either feasible or optimal) and the optimal objective value. In this case, Gambella et al. (2018) give the optimal objective value.

Table 5.3 Performance Comparison

Name	Opt Obj Val	Method													
		Gambella		MTZ-Linearized			MTZ-Original			Primal-dual			STSP-based		
		Comp Time (s)	Solver Gap (%)	Comp Time (s)	Solver Gap (%)	Opt Gap (%)	Comp Time (s)	Solver Gap (%)	Opt Gap (%)	Comp Time (s)	Solver Gap (%)	Opt Gap (%)	Comp Time (s)	Solver Gap (%)	Opt Gap (%)
n_4s_1	40.8	0.21	0	0.15	0	0	0.34	0	0	0.31	0	0	0.35	0	0
n_4s_2	38.7	0.16	0	0.09	0	0	0.33	0	0	0.21	0	0	0.26	0	0
n_4s_3	65.6	0.24	0	0.09	0	0	0.31	0	0	0.21	0	0	0.23	0	0
n_4s_4	66.4	0.2	0	0.1	0	0	0.43	0	0	0.33	0	0	0.27	0	0
n_4s_5	60.5	0.15	0	0.09	0	0	0.32	0	0	0.3	0	0	0.29	0	0
n_4s_6	55.0	0.2	0	0.23	0	0	0.53	0	0	0.21	0	0	0.25	0	0
n_4s_7	80.5	0.25	0	0.29	0	0	0.36	0	0	0.42	0	0	0.26	0	0
n_4s_8	59.3	0.17	0	0.26	0	0	0.41	0	0	0.24	0	0	0.21	0	0
n_4s_9	45.1	0.18	0	0.17	0	0	0.32	0	0	0.17	0	0	0.23	0	0
n_4s_1	70.0	0.18	0	0.23	0	0	0.23	0	0	0.27	0	0	0.23	0	0

Name	Opt Obj Val	Method													
		Gambella		MTZ-Linearized			MTZ-Original			Primal-dual			STSP-based		
		Comp Time (s)	Solver Gap (%)	Comp Time (s)	Solver Gap (%)	Opt Gap (%)	Comp Time (s)	Solver Gap (%)	Opt Gap (%)	Comp Time (s)	Solver Gap (%)	Opt Gap (%)	Comp Time (s)	Solver Gap (%)	Opt Gap (%)
n_8s_1	63.6	0.91	0	107.9	0	0	20.86	0	0	7.73	0	0	62.47	0	0
n_8s_2	71.5	0.41	0	31.77	0	0	38.14	0	0	33.79	0	0	43.58	0	0
n_8s_3	67.6	0.61	0	36.97	0	0	54.93	0	0	132.5	0	0	9.82	0	0
n_8s_4	68.0	0.5	0	45.3	0	0	29.58	0	0	17.67	0	0	71.06	0	0
n_8s_5	61.2	0.58	0	2.51	0	0	5.65	0	0	3.29	0	0	4.16	0	0
n_8s_6	62.6	0.64	0	43.79	0	0	26.61	0	0	24.05	0	0	40.68	0	0
n_8s_7	82.9	0.97	0	8.75	0	0	12.26	0	0	61.08	0	0	76.04	0	0
n_8s_8	74.1	1.02	0	28.32	0	0	15.84	0	0	21.3	0	0	37.54	0	0
n_8s_9	61.7	0.45	0	32.21	0	0	8.04	0	0	66.22	0	0	33.62	0	0
n_8s_1	74.0	0.8	0	30.84	0	0	7.45	0	0	32.78	0	0	4.51	0	0
$n_{12}s$	76.1	15.45	0	7200	36.0	3.6	7200	65.6	4.5	7200	13.4	1.3	7200	42.9	6.3
$n_{12}s$	72.3	12.52	0	7201	28.9	0	7200	61.6	2.8	55	0	0	5596	0	0
$n_{12}s$	67.7	9.59	0	2129	0	0	7200	55.0	1.5	115	0	0	7200	39.9	0.5
$n_{12}s$	71.9	16.51	0	5434	0	0	7200	42.5	1.5	7200	9.4	2.2	7200	14.0	0.0
$n_{12}s$	61.6	8.37	0	3957	0	0	5236	0	0	39	0	0	2913	0	0
$n_{12}s$	69.0	70.65	0	7200	37.8	0	7200	48	0.0	188	0	0	7200	34	0
$n_{12}s$	86.2	14.62	0	7201	29.3	0.9	7200	52	0.0	1513	0	0	7200	36	0.9
$n_{12}s$	78.5	50.96	0	2079	0.0	0	7201	47	0.0	176	0	0	7202	37	0.2
$n_{12}s$	72.0	15.39	0	7200	22.6	0.3	7200	38	1.4	153	0	0	7201	30	0.6
$n_{12}s$	82.0	78.67	0	7201	32.7	0.0	7200	51	0.0	240	0	0	7201	39	1.1
$n_{16}s$	88.5	7200	3.6	7201	69.4	2.4	7200	84.8	5.2	7201	55.6	1.5	7201	70.4	2.4
$n_{16}s$	81.6	3039	0	7200	61.6	4.1	7200	80.0	10.0	7201	48.9	4.4	7201	56.1	1.8
$n_{16}s$	81.5	131	0	7201	67.9	1.0	7200	88.2	21.7	7201	37.4	2.3	7201	70.4	1.5
$n_{16}s$	83.7	1394	0	7201	53.7	3.7	7200	84.5	4.3	7201	57.9	3.3	7201	69.1	3.2
$n_{16}s$	67.8	4692	0	7200	57.9	0.1	7200	82.0	8.2	7201	47.7	0.1	7201	59.3	0.1
$n_{16}s$	79.0	2167	0	7201	66.0	0.4	7200	81.8	6.7	7201	21.0	0.0	7201	62.4	1.5
$n_{16}s$	86.9	1283	0	7201	59.8	0.0	7200	87.1	25.4	7200	54.5	1.2	7201	58.2	2.3
$n_{16}s$	84.5	297	0	7201	70.4	0.8	7200	83.7	11.8	7201	50.1	1.8	7201	62.0	5.4
$n_{16}s$	77.0	714	0	7200	66.1	0.6	7201	86.2	13.0	7200	29.2	0.2	7201	65.6	3.3
$n_{16}s$	82.0	2362	0	7201	66.0	1.1	7200	83.8	19.5	7201	55.6	1.4	7201	68.8	0.3

