國 立 成 功 大 學

工 業 與 資 訊 管 理 研 究 所

碩 士 論 文

以艦載無人船隊執行區域覆蓋任務之最佳子母船聯合路線規劃問題研究

Optimal Collaborative Path Planning for Area Coverage Tasks by a Parent Boat Carrying Unmanned Surface Vehicles

研 究 生 ：皮卡漢 Ari Carisza Graha Prasetia

指導教授：王逸琳 教授

中 華 民 國 一 百 零 八 年 五 月

doi:10.6844/NCKU202000986

國立成功大學

碩士論文

以艦載無人船隊執行區域覆蓋任務之
最佳子母船聯合路線規劃問題研究
Optimal Collaborative Path Planning for Area Coverage Tasks by a Parent Boat Carrying Unmanned Surface Vehicles

研究生：皮卡漢

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
丁慶榮
[illegible]

指導教授：王逸琳

系(所)主管：[illegible]

中 華 民 國 109 年 5 月 29 日

doi:10.6844/NCKU202000986

摘要

隨著傳感和無線通信技術的進步，以遠端遙控的無人船隊（USV）可取代人力駕船方式，在特殊（譬如易擱淺、觸礁）的水域巡航以執行搜尋或科研任務。當 USV 的航線可進行標靶區域的任務時，該標靶區域即可視為被「覆蓋」，獲取其任務效果。假設水域中有多個標靶區域散佈四處，若能有效地規劃多台 USV 路線以同時巡航，將能在更短時間內完成更多的區域覆蓋任務以獲取最大的任務效果。然而， USV 體積小、續航力短，必須與其被搭載的母船(PB)會合才能進行充換電以重新出發。因此，母船的路線也應一併規劃才能發揮最大綜效。

本論文即探討在時限內能獲取最大任務效果的子母船聯合覆蓋路徑規劃問題，首先我們先針對母船之航線已知（但時程未知）的簡化版問題設定，建構一個以時空網路為基礎的整數規劃模型，由於該模型含有大量變數與限制式，僅能處理極小規模的情境，因此我們再以分進合擊的方式將分佈四處之標靶區域劃分成合適個數的分群任務區域，而各區域再以整數規劃模型(ICH)或局部搜尋演算法(ILS)分別處理，最後再串接母船航線。此簡化問題的測試結果顯示，ILS 演算法效率極佳，但 ICH 模型僅適用於標靶必須較均勻分佈的情境。之後，我們再進一步將 PB 路線設成未定，亦建構整數規劃模型、修改 ICH 與 ILS 演算法，並設計「分段式啟發演算法」(SSH)，將規劃期間切成數個時段，再依序處理規模較小的整數規劃問題以串接成最後決策。測試結果顯示 SSH 與 ILS 皆有不錯之求解表現。

關鍵字：覆蓋路徑規劃，整數規劃，局部搜尋演算法，時空網絡，無人船

doi:10.6844/NCKU202000986

Abstract

Due to improvements in sensing and wireless communication technology, a fleet of unmanned surface vehicles (USVs) can now be used to search water areas for the purpose of carrying out multiple targeted operations. Specifically, multiple USVs can conduct search tasks simultaneously. However, due to its limited battery or fuel capacity, a USV requires recharging (or refueling) or battery swapping to continue longer range missions. Thus, we propose a mechanism in which USVs can depart from and return to a parent boat (PB) for recharging and data collection.

Assuming each target has a value of importance, this thesis investigates how to calculate optimal routes for both PB and USVs so that USVs can cover all the target areas, with the total target values as the objective for operating in a limited amount of time. We propose an integer programming formulation on a time-space network for cases of a given PB path (C1) and the case where the PB path is not yet known (C2), based on techniques similar to the coverage path planning (CPP), the orienteering problem, and the two-echelon routing problem in the literature. Although mathematical formulations can be used to solve the problem, this is quite time-consuming.

We thus designed three methods to solve the problems related to calculating good routings in a shorter time. The first method is the Iterative Clustering Heuristic (ICH) based on the clustering technique, which was intended to limit the workspace for each USV. The second method is the Iterative Local Search (ILS) algorithm based on multiple local search procedures, which was intended to further shorten the computational results. The third method is the Sequential Segmentation Heuristic (SSH), which is based on the segmentation process and was intended to limit the time-bound in each segment. The results of our preliminary computational experiments for both cases and a real-world scenario problem setting indicate that the proposed solution methods can calculate good search plans faster than the proposed integer programming model.

Keywords: Coverage Path Planning, Integer Programming, Local Search, Time-Space Network, Unmanned Surface Vehicles

doi:10.6844/NCKU202000986

Acknowledgements

First of all, I would like to express my gratitude for my advisor, Prof. I-Lin Wang (王逸琳) for his assistance during my study, for his guidance and dedication so that this thesis is completed. I learned a lot from him that is very useful for my future. I also would like to thank all committee members, Prof Hsieh (謝中奇), Prof. Lee (李賢得), Prof. Hsieh (解巽評) and Prof. Ting (丁慶榮). 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 Jib, Chia-Hao , Garry, Xue-Mei and Meidy. To my lab mates, Jessica (岳晏慈), Steven (陳彥瑋), and Edward (王宗瀚).To my juniors 張書桓, 胡承中, and Rani. Thank you all for helping me to be able to face many struggles and hardship during my study, willing to have a discussion and sharing, giving me suggestions or making this lab lively so that I can relieve the stress. I must say that having you guys as lab mates were a wonderful experience, I am able to have an enjoyable life here in Taiwan and it has become one of the precious moments in my life.

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 the difficulties that I have 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.

Ari Carisza Graha Prasetia

Department of Industrial and Information Management

National Cheng Kung University, Tainan City, Taiwan R.O.C.

May 2020

doi:10.6844/NCKU202000986

Table of Contents

doi:10.6844/NCKU202000986

doi:10.6844/NCKU202000986

doi:10.6844/NCKU202000986

List of Figure

doi:10.6844/NCKU202000986

doi:10.6844/NCKU202000986

doi:10.6844/NCKU202000986

List of Table

doi:10.6844/NCKU202000986

doi:10.6844/NCKU202000986

Chapter 1

INTRODUCTION

To conduct search operations in a wide water area such as shallow sea or a lake, a fleet of unmanned surface vehicles (USVs) can be deployed to patrol different water areas simultaneously. These USVs are usually smaller, more mobile, faster, and can patrol over shallow areas that cannot be passed through by large vessels. However, their movement range is limited due to a smaller tank of fuel or the use of a battery. A USV can be recharged (or refueled) if it can return to a large vessel that serves as a parent boat. In other words, a parent boat serves as a mobile recharging (or refueling) station, where each USV can depart from it to patrol and then return to it for recharging (or refueling). It is assumed that the entire water area can be approximated by a grid network, where some nodes or arcs must be searched (i.e., passed) by either USVs or the parent boat, depending on the depth of the water in the area.