Based on the experiments above, Gambella et al. (2018) perform better than the other models. Up to 16 targets, the formulation provided by Gambella et al. (2018) was able to find an optimal solution in less than 7,200 seconds, except for problem $n_{16}s_1$. For that specific problem, there is still a 3.6% gap given by the commercial solver. However, the objective value is still smaller than the rest of the model. Therefore, the formulation provided by Gambella et al. (2018) is used as a benchmark.

Table 5.4 shows the summary of the experiment based on the number of targets.

Table 5.4 Summary of Performance Experiment

n	Gambella			MTZ-Linear			MTZ-Original			Primal-Dual			STSP		
	Avg Comp Time (s)	Comp Time Std Dev	Solver Gap (%)	Avg Comp Time (s)	Comp Time Std Dev	Opt Gap (%)	Avg Comp Time (s)	Comp Time Std Dev	Opt Gap (%)	Avg Comp Time (s)	Comp Time Std Dev	Opt Gap (%)	Avg Comp Time (s)	Comp Time Std Dev	Opt Gap (%)
4	0.23	0.12	0	0.38	0.71	0	0.84	1.60	0	0.54	0.91	0	0.61	1.18	0
8	0.69	0.22	0	36.83	28.46	0	21.94	15.75	0	40.04	38.39	0	38.35	26.37	0
12	29.27	26.85	0	5680	2172.5	0.8	7004	621.26	1.3	1688	2937	1.7	6611	1394	1.4
16	2328	2195.6	0.4	7201	0.17	1.4	7204	0.16	12.6	7201	0.16	1.6	7201	0.14	2.2

From Table 5.4, we observe that our formulation, the MTZ with the linearized objective function, performs better than the primal-dual, the STSP-based, and the MTZ with the quadratic objective function formulations. For the rest of the computational results, the order of the performance is primal-dual, STSP-based, and MTZ with the quadratic objective function, respectively.

A further experiment was conducted on small networks (*n* equal to 3, 5, 7, and 9 targets) to compare the algorithm provided by Gambella et al. (2018) and the algorithm proposed in this research. Similar to Gambella et al. (2018), this research's model and algorithm need to determine the target visiting sequence and the corresponding takeoff and landing coordinates. Table 5.5 shows the comparison in terms of the objective function (makespan) and computational time.

Table 5.5 Comparison with Gambella et al. (2018)

n	Gambella et al. (2018)			This research's model			This research's algorithm		
	Avg Make-span	Avg Comp Time	Comp Time St Dev	Avg Make-span	Avg Comp Time	Comp Time St Dev	Avg Make-span	Avg Comp Time	Comp Time St Dev
3	41.706	0.112	0.031	41.708	0.085	0.031	41.708	0.047	0.005
5	59.610	0.170	0.009	59.614	0.188	0.034	59.614	0.054	0.003
7	65.142	0.631	0.266	65.146	0.843	0.328	65.146	0.060	0.001
9	63.530	3.046	0.352	63.533	17.228	0.000	63.533	0.073	0.003

Table 5.5 concludes that this research model gives the same objective value as Gambella et al. (2018). In terms of computational time, this research's algorithm is the shortest. It gives the same objective value as this research's model.

To analyze the performance of the model and algorithms further, three parameters are used; computational time, optimality gap, and whether the SOCP can find optimal makespan or not. The experimental results for all test cases are summarized in Table 5.6. From the results, GUROBI can find an optimal solution for cases up to nine targets. Meanwhile, GUROBI can find an optimal solution for two out of five subcases with eleven targets. For cases with more than eleven targets, GUROBI could not find an optimal solution within 1,800 seconds. To calculate the ALG-SA computational time, we include the computational time for simulated annealing that calculates a good target visiting sequence and the computational time of using SOCP to calculate the exact takeoff and landing point. In all test cases, ALG-SA can solve the problem in less than 1 second. For further analysis, the optimality gap for GUROBI and ALG-SA are provided. The GUROBI optimality gap measures the gap between the objective values calculated by its best feasible solution and its best linear relaxation lower bound. Meanwhile, to calculate the optimality gap for ALG-SA, we compare the results obtained from GUROBI and ALG-SA. The formulation for the ALG-SA optimality gap is as follows:

$$OptGap(\%) = \left| \left((SA_{result} - GUROBI_{result}) / GUROBI_{result} \right) 100\% \right| \qquad (5.67)$$

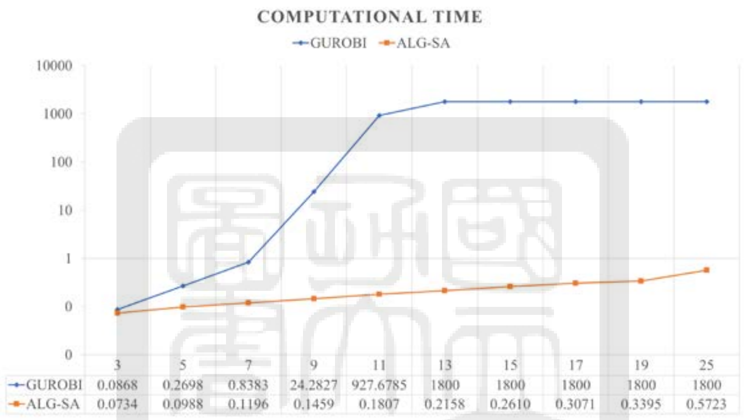
Table 5.6 Experiment results of the model with GUROBI and ALG-SA for SVST

***n* target(s)**	**GUROBI**				**ALG-SA**			
	Avg optimal makespan (time unit)	**Avg comp. time (sec)**	**Stdev comp. time**	**Opt. gap (%)**	**Avg optimal makespan (time unit)**	**Avg comp. time (sec)**	**Stdev comp. time**	**Opt. gap (%)**
3	67.87	0.09	0.01	0.00%	67.87	0.07	0.01	0.00%
5	96.15	0.27	0.12	0.00%	96.15	0.10	0.00	0.00%
7	104.46	0.84	0.21	0.00%	104.46	0.12	0.01	0.00%
9	115.19	24.28	6.89	0.00%	115.30	0.15	0.00	0.11%
11	126.58	927.68	594.83	8.23%	134.71	0.18	0.02	5.85%
13	135.54	1,800	0	71.08%	139.93	0.22	0.01	9.95%
15	148.49	1,800	0	85.32%	143.53	0.26	0.02	5.09%
17	N/A*	1,800	0	N/A	179.10	0.31	0.03	N/A
19	N/A*	1,800	0	N/A	170.79	0.34	0.02	N/A
25	N/A*	1,800	0	N/A	219.59	0.57	0.03	N/A

Figure 5.4 illustrates the difference in computational time between GUROBI and ALG-SA for small instances. ALG-SA was able to find an optimal solution in less than 1 second for small instances. We also test cases of larger sizes, with *n* equal to 30, 50, and 70. The result for the large instances is presented in Table 5.7.

Table 5.7 Experiment results for single target large instances

***n* targets**	**Optimal Makespan (time unit)**	**Mean (s)**	**Standard Deviation (s)**
30	1247.05	0.87	0.07
50	1468.84	9.98	3.29
70	3509.34	23.40	1.23



	3	5	7	9	11	13	15	17	19	25
GUROBI	0.0868	0.2698	0.8383	24.2827	927.6785	1800	1800	1800	1800	1800
ALG-SA	0.0734	0.0988	0.1196	0.1459	0.1807	0.2158	0.2610	0.3071	0.3395	0.5723

Figure 5.4 Comparison of computational time between GUROBI and ALG-SA

2. Multiple-target-per-flight scenario (SVMT)

This research compares with Klaučo et al. (2014) to verify the SVMT model. Unlike the SVST cases, to solve SVMT cases, Klaučo et al. (2014) require a target visiting sequence to be given. Therefore, the simulated annealing algorithm cannot be compared. Table 5.8 shows the comparison between this research's model and the model provided by Klaučo et al. (2014).