There are two main cases analyzed in this thesis, a case where a PB route is known, and a case where the PB route is not yet known. The main focuses of this thesis are three-fold: to optimize routings of both PB and USVs that finish search tasks in a minimum amount of total time, to investigate the impact of a given PB route to the performance of the proposed Integer Programming (IP) model, and to propose some heuristics that can help to calculate these solutions faster. This chapter is divided into 5 sections. We first give the background for our research in Section 1.1. In Section 1.2, we explain the research objective. The research scope is defined in Section 1.3, the research contributions are given in section 1.4., and the outline for the thesis is provided in Section 1.5.

1.1. Research Background

Our research is aimed at maximizing the total value of targets secured, or to provide service for a set of selected targets within a limited operation time, using a fleet of unmanned surface vehicles (USVs) like a swarm operation. Due to its size and capabilities, USVs can be used to navigate autonomously in a dynamic and uncertain environment.

doi:10.6844/NCKU202000986

Figure 1.1. USVs deployment from a parent boat (Matos et al., 2017)

Since unmanned surface vehicles (USVs) are now heavily relied upon for surveillance, intelligence, search, and rescue (Campbell, 2012), the difficulties related to finding effective routing for their operations can be significantly affected by the chaotic nature of the wake region downstream of the ship stern as well as the sea conditions during the time of the incident. Several aspects have to be considered, such as the number of units needed for the operation and the coordinates of the targets. This operation is conducted in an environment where different types of services can be performed in a predetermined route for a big ship to operate on as well as the necessary, for example, drones and the ground vehicle or drones and PB needed to cover a set of locations. This operation can be described as a joint operation problem of a PB with multiple USVs or as a two-echelon problem.

The created model is intended to represent the corresponding problem and illustrate how differences in the number of USVs between each iteration and network structure can affect the time required for each USV to complete a task. In addition, we hope to find sufficient routing to perform the operations while considering USV battery capacity in a case of a PB path that is predetermined where targets can be covered in a limited amount of operating time. Understanding the time needed and the resources are very important for the decision-maker. Since the expected computational results will consume a lot of time, the numerical evaluations of the optimization model will be used to illustrate different cases in simulated networks and networks based on real data. These cases are then used to analyze how parameter changes can affect the performance of the proposed IP mathematical formulations and the heuristic.

To perform the search task for each USV with consideration on the energy consumption, efficient path planning is needed. Traditional path planning algorithms, such as those based

doi:10.6844/NCKU202000986

on the Traveling Salesman Problem (TSP), are not ideal for this kind of problem since such formulations only consider the distances traveled but ignore the energy or range constraint due to the limited fuel or battery capacity. Therefore, this research mainly analyzes other energy-based optimization methods utilizing Energy Efficient Coverage Path Planning (EECPP), another formulation that considers the energy consumption characteristics of drones in path planning optimization. We not only consider the energy consumed when traveling between consecutive waypoints (similar to the TSP), but we also consider the energy consumed by the USV when it has to return to the recharging facility (e.g., a PB). The IP model is structured based on an Integer Linear Programming model. A linear energy consumption model allows for linear energy consumption constraints, which are required to derive a compatible result with some optimization solvers such as GUROBI.

1.2. Research Objective

Our research objective is to develop mathematical models and heuristic algorithms for the purpose of planning the paths of multiple USVs and a PB to conduct a collaborative search task in order to obtain sufficient information related to the effects of having a known PB path on the performance of the proposed model. The task requires visiting some target nodes, staying there for some time to perform some on-site tasks (e.g., sample collection, sensing, or some experiments), and then moving to rendezvous with the PB for battery recharging or swapping. Assuming it is possible to obtain some value (importance level) for each completed task, we want to determine the total collected values that can be maximized within a given time limit.

1.3. Research Scope

The joint operation is defined as follows: USVs are given a set of areas in which to operate under a working time constraint to complete the operations on the targets within this area while the parent boat serves a supportive role for the USVs. In the case of C1, the PB route is already fixed, while in the case of C2, this path is not fixed yet, and therefore, our methods have to find the path for the PB. For both of these cases, the USVs can move anywhere in their area while the parent boat can only move in the designated path/area. During the operation, the parent boat will be able to launch and retrieve a USV from different

doi:10.6844/NCKU202000986

points. It also has the option to wait for a USV to finish an operation before moving to the next area.

In this model, the Coverage Path Planning (CPP) approach can be used to model the USV routes. To find the area for the paths, the area is decomposed using a grid-based method. For the optimization of this problem, a new method is proposed to combine the CPP with another method called a time-space network two-echelon routing problem. This method is used to consider a joint operation between a USV and a parent boat so that both the USV and the parent boat can be solved simultaneously rather than individually.

1.4. Research Contribution

Our research contribution can be seen from two perspectives. To conduct an algorithm that can help to solve a joint operation problem faster. Aside from an algorithm, to create possible mathematical models and heuristic models that can help to solve the problem faster is also a direction for this research. While several approaches are available for solving this problem, none appear to consider energy consumption as a function in their programming. Routes found using the above approaches may be infeasible for operation scenarios with batteries that are larger than necessary or too heavy to carry. Therefore, this research may complement research such as energy analysis-based research intended to find multiple USV capabilities related to conducting path planning combining the concept of a two-echelon routing orienteering problem with the concept of coverage path planning. Another point is that by analyzing the PB path in specific cases, this research can be used to evaluate a PB path decision that has been made beforehand and provide better decision making for the user in securing as many targets as possible in a limited time.

For the second perspective considering the number of PBs and USVs, most of the related research in a joint operation considers only cases of one-one operation. On the other hand, this research, which focuses on multiple USVs, may provide a better insight into managing multiple USVs and the computational time needed to do so, which is a weakness in research on joint operation path planning.

doi:10.6844/NCKU202000986

1.5. Thesis Outline

The rest of this thesis is organized as follows. In Chapter 2, we provide some previous studies and theories that related and support our research, together with our research contribution. The problem definition and formulation, including initial mathematical models, will be explained in Chapter 3. Chapter 4 explains the heuristic and algorithm that we develop for this research. Chapter 5 explains computational experiments for case C1 and case C2 and the summary of the computational experiments. Chapter 6 is for the conclusions and future research suggestions for our research.

doi:10.6844/NCKU202000986

Chapter 2

LITERATURE REVIEW

In this chapter, the literature review is divided into four sections. The main theme of our research is the combination of a joint operation and an orienteering problem for area coverage. The literature related with the joint operation of unmanned vehicles is described in Section 2.1, followed by USV characteristics in Section 2.2, the orienteering problem in Section 2.3, and coverage path planning in Section 2.4, together with related studies related to each theory.

2.1. Joint Operation of Unmanned Vehicles

A join operation of unmanned vehicles is defined as having an operation involving multiple unmanned vehicles combined with other vehicles. In this research, the vehicles are defined as unmanned surface vehicles (USVs) and the main vehicle or parent boat (PB). Since the topic of unmanned surface vehicle optimization is relatively new, similar studies that have been done previously include a joint operation examined by Garone et al. (2010) on a class of carrier vehicles in which a slow carrier with an infinite operating range cooperates with a faster carrier vehicle with a limited range in which to operate. The main objective function in their research was to make the faster vehicle visit a given collection of points in a minimum amount of time. The concept is similar to the one applied in the present research; however, it did not consider battery consumption or multiple vehicles.

Another study similar to the present one was conducted by Mirhedayanti et al. (2019). They address a new variant of the 2E-LRP (two echelon-location routing problem) that considers synchronization of transshipment and a sequence of delivery and pickup activities by using a primary vehicle and a secondary vehicle. Based on their research, larger instances of data may take a long time to solve since the synchronization problem is a difficult optimization problem. This is because it combines two NP-hard problems: facility location and vehicle routing, and it is especially difficult for problem involving larger instances. The results obtained with commercial solvers show that only very small instances are solved optimally, and larger instances cannot be solved with a feasible solution.

doi:10.6844/NCKU202000986

In a more complex environment, Mathew, Smith, and Waslander (2015) studied the path planning problem for a team of vehicles in an urban environment. Mathew computed a route for a truck and a quadrotor in which they meet and share edges during docked travel. This scenario is similar to the operation in the current study, which involves simulating the recharging of USVs. They also added further research suggestions on using several multi-robot systems in scenarios involving search and rescue, surveillance, and exploration in a joint operation optimization situation.

Another example of joint operation research can be observed from the truck-drone synchronization conducted in Ferrandez et al. (2016). They added that by observing the total time, cost, and energy involved in a hub configuration (star-distance), it is easier to contrast this configuration with truck-only delivery using a TSP route. However, the refueling constraint was not considered in this research, which proves to be a good example since refueling can make a joint operation more complex and has the potential to be applied for any other type of vehicle, especially in an uncertain environment. The distance used in this research and the grid computation is based on the Euclidean plane, which was validated by Carlsson and Song (2017) for a collection of computational experiments.

The synchronization optimization problem in the literature on 2E-LRP is a relatively new main area for research, as stated in a survey paper on two-echelon routing problems by Cuda, Guastaroba, and Speranza (2015) that highlights these aspects as worthy of investigation.

2.2. USV Characteristic

Unmanned surface vehicles as an alternative for search and rescue attempts have been proven to be effective in drowning scenarios (Matos et al., 2017). The USV is expected to conduct an intelligent, search, and secure (ISS) mission relying on power supplied by a unique hybrid system. The hybrid power system employs a diesel generator, a lithium-ion battery pack, a fuel cell system, wave power, and a solar array (Khare et al., 2011). Because of this limited power supply, USVs must be charged periodically before carrying out a task. Due to the nature of this research, a USV, rather than the parent boat, can be used to save the target faster. This scenario resembles the routes corresponding to a case where the UAV is about four times as fast as a truck. It is also worth mentioning that the optimal sequence

doi:10.6844/NCKU202000986

of the client destinations is not necessarily the same as that induced by the TSP tour (Carlsson and Song, 2017).

Specht et al. (2017) concluded that USVs provide a smart platform that can help people to finish projects safely and quickly, and can be widely used on rivers, busy channels, shallow water, and lakes. They are therefore a promising strategy for use in rescue operations. Specht also suggests that due to USV capabilities and intelligence, USVs can help avoid casualties during military operations and can, therefore, be applied to antisubmarine and anti-mine warfare, maritime security protection, and other military fields. These developments and their extensive validation in several trials and demonstrations are, therefore, a relevant contribution to the real-world deployment of automatic platforms in search and secure operations and can be implemented in traditional search and rescue teams.

Summary of studies related to combined vehicle's operation are shown in Table 2.1

Table 2.1. Combined Operation Related Researches

Literature	**Method**	**Vehicles Operation**	**Refuel**	**Application**
Garone et al. (2010)	M, H, Exp	Optimal flight settings		Transport
Mathew, Smith, and Waslander (2015)	M, H, Exp	Routing set of location	v	Transport
Ferrandez et al. (2016)	M, H, Exp	Routing set of location		Transport
Carlsson and Song (2017)	M, H, Exp	Routing set of location		General
Mirhedayanti et al (2019)	M, H, Exp	Routing set of location		Transport
This Research	M, Branch and Bound, Exp	Routing set of location	v	General

doi:10.6844/NCKU202000986

The notation that is used in Table 2.1 and later in Table 2.3 – 2.4 is explained in Table 2.2:

Table 2.2. Glossary of abbreviations

M	Mathematical model ready for input into an off-the-shelf solver (eg. Gurobi)
H	One or several customized heuristic algorithms. Literature may provide memory and computational complexity analysis of the proposed algorithms.
Exp	Computational experiments on artificial or real-world data

2.3. Orienteering Problem

When there is a limited amount of time with a set of targets, it is impossible to visit every destination available. Therefore, each USV has to select what is believed to be the most valuable or profitable (with higher profit) target. Making a feasible plan to visit the most relevant attractions in the available period is often a difficult task. This problem is considered to be an orienteering problem. An orienteering problem (OP) is a routing problem, where the aim is to generate a path through a set of nodes, which will maximize the total score and not exceed the budget, which in this case, is limited time.

For example, for the initial value, we have the importance value of each target. Based on these values, in the present case, we can consider that each USV is required to service the maximum set of targets based on their importance value during a limited period of time. In a real-world scenario, this can be picking up important items or performing rescue operations for a set of victims who are still alive based on pre-recorded data indicating them as a priority.