Table 5.8 Comparison with Klaučo et al. (2014)

n	**Klaučo et al. (2014)**			**This research's model**		
	Avg Makespan	**Avg Comp Time**	**Comp Time St Dev**	**Avg Makespan**	**Avg Comp Time**	**Comp Time St Dev**
3	51.827	0.078	0.028	51.848	0.022	0.001
5	73.505	0.535	0.248	73.527	0.097	0.012
7	81.722	0.725	0.141	81.753	0.100	0.018
9	86.155	0.316	0.049	86.183	0.124	0.003

Based on the result provided in Table 5.8, this research's model can give a similar result with Klaučo et al. (2014). Furthermore, based on the computational time, this research's model can obtain an optimal value faster than Klaučo et al. (2014).

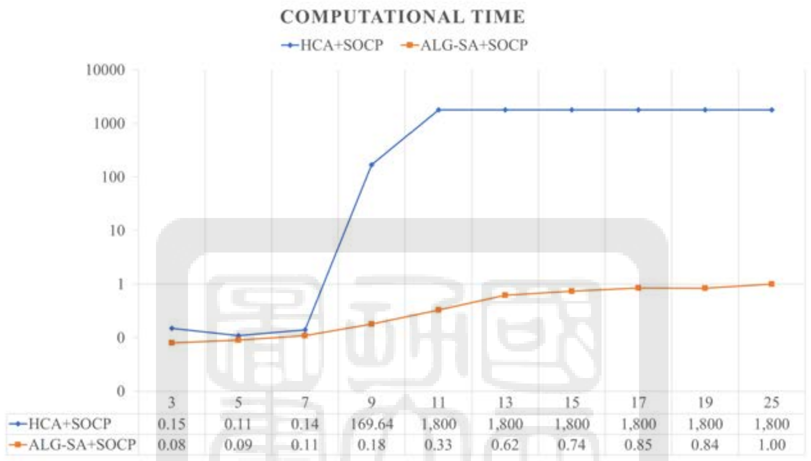
Since the model and algorithm are verified, this research conducted further experiments to see the model's performance in computational time and optimality gap. The input for the mathematical programming model requires an algorithm to determine which target belongs to which cluster. A hierarchical clustering analysis (HCA) assigns the target to its corresponding cluster. The hierarchical clustering analysis (HCA) is a clustering method. HCA is chosen to compare the performance since it is unnecessary to specify the number of clusters before the calculation, unlike the *k*-means method. Furthermore, HCA does not require the target visiting sequence. Thus, HCA calculates a target-cluster assignment, and then the SOCP will solve the target visiting sequence.

For the SVMT case, this research assessed the performance using a simulated annealing algorithm (ALG-SA) to determine the target visiting sequence. However, since this is a multiple-target case, the target-cluster assignment is required. A clustering algorithm (CA) is applied to determine the target assignment.

Table 5.9 Experiment results of HCA + SOCP and ALG-SA + CA + SOCP for SVMT

	HCA + SOCP				ALG-SA + CA + SOCP			
***n* target (s)**	**Avg optimal makespan (time unit)**	**Avg comp. time (sec)**	**Stdev comp. time**	**Opt. gap (%)**	**Avg optimal makespan (time unit)**	**Avg comp. time (sec)**	**Stdev comp. time**	**Opt. gap (%)**
3	35.85	0.15	0.02	0.00%	35.85	0.08	0.00	0.00%
5	42.44	0.11	0.12	0.00%	42.44	0.09	0.00	0.00%
7	65.11	0.14	0.08	0.00%	65.11	0.11	0.00	0.00%
9	73.08	169.64	28.17	0.00%	73.15	0.18	0.07	0.10%
11	73.95	1,800	0	45.02%	80.2	0.33	0.25	8.45%
13	75.20	1,800	0	75.36%	81.3	0.62	0.02	8.11%
15	86.60	1,800	0	79.06%	85.40	0.74	0.12	1.39%
17	N/A*	1,800	0	N/A	131.18	0.85	0.08	N/A
19	N/A*	1,800	0	N/A	159.83	0.84	0.07	N/A
25	N/A*	1,800	0	N/A	183.06	1.00	0.00	N/A

Table 5.9 shows the experiment results for the model HCA + SOCP and the algorithm ALG-SA + CA + SOCP. From the figures, it can be inferred that the model cannot find an optimal solution for cases of more than eleven targets. Furthermore, when the target number is increased to 17, the model cannot find a feasible solution. Meanwhile, the algorithm can provide a good solution for small and large instances in a considerable amount of time. Figure 5.5 illustrates the computational time difference between the CA + SOCP and ALG-SA + SOCP.