doi:10.6844/NCKU202000986

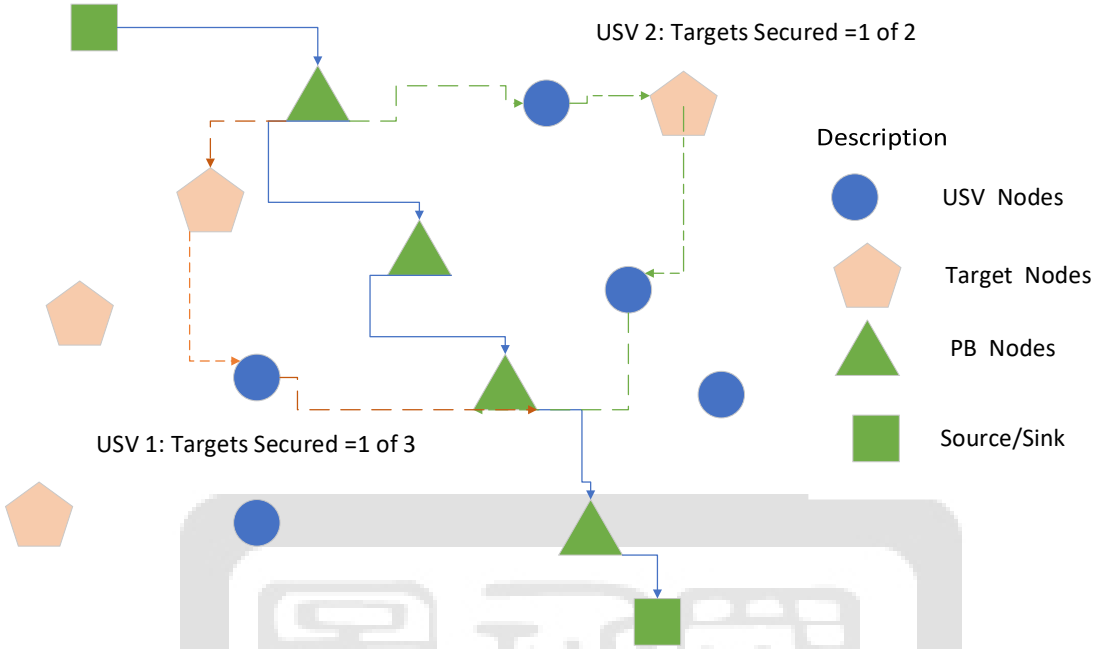


Figure 2.1. Orienteering Problem Environment

The orienteering part of this research will be that when considering limited resources (time), USVs will have to find the best routing to secure as many targets as possible and return to the parent boat within a given time limit. From Figure 2.1, out of all three feasible targets, USV 1 secures 1, and USV 2 secures 1 out of two possible targets. The figure illustrates the routing on which each USV can return to finish the operation within the time limit. This situation is used to describe the situation presented in this research.

In previous research, such as Dallard et al. (2007) and Vansteenwegen et al. (2009), the researchers developed algorithms based on orienteering problems to consider which target should be prioritized based on a specific value. Dallard et al. (2007) developed a further algorithm using particle swarm optimization based on the target's distance. Vansteenwegen et al. (2009) developed the iterated local search algorithm based on the insertion method and moved the targets based on their value and distance, which led to the recent development of a heuristic for an orienteering problem. This research can be used to further enhance our knowledge in developing an algorithm based on the orienteering problem presented in this work.

Labadie et al. (2012) also stated that the team orienteering problem (TOP) is a known

doi:10.6844/NCKU202000986

NP-hard problem that typically arises in vehicle routing and production scheduling contexts. This review is thus our consideration in trying to incorporate a vehicle routing problem into the orienteering problem for a joint operation since this topic can be considered as new in this field. Hajj, Dang, and Moukrim (2016) presented ideas suggesting that further development of a branch-and-cut-and-price type of algorithm is a promising direction towards improving the TOP solution method. They also added that to solve other variants of the TOP and VRP, such as a team orienteering problem with time windows and synchronization would be appropriate future directions in orienteering problem solutions.

To develop an orienteering problem, Mukhina, Visheratin, and Nasonov (2019) state that the majority of problems related to orienteering can be defined using three core components: a set of targets, constraints, and a scoring function. The fourth core component, the algorithm, utilizes other components to generate the best solution for a specific problem. Based on this knowledge, we organize this research based on this structure to develop the optimal solution to the given problem while considering the fuel consumption for each USV and synchronization to the parent boat. In our research, each target is clustered based on the position of the individual target. This was previously done by Yahiaoui, Moukhrim, and Serairi (2019) in order to reduce the computational time necessary to solve a set of targets. Since none of the previously mentioned studies considered the refueling capabilities of their vehicles, this is an added benefit to our research.

The summary of the studies related to the orienteering problem is shown in Table 2.3.

Table 2.3. OP related researches

Literature	Method	Routing	Refuel	Application
Dallard et al (2007)	M, H Exp			General
Vansteenwegen et al. (2009)	M, H Exp	V		General
Labadie et al. (2012)	M, H Exp	V		General
Hajj, Dang, and Moukrim (2016)	M, Branch and Bound, Exp	V		General
Mukhina, Visheratin, and Nasonov (2019)	M, H Exp	V		General

doi:10.6844/NCKU202000986

The summary of the studies related to the orienteering problem is shown in Table 2.3. (Cont.)

Literature	Method	Routing	Refuel	Application
Yahiaoui, Moukhrim and Serairi (2019)	M, H Exp	V		General
Our research	M, Branch and Bound, Exp	V	V	General

2.4. Coverage Path Planning

Coverage Path Planning (CPP) is the process of defining a path that passes over given points of an area of interest while avoiding obstacles (Galceran & Carreras, 2013). Based on this definition, our research can be categorized as coverage path planning for a given area of interest. Zhang, Liu, and Li (2019) presented a strategy for an intelligent search performed by a USV. Before the start of the operation, global path planning was carried out based on prior information, such as the initial position of USV, the predicted position of the target, and the range of the search area. An example of this prior information is a grid method used to plot the selected targets in an area. There are two major steps that CPP takes into consideration. The first step is the process of dividing the given area or the target space into sub-regions (called cells), also known as area decomposition and optimization.

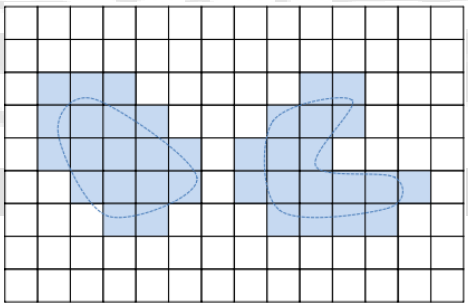


Figure 2.2. Grid-based method (Galceran and Carreras, 2013)

Grid-based methods were introduced by Moravec and Elfes (1985) as a representation of an environment that is decomposed into a set of uniform grid cells. This method, when combined with the time-space network nature of the present research, will increase the reliability of our results.

doi:10.6844/NCKU202000986

Given an already decomposed area, each USV is going to be deployed to search for the optimal routing path for an optimal solution. Some examples of these optimization techniques were performed by Avellar et al. (2015), who examined minimum time coverage of ground areas using multiple drones, and Chavez et al. (2019), who investigated vehicle planning with time windows. Both of these studies used an optimal coverage method as the area decomposition technique and the vehicle routing problem (VRP) to optimize the problem based on the area decomposition. Based on this research, VRP programming may be used as a feasible option for a multi-vehicle related problem, especially for coverage path planning for multiple vehicles, as is the case in the present research.

Zhang et al. (2015) state that the use of a time-space network is very effective for multi-vehicle problems. The time-space network is also a common method applied to slot allocation problems considering spatial and temporal requirements. The time-space network shows the position of an individual in time and space. Therefore, our method adapts it to the grid-based method to generate the optimal result. After decomposition, then the optimization step can be started.

There are many examples of multiple vehicle implementations based on the grid method, such as Carabaza et al. (2018) and Fu et al. (2018). Fu proposed coverage path planning based on the limited battery energy on multiple UAVs, where the objective function was determining the cost to decide whether the path is feasible and the cost of the UAV flight energy consumption. Carabaza presented a new approach based on Ant Colony Optimization to determine path planning for a fleet of unmanned vehicles in a limited time frame. Both of these papers can be used as the base algorithm in our research to develop unmanned vehicle tasks with limited battery energy for limited time operations.

doi:10.6844/NCKU202000986

The summary of the studies related to CPP related researches is shown in Table 2.4.

Table 2.4. CPP related researches

Literature	**Method**	**Area Decomposition**	**Unmanned Vehicles**	**Application**
Galceran and Carreras, (2013)	M, Exp	Grid-based	Multiple	General
Avellar et al. (2015)	H, Exp	Optimal Coverage	Multiple	Agriculture
Fu et al. (2018)	M, Exp	Grid-based	Multiple	General
Carabaza et al. (2018)	M,H, Exp	Grid-based	Multiple	General
Chavez et al. (2019)	M, Exp	Grid-based	Single	General
Zhang et al (2019)	M, Exp	Optimal Coverage	Singe	General
Our research	M, Branch and Bound, Exp	Grid-based	Single Multiple	General

doi:10.6844/NCKU202000986

Chapter 3

MODEL DEVELOPMENT

This chapter describes the modeling approach, the mathematical formulations and notations that will be used in this research as well as some examples about the IP model. The chapter of model development is divided into three sections. Section 3.1 descirbes our model framework. The problem definition is introduced in Section 3.2, followed by the mathematics model in Section 3.3.

3.1. Model Approach

As mentioned in Chapter 1, our primary objective is to plan a route for the join operation to service the targets during a given set of time. To do this, we need to plan an area of the grid network for USVs to operate. As shown in Figure 3.1, the grid will be used to define the workspace of the USVs to cover the points, and these points are considered as nodes.

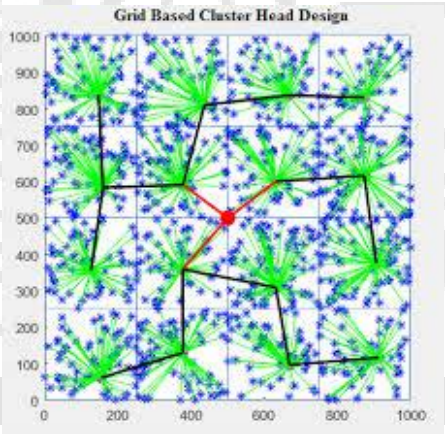


Figure 3.1.Grid-Based Decomposition (Kareem et al., 2018)

Based on these generated grids, a feasible path to achieving the optimal result is going to be generated for the parent boat, and this path will be used as the workspace for the operational movement of the parent boat. To define a route for the USV, we need to define the free workspace area into several waypoints. Facing this situation, then we can use the proposed mathematical programming-based approach. After dividing the size of the grid and the number of nodes in the system, we will create the time-space network based on this area

doi:10.6844/NCKU202000986

of space that we are going to analyze. The time-space network is a method that we can use to analyze a model of the area in multiple time durations, as shown in Figure 3.2. In Figure 3.2, terminal 1, 2, and 3 can be considered as the nodes or the area number in our grids, while the t =0, t=1, etc., are the time considerations. The use of the time-space network is beneficial to understand the details of the movement of the USVs in our model however as can be seen from the figure, in each time, there will be multiple nodes considered that will require a lot of computational time

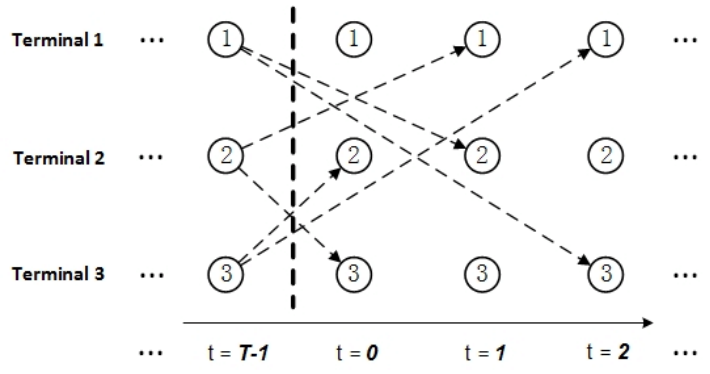


Figure 3. 2.Time-Space Network (Wang and Wallace, 2016)

3.2. Problem Definition

Our primary objective is to plan a route for a joint operation to service the targets during a given period of time. To do this, we need to plan a grid network area for the USVs to operate in. Based on these generated grids, a feasible path to achieving the optimal result can be generated for the PB, and this path will be used as the workspace for the operational movement of the PB. To define a route for the USV, we need to divide the free workspace area into several waypoints. The mechanism for the routing of each vehicle is constructed using a time-space network.

In this problem, there are two types of workspaces. The first is the USV workspace, which is exclusively for the USV, and the second is the PB workspace space, which is accessible to both the PB and the USV. Under actual conditions, the USV may roam freely in its workspace. To make a route for the USV, it is necessary to decompose the USV area into sub-regions. We use a grid-based method involving a decomposition technique that divides the airspace into a virtual uniform grid cell. Suppose that we have already done the

doi:10.6844/NCKU202000986

preprocessing step and obtained the grid map for the PB area and USV area. We can transform this grid map into a network graph by changing the virtual cells into nodes and the connections between each cell into an arc. As for the PB workspace, the nodes represent the potential places for the PB to launch the USV, and the arcs represent the route. We then combine these two workspaces. Figure 3.3 shows the network graph for the combined operation of the PB and USVs.