	3	5	7	9	11	13	15	17	19	25
HCA+SOCP	0.15	0.11	0.14	169.64	1,800	1,800	1,800	1,800	1,800	1,800
ALG-SA+SOCP	0.08	0.09	0.11	0.18	0.33	0.62	0.74	0.85	0.84	1.00

Figure 5.5 Comparison of computational time for SVMT

Besides testing small cases, we also test large cases (*n*=30, 50, or 70) by ALG-SA.

Table 5.10 Experiment results on large instances for SVMT

***n* targets**	**Optimal Makespan (time unit)**	**Mean (s)**	**Standard Deviation (s)**
30	274.66	1.48	0.09
50	492.77	1.99	0.18
70	695.15	3.01	0.15

From the single-target and multiple-target experiment results, it is inferred that the application of the ALG-SA algorithm improves the performance of SOCP significantly. ALG-SA helps to find the target visiting sequence, which becomes a bottleneck in the SOCP formulation. Implementing the ALG-SA makes it possible to find a good solution for large cases within seconds.

5.3.2. Two-vehicles model (TV)

In the multiple-vehicle single-carrier model, the mathematical programming model for multiple-vehicle cases only gives the exact locations for takeoff and landing coordinates. Thus, to know the algorithm's performance, a total enumeration is conducted for small instances. Then, experiments on large instances are also conducted.

1. Single-target-per-flight scenario (TVST)

   For a single-target-per-flight scenario, there are three scenarios, ABAB, ABBA, and AABB. To know which scenario produces the shortest makespan, one way is to enumerate all possible scenarios. However, enumerating all of the possible scenarios is expensive in terms of computational time. Thus, ALG-SP is used to determine which scenario results in the best makespan. To show the performance of ALG-SP, several small cases are tested with different problem settings. We compare them to the result calculated by a total-enumeration-based model to see which scenario produces the smallest makespan. The experiments are conducted on eight random problems with different vehicle endurance lengths.

Table 5.11 Total Enumeration for TVST

Problem name	**Vehicle endurance**	**Total enumeration**			**ALG-SP best sequence**
		ABAB	**ABBA**	**AABB**	
P1	10	63.81	64.25	**63.43**	AABB
P2	10	12.88	12.88	**10.52**	AABB
P3	20	**33.89**	**33.89**	33.89	ABBA
P4	20	18.51	18.51	**18.32**	AABB
P5	30	**18.51**	**18.51**	21.36	ABAB
P6	30	**18.51**	**18.51**	23.02	ABAB
P7	40	**14.14**	**14.14**	16.26	ABBA
P8	40	**11.44**	**11.44**	13.37	ABAB

Based on Table 5.11, ALG-SP can provide the best scenario, compared with the total enumeration. ALG-SP can be used to determine which scenario that gives the best sequence. Then, ALG-SP's performance on large instances is assessed by conducting further experiments. Given a target visiting sequence, we decompose all targets into two targets (one target for one vehicle), respectively. Then, we solve that particular group and solve the following group using SOCP. Table 5.12 shows the experiment results of several test cases in terms of computational time.

Table 5.12 Experiment Results for TVST

***n* targets**	**ALG-SA (s)**	**ALG-SP (s)**	**SOCP**		**Total Comp Time**	
			Comp Time (s)	**Opt Makespan**	**Mean (s)**	**Std. Deviation**
5	0.07	0.001	0.10	83.56	0.17	0.002
9	0.11	0.001	0.17	128.18	0.28	0.002
11	0.13	0.001	0.21	155.00	0.35	0.019
13	0.15	0.001	0.26	142.94	0.40	0.003
17	0.18	0.002	0.33	188.86	0.51	0.015
21	0.21	0.002	0.43	238.55	0.64	0.011
25	0.33	0.004	0.50	252.72	0.84	0.150
30	0.57	0.006	0.60	298.00	1.18	0.012
50	0.94	0.016	1.04	458.70	1.99	0.017
70	1.23	0.031	1.45	699.10	2.71	0.031

2. Multiple-target-per-flight scenario (TVMT)

The mathematical programming model only determines the exact takeoff and landing coordinates like the single-target-per-flight cases. The input for this mathematical programming model needs to be determined using algorithms. A total enumeration is conducted for small cases to see the algorithm's performance,

Suppose there are four targets in the network, one starting point, and one ending point. After solving the ALG-SA, the following sequence is: (39, 16), (33, 1), (29, 49), (50, 44), (47, 41), (47, 22). For simplicity, rename the starting point (39, 16) to be *s*, ending point (47, 22) to be *e*, and the target points (33, 1), (29, 49), (50, 44), (47, 41) to be 1, 2, 3, 4 respectively. The goal is to determine the takeoff and landing point for composition. Table 5.13 shows the total enumeration to find the best composition.