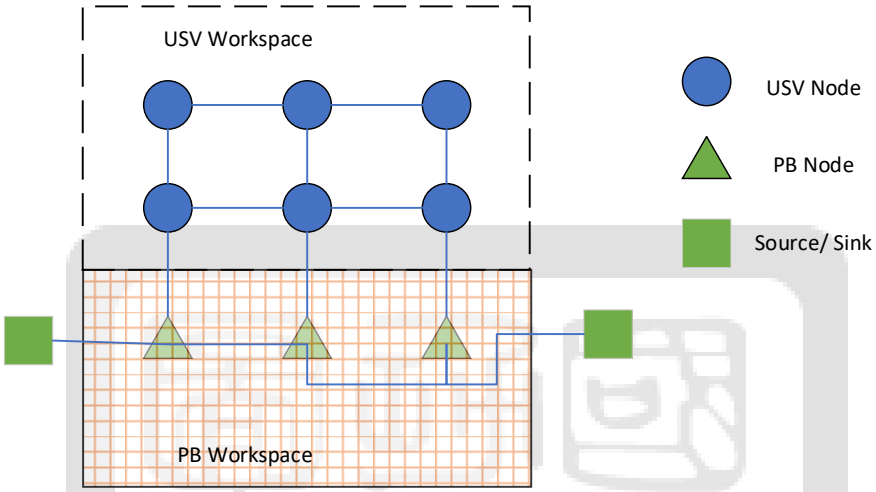


Figure 3.3.The schematic network for combined PB and USVs operation

Notice that in the network graph shown in Figure 3.3, there are arcs connecting the two workspaces. From a USV routing perspective, these arcs turn two distinct workspaces into one workspace. However, from a PB routing perspective, there are still two different workspaces, and the PB is only able to access one of them. An important point here is that the PB is required to follow the predetermined path when the exact movement is not yet known. There are multiple reasons for this mechanism. First, in the real world, the path for the PB is not as flexible as that for a USV due to its size, the locations of the harbors, and the need to avoid collisions with the other ship. This network then has similarity in terms of properties with the two-echelon routing problem network. We can use this concept to design our mathematical models. In order to record both the time and battery power consumption of the USVs, together with the PB movement at each time period, we use a time-space network. Figure 3.4 shows the results of the transformation from the original network into the time-space network.

doi:10.6844/NCKU202000986

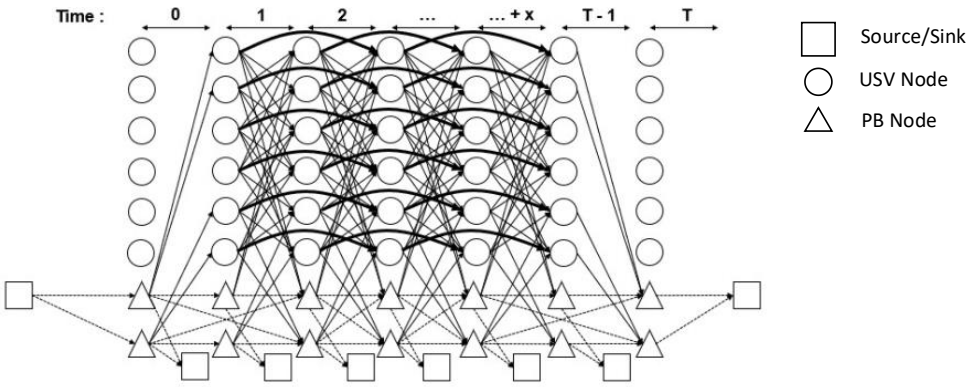
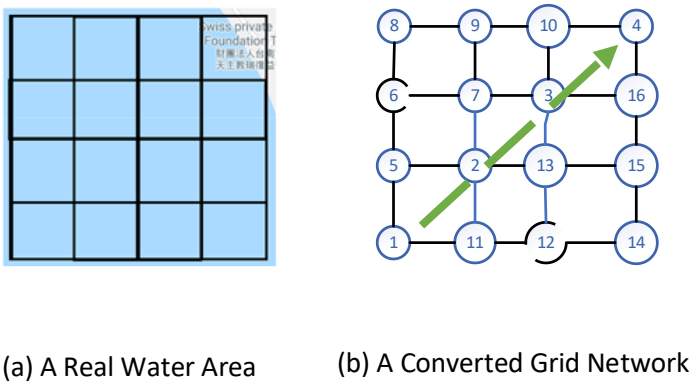


Figure 3.4.The schematic diagram for the proposed model

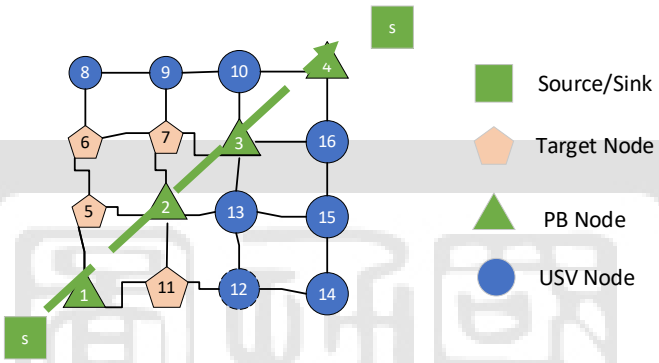
The use of time-space path network in our model is important to limit the possibilities of subtour eliminations and being able to provide a detailed solution, as suggested by Zhang et al. (2015). In summary, we attempt to model a combined PB and USV operation to cover specifically designated target areas. To build the proposed model, we use several methods. To model how the USV performs its task, we use a coverage path planning method. We adopt the two-echelon routing problem concept since we will determine the route for the PB (first echelon) and the USV (second echelon). By combining the coverage path planning and the two-echelon routing problem, the model for the combined PB and USV operation can be developed. We also use a time-space network to track the movement of the USV at each time point, including its energy consumption and the movement of the PB.

An example of a case with one PB with multiple USVs is as follows: Suppose that we have a given area that is already being decomposed into a 4x4 grid size. Based on this workspace, suppose that we have several nodes that are selected as the working space for the PB path. As shown in Figure 3.5, with node 1, node 2, node 3, and node 4 as the PB predetermined path, and all the other nodes considered as USV area nodes, nodes 11, 5, 6, and 7 are considered as the target based on their importance. Each target has its own service time (s_i) representing the required time for completing a task there, and a collected value (D_i) after the task has been completed.

doi:10.6844/NCKU202000986



(a) A Real Water Area (b) A Converted Grid Network



(c) Detailed Node Representations

Figure 3.5. Area Information: illustration

The use of a grid network simplifies the USV movement. In particular, we assume a USV can only access from a node to its neighboring node at one time. The region ranges for node 1 comprises cells 11, 2, and 5, and the area range for node 9 is all cells adjacent to it (cells 8, 7, and 10). These ranges determine the USV movement. For example, if a vehicle is located in node 1, the USV will only be able to scan cells 11, 2, and 5. To simplify this problem, we assume to use only one USV in this example. Based on its battery capacity and energy consumption, there are specific options that the USV will have to consider to do the operations, such as staying in the same node or moving back to the PB, in order to reach a target. Information for the traversal time (t_{ij}) of all arcs are known beforehand. Here in our illustrative example, we assume all the service time on a target s_i and the travel time t_{ij} on an edge (i, j) to consume 1 unit of time.