Table 5.13 Composition of Total Enumeration for TVMT

Composition	Sequence	Makespan
1	s, 1, 2 – 4, e	**54.58**
2	s, 1, 2, 3 – 4, e	57.36
3	s, 1 – 2, 3, 4, e	61.12
4	s, 1, 2 – 3, 4, e	60.43
5	s, 1 – 3, 4, e	68.11
6	s, 1 – 2, 3 – 4, e	57.39
7	s, 1 – 4, e	69.15

Since all the possible compositions are already enumerated, we compare the result by total enumeration to the result by ALG-SP. The total enumeration result shows that the shortest makespan is 54.58, produced by composition 1 (s, 1, 2 – 4, e).

Algorithm ALG-SP will determine the target composition using the procedure that has been discussed in Chapter 4. The result from the ALG-SP shows that the best composition is s, 1, 2 – 4, e. This proves that ALG-SP gives a good estimate to determine which composition is the best.

This research conducts several experiments to show the performance of ALG-SP in terms of computational time. To obtain the exact takeoff and landing coordinates for target composition, we conduct a 4-stage heuristic: simulated annealing (ALG-SA), clustering analysis (CA), Shortest Path (ALG-SP), and SOCP. Table 5.14 shows the computational time for the experiments.

Table 5.14 Experiment Results for TVMT

***n* targets**	**HCA (s)**	**ALG-SA**		**ALG-SP (s)**	**Target comp. SOCP (s)**	**Makespan**	**Total Comp. Time**	
		SA (s)	**Individual SOCP (s)**				**Mean (s)**	**Std. Deviation**
5	0.001	0.06	0.06	4.36E-05	0.02	65.6	0.14	0.01
9	0.001	0.09	0.13	5.13E-04	0.02	124.01	0.24	0.01
15	0.001	0.29	0.25	0.001	0.02	163.21	0.56	0.01
25	0.001	0.46	0.46	0.003	0.02	333.72	0.95	0.01
30	0.001	0.84	0.99	0.016	0.02	410.74	1.86	0.70
50	0.001	0.85	0.51	0.014	0.03	657.25	1.39	0.01
70	0.001	1.15	1.39	0.03	0.03	961.89	2.57	1.05

5.4. Summary

There are two main scenarios to solve the carrier vehicle traveling salesman problem (CVTSP): the single-vehicle single-carrier scenario and the two-vehicles single-carrier scenario. Each scenario has two different target settings: the single-target-per-flight scenario and the multiple-target-per-flight scenario (target composition). Solving for an optimal solution by GUROBI is expensive in terms of computational time, especially for larger instances. Therefore, two main algorithms are proposed: a simulated-annealing-based algorithm (ALG-SA) and a shortest-path-based algorithm (ALG-SP). ALG-SA calculates a good target visiting sequence, and ALG-SP calculates a good target composition.

Several experiments are conducted to assess the performance of the algorithm. For the single-vehicle-single-carrier cases, this research compares the performance of the MIP model with ALG-SA. Based on a good target visiting sequence by ALG-SA, we then solve the SOCP to obtain the exact takeoff and landing coordinates. Based on the computational experiments for the two sub scenarios (single-target and multiple-target), the MIP model cannot find an optimal solution within 1,800 seconds when there are more than 11 targets. However, when ALG-SA is implemented, the total computational time reduces significantly. Implementing ALG-SA reduces the total computational time to less than 1 second for cases up to 25 targets. ALG-SA also proved to be capable of handling large instances in a considerable amount of time.

For the two-vehicle cases, our mathematical programming model requires the target visiting sequence as an input. Then it can calculate the exact takeoff and landing coordinates for the two-vehicle cases. The target visiting sequence can also be calculated by the algorithms developed in this research. A total enumeration is used to verify the correctness of our algorithm since there is no other mathematical model in the literature to compare.

Based on the total enumeration for the single-target and multiple-target cases, it was verified that our algorithm could provide the same result as the total enumeration. Our algorithm can deal with larger cases. From the experiments that have been conducted, ALG-SA and ALG-SP can solve large cases within seconds. For example, the largest case we have tested has 70 targets, which can be solved in less than 3 seconds.