doi:10.6844/NCKU202000986

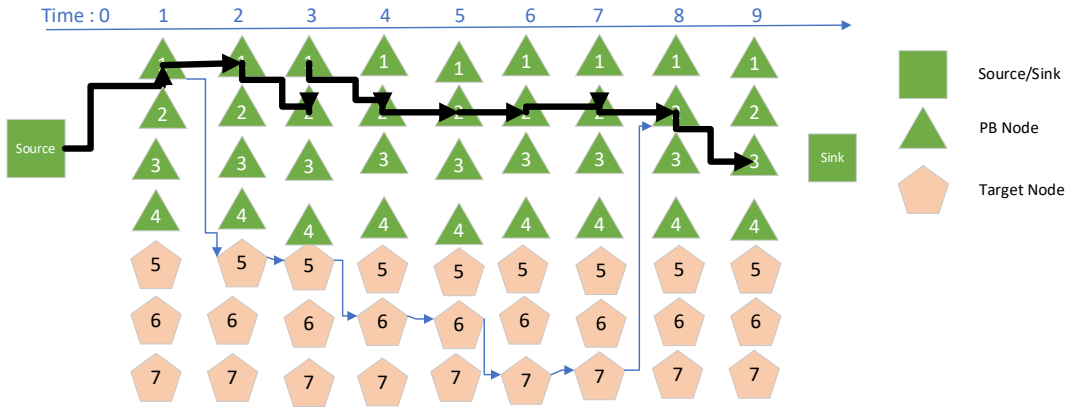


Figure 3.6.Time-Space Network Structure

Suppose the PB, together with the USV, starts from node 1. The USV has its maximum battery capacity and a limited time to operate, and its operating time equals 10 units of time. We seek the route that gives the optimal time to operate to reach the target during the given time. We can try to solve the problem manually. For the first iteration, suppose we make the USV move with the PB to move from the source to node 1 and then launch the USV in node 1. In 10 units of time based on the manual calculation, going to nodes 5, 6, and 7 in general can give us total objective values that are higher than retrieving target 11. This is the nature of the orienteering problem in our research, where we want to obtain the greatest total objectives values as possible in a limited amount of time. Therefore, target 11 could not be retrieved in 10 units of time, but when the operation time-bound was extended to 18 units of time, the optimal route was different. To simplify the illustrations, the time-space network shown in Figure 3.6 and Figure 3.7 is the optimal path for the USV and PB movement.

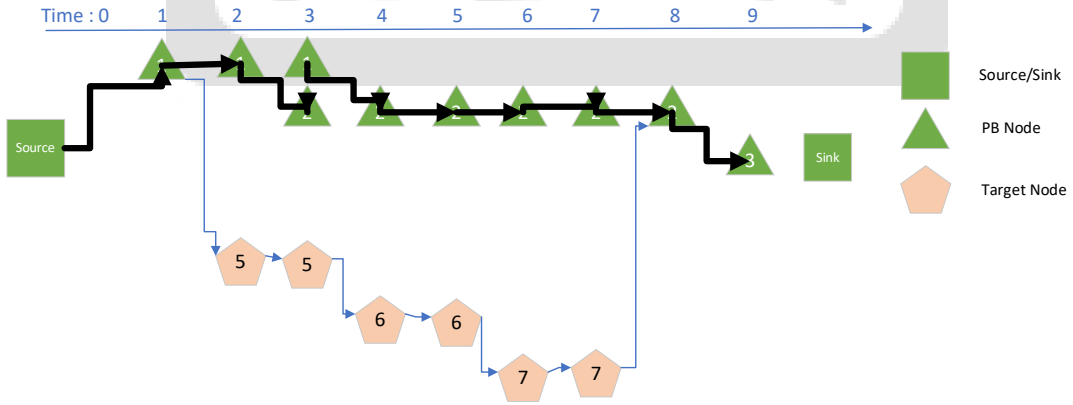


Figure 3.7.Model Result in 10 Sets of Time

doi:10.6844/NCKU202000986

Since both the USV and PB are starting from node 1, it can be seen that due to the differences in speed between the PB and the USVs, each USV was able to reach targets 5, 6, and 7 and to stay in that node for 1 set of time and to return to the PB path (node 2) to continue the operation. Notice that in each operation, the total time is limited so as not to exceed the maximum routing time. In this routing, there are several possibilities that each USV can take to reach the target. For example, the USVs can take off at node 2 and go from that node to reach targets 5, 6, and 7. However, if the USV follows this movement, then the total operating time may be different. However, once the operation time-bound is set to 18 units of time, after the USV and the PB meet in node 2 at time 8, the USV can be launched to retrieve target 11.

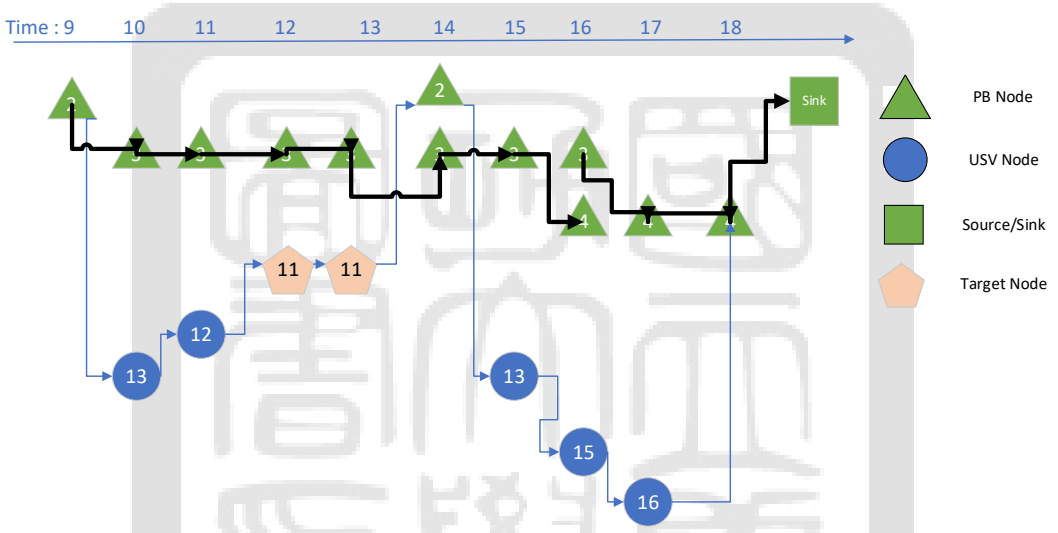


Figure 3.8.Model Result in 18 Sets of Time

Since both USVs and the PB are starting from node 2, the total maximum time needed for every variable to return to the sink node now requires 18 total units of time. This gave us multiple possibilities of how the modeling can be used to find the optimal time needed for operations as well as to generate the optimum time required if the time is limited based on the importance of each target. Facing this situation, we can use the proposed mathematical programming-based approach. After dividing the size of the grid and the number of nodes in the system, we create the time-space network based on the area of space that we intend to analyze.

Let $G=(V,A)$ be a complete undirected graph, where V is the set of all nodes, including one main depot (node 1) and one destination (node n). Let V_p be a subset of V representing

doi:10.6844/NCKU202000986

the rendezvous nodes in the PB path workspace, where V_u is a subset of V representing the cells in the USV workspace, which is acquired from the preprocessing step. Each rendezvous node in the PB space represent a location where the PB can stop to launch the USV.