Chapter 6
CONCLUSIONS AND FUTURE RESEARCH

6.1. Conclusions

This thesis focuses on the vehicle(s) conducting joint operations with its carrier, considering the vehicle's endurance. The coordination with a carrier accommodates the limitation. Our objective is to minimize the makespan of visiting all targets by vehicles launched from, landed to, and carried by a carrier. Unlike previous research that treats the makespan as an integrated duration, this research is the first to divide the entire duration into two parts: traveling and waiting periods. We hope to minimize the makespan, and we also wish to minimize the travel time in the hope of fuel reduction (which may reduce the CO_2 emission). The mathematical programming model can calculate an optimal solution using GUROBI, but would suffer from long computational time, especially for larger cases. Therefore, this research introduces several heuristical algorithms to calculate good solutions within a much shorter computational time. To this end, this research proposes two heuristical algorithms: one based on the simulated annealing mechanism (ALG-SA) and the other based on the shortest path algorithm (ALG-SP). These algorithms are used to calculate a good target visiting sequence and flight plan as an input for the SOCP model to calculate the vehicle takeoff and landing coordinates in a shorter computational time. To assess the performance of the algorithm, this research conducts some computational experiments. The conclusions derived from our study are as follows:

1. Carrier-vehicle Traveling Salesman Problem (CVTSP) is a problem that deals with the coordination between a carrier and a vehicle to determine the visiting sequence as well as the exact location of takeoff and landing coordinates. This research deals with two main scenarios of the CVTSP: the single-vehicle cases and the multiple-vehicle cases. Due to computational complexity, this research limits the number of vehicles to one or two.

2. This research further divides each scenario into two sub scenarios: the single-target and the multiple-target per flight scenarios. While most related literature deals with the single-target per flight cases, we also deal with the multiple-target per flight cases and propose the first integer programming model for solving it.
3. Solving a CVTSP using a mathematical programming model by GUROBI requires a long computational time, especially when dealing with large instances. Therefore, this research develops efficient algorithms to solve CVTSP.
4. A clustering algorithm (CA) is used to solve the problem of assigning each target point to a suitable cluster. CA is used when dealing with two vehicles with multiple-target per flight (target composition) since it is necessary to know the assignment of each target in advance.
5. A simulated annealing algorithm (ALG-SA) is designed to determine a good target visiting sequence. The result from ALG-SA will be used as an input for the SOCP. Based on the computational experiments, it is proved that ALG-SA can shorten the computational time for both small instances and large instances, while GUROBI only finds an optimal solution for small instances.
6. The shortest path algorithm (ALG-SP) is designed to determine the target composition for the two-vehicle and the multiple-target per flight cases. Since the mathematical programming model needs some input from the algorithms, this research cannot compare the mathematical programming model and ALG-SP because these two mechanisms are used in different stages. To assess the performance, this research uses a total enumeration to obtain an optimal solution to compare the result by our algorithm. It is proved that our algorithm can provide the same optimal result as the total enumeration does.
7. This research conducts extensive computational experiments on cases ranging from small and large instances. For each case, there are five subcases with randomized network

configurations. The results indicate that our proposed algorithms can calculate good solutions for small and large instances within a considerable amount of time.

6.2. Future Research

Solving a vehicle routing problem in a Euclidean space requires a specific method since only quadratic constraints can be used. Thus, to the best of our knowledge, only a few research deals with the routing problem in the Euclidean space. Therefore, there are some related issues worth future investigations.

1. Increasing the number of vehicles and carriers

   By increasing the number of vehicles and carriers, it is expected to reduce the total makespan. However, adding more vehicles and carriers means adding more variables and constraints to make the formulation more complex. Furthermore, adding more vehicles and carriers does not always necessarily mean good. If there are too many vehicles and carriers, the final result may become worse. Therefore, this particular direction can become one of the considerations.
2. Range of communication between vehicle and carrier

   This research always assumes that the vehicle is always able to return to the carrier, without considering the distance between the vehicle and carrier. However, in an actual situation, the vehicle and carrier may have a communication range.
3. Vehicle speed and battery consumption

   This research assumes that the vehicle and carrier always travel at constant speeds to simplify the problem. In practice, the vehicle may travel at different speeds. In addition, different speeds might affect the battery consumption rate. This direction might represent the situation in a real case since it is unlikely that a vehicle will travel at a constant speed throughout the mission.

4. Several types of vehicles

   This research assumes that the vehicle is homogenous, so the vehicle has the same characteristic to simplify the model. Several types of vehicles may be deployed for different missions in an actual situation, depending on how suitable a vehicle is.

5. Development of better mathematical programming models and algorithms

   The mathematical programming models in this research are pretty time-consuming, especially when solving for the best target visiting sequence. Efficient and effective mathematical programming models and algorithms require to be designed.