A set of targets V_T, residing at $|V_T|$ different locations, where each target $i \in V_T$ is associated with an importance value (D_i). The PB workspace and the USV workspace may overlap each other. Let $R_i = \{j | d_{ij} \leq \gamma, j \in V\}$ be a set of nodes adjacent (i.e., accessible in 1 unit of time) from node $i \in V$. Let A be the set of all edges, whose subset $A_p = \{(i', j') | i', j' \in V_p\}$ is a set of all edges located in the PB path, subset $A_u = \{(i", j") | i", j" \in V_u\}$ is a set of all edges located in the USV workspace, subset $A_c = \{(i', j") | j" \in R_{i'}\} \cup \{(i", j') | j' \in R_{i"}\}$ is a set of all edges connecting the PB nodes with the USV workspace, and subset $A_s = \{(i", i") | i" \in V_u\} \cup \{(i', i') | i' \in V_p\}$ is a set of staying edges (self-loop) that indicate that the USV stays at a node $i" \in V_u$ to perform the service or to wait at a node $i' \in V_p$ when necessary. The PB will only move along edges in A_p, while the USV can move along all other edges. We introduce the following assumption for our mathematical programming model:

1. The service times and values for all target cells (nodes) and traversal time for all edges are known constants.
2. The targets are evenly distributed in the USV workplace.
3. The PB can only traverse edges in its network, and the PB path is known beforehand.
4. When the USVs return to the PB, each USV can charge its battery and start the next routing immediately. The time for charging the battery is short and negligible.

3.3. Mathematics Model

3.3.1. Notation

The following notations that being used for the mathematic model are listed below:

doi:10.6844/NCKU202000986

Table 3. 1. Notations for the Mathematical Formulations

Decision Variable	
x_{ijt}^{k}	1, if USV k at a time t start to passing edge (i, j), $\forall(i,j) \in A$; and 0, otherwise
y_{ijt}	1, if the PB at a time t start to passing edge (i, j), $\forall(i,j) \in A_p$; and 0, otherwise
z_{it}^{k}	1, if the PB and USV k at a time t both are at node i, and 0, otherwise
M_{ijt}^{k}	1, if the PB and USV k both move along edge (i, j) in $A_p \cup A_s$, and 0, otherwise
R_{ijt}^{k}	1, if the USV k needs to be recharged along edge (i,j) in $A_p \cup A_c \cup A_s$, and 0, otherwise
w_i	1, if target i is covered, $\forall i \in V_T$, and, 0 otherwise
α_{ijt}^{k}	the initial battery level for USV k to move along edge (i, j) starting at time t
β_{ijt}^{k}	the ending battery level for USV k to move along edge (i, j) starting at time t
Set	
V	Set of all USV and PB workplace nodes, $V = V_u \cup V_p$
V_p	Set of PB path nodes
V_u	Set of USV workspace nodes
A	Set of all workplace edges, $A = A_p \cup A_u \cup A_c \cup A_s$
A_p	Set of edges for the PB path, $A_p = \{(i', j') \mid i', j' \in V_p\}$
A_u	Set of edges for the USV workspace, $A_u = \{(i", j") \mid i", j" \in V_u\}$
A_c	Set of connecting edges between PB and USV nodes, $A_c = \{(i', j") \mid j" \in R_{i'}\} \cup \{(i", j') \mid j' \in R_{i"}\}$
A_s	Set of staying edges (self-loop), $A_s = \{(i", i") \mid i" \in V_u\} \cup \{(i', i') \mid i' \in V_p\}$
A_{source}	Set of starting edges connecting to the origin node γ, $A_{source} = \{(\gamma, i) \mid i \in V\}$
A_{sink}	Set of ending edges connecting to the sink node ψ, $A_{sink} = \{(i, \psi) \mid i \in V\}$
N_i	set of nodes adjacent to the node i within range γ, $N_i = \{j \mid d_{ij} \leq \gamma, j \in V\}$
T	Set of time periods

doi:10.6844/NCKU202000986

Table 3. 2. Notations for the Mathematical Formulations (Cont.)

Parameters of the model	
Δ_{ij}	Travel time for passing edge (i,j), $\forall(i,j) \in A$
s_i	Service time for visit node i, $\forall i \in V_T$
D_i	The value obtained by covering a target node i, $\forall i \in V_T$
b_{ij}	Battery consumptions for passing the edge (i, j), $\forall(i,j) \in A$
B	Maximum battery capacity

3.3.2. Mathematical Objective and Constraints

Given a set of nodes, one USV is deployed to cover all the targets, with support from a PB. The objective function is intended to maximize the number of targets that can be serviced within a limited time period and is shown as Equation (3.1). T is the maximum operating time that is used to set the maximum operating time for each USV, with w_i as the decision variable determining whether the target i is serviced (i.e., covered) and its corresponding value of importance with i, $\forall i \in V_T$.

$$\max \sum_{i \in V_T} D_i w_i \tag{3.1}$$

The constraint shown in Equation (3.2) defines the flow balance for the PB in the PB path.

$$\sum_{(i,j) \in A_p \cup A_{\text{sink}} \cup A_s} y_{ijt} - \sum_{(j,i) \in A_p \cup A_{source} \cup A_s} y_{ji(t-\Delta_{ji})} = 0 \quad \forall i \in V_p, t \in T \tag{3.2}$$

The constraint (3.3) defines the flow balance for the USV k, note that a USV can appear in any node in V.

$$\sum_{(i,j) \in A_p \cup A_u \cup A_s \cup A_c} x^k_{ijt} - \sum_{(j,i) \in A_p \cup A_u \cup A_s \cup A_c} x^k_{ji(t-\Delta_{ji})} = 0 \quad \forall i \in V, t \in T, k \in K \tag{3.3}$$

Constraint (3.4) guarantees a covered target $i \in V_T$ requires some USV k to stay there for at least s_i units of time.

$$\sum_{t \in T} x^k_{iit} \geq s_i w_i \quad \forall i \in V_T, k \in K \tag{3.4}$$

doi:10.6844/NCKU202000986

The constraints shown in equations (3.5)-(3.6) are used to set both the USV and PB to move together, which is represented by a decision variable z_{it}^k in which $i \in V_p$.

$$z_{it}^k \geq \sum_{(i,j)\in A_p \cup A_s \cup A_c} x_{ijt}^k + \sum_{(i,j)\in A_p \cup A_s} y_{ijt} - 1 \qquad \forall t \in T, i \in V_p, k \in K \tag{3.5}$$

$$z_{it}^k \leq \left(\sum_{(i,j)\in A_p \cup A_s \cup A_c} x_{ijt}^k + \sum_{(i,j)\in A_p \cup A_s} y_{ijt}\right)/2 \qquad \forall t \in T, i \in V_p, k \in K \tag{3.6}$$

The constraints in equations (3.7)-(3.8) ensure