Reference

Agatz, N., Bouman, P., & Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science, 52*(4), 965-981. doi:10.1287/trsc.2017.0791

Chowdhury, S., Emelogu, A., Marufuzzaman, M., Nurre, S. G., & Bian, L. (2017). Drones for disaster response and relief operations: A continuous approximation model. *International Journal of Production Economics, 188*, 167-184. doi:10.1016/j.ijpe.2017.03.024

de Andrade, R. C. (2016). New formulations for the elementary shortest-path problem visiting a given set of nodes. *European Journal of Operational Research*, 254(3), 755-768. https://doi.org/10.1016/j.ejor.2016.05.008

Dorling, K., Heinrichs, J., Messier, G. G., & Magierowski, S. (2017). Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47*(1), 70-85. doi:10.1109/tsmc.2016.2582745

Duhamel, C., Lacomme, P., & Prodhon, C. (2011). Efficient frameworks for greedy split and new depth first search split procedurs for routing problems. *Computers & Operations Research, 38*(4), 723-739. doi:10.1016/j.cor.2010.09.010

Erdoğan, G., & Yıldırım, E. A. (2020). Exact and heuristic algorithms for the carrier–vehicle traveling salesman problem. *Transportation Science*. doi:10.1287/trsc.2020.0999

Fahradyan, T., Bono Rossello, N., & Garone, E. (2020). Multiple carrier-vehicle travelling salesman problem. In *Modelling and Simulation for Autonomous Systems* (pp. 180-189).

Gambella, C., Lodi, A., & Vigo, D. (2018). Exact solutions for the carrier–vehicle traveling salesman problem. *Transportation Science, 52*(2), 320-330. doi:10.1287/trsc.2017.0771

Garone, E., Determe, J.-F., & Naldi, R. (2012). *A travelling salesman problem for a class of heterogeneous multi-vehicle systems*. Paper presented at the IEEE 51st IEEE Conference on Decision and Control (CDC). https://ieeexplore.ieee.org/document/6426488/

Garone, E., Naldi, R., Casavola, A., & Frazzoli, E. (2008). *Cooperative path planning for a class of carrier-vehicle systems*. Paper presented at the 2008 47th IEEE Conference on Decision and Control.

Garone, E., Naldi, R., Casavola, A., & Frazzoli, E. (2010). *Cooperative mission planning for a class of carrier-vehicle systems*. Paper presented at the 49th IEEE Conference on Decision and Control (CDC).

Ha, Q. M., Deville, Y., Pham, Q. D., & Hà, M. H. (2018). On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies, 86,* 597-621. doi:10.1016/j.trc.2017.11.015

Klaučo, M., Blažek, S., Kvasnica, M., & Fikrar, M. (2014). M*ixed-integer SOCP formulation of the path planning problem for heterogeneous Multi-Vehicle Systems*. Paper presented at the 2014 European Control Conference (ECC).

Li, Z., & Garcia-Luna-Aceves, J. J. (2006). Finding multi-constrained feasible paths by using depth-first search. *Wireless Networks, 13*(3), 323-334. doi:10.1007/s11276-006-7528-8

Luo, Z., Liu, Z., & Shi, J. (2017). A two-echelon cooperated routing problem for a ground vehicle and its carried unmanned aerial vehicle. *Sensors (Basel), 17*(5). doi:10.3390/s17051144

Manyam, S. G., Rasmussen, S., Casbeer, D. W., Kalyanam, K., & Manickam, S. (2017). *Multi-uav routing for persistent intelligence surveillance & reconnaisance mission*. Paper presented at the 2017 International Conference on Unmanned Aircraft Systems (ICUAS)

Otto, A., Agatz, N., Campbell, J., Golden, B., & Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. *Networks, 72*(4), 411-458. doi:10.1002/net.21818

Poikonen, S., & Golden, B. (2020). The mothership and drone routing problem. *INFORMS Journal on Computing, 32*(2), 249-262. doi:10.1287/ijoc.2018.0879

Poikonen, S., Golden, B., & Wasil, E. A. (2019). A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing, 31*(2), 335-346. doi:10.1287/ijoc.2018.0826

Tokekar, P., Hook, J. V., Mulla, D., & Isler, V. (2016). Sensor planning for a symbiotic uav and ugv system for precision agriculture. *IEEE Transactions on Robotics, 32*(6), 1498-1511. doi:10.1109/tro.2016.2603528

Wang, X., Poikonen, S., & Golden, B. (2016). The vehicle routing problem with drones: several worst-case results. *Optimization Letters, 11*(4), 679-697. doi:10.1007/s11590-016-1035-3

Yakıcı, E. (2016). Solving location and routing problem for UAVs. *Computers & Industrial Engineering, 102*, 294-301. doi:10.1016/j.cie.2016.10.029

Yu, K., Budhiraja, A. K., & Tokekar, P. (2018). *Algorithms for routing of unmanned aerial vehicles with mobile recharging stations*. Paper presented at the 2018 IEEE International Conference on Robotics and Automation (ICRA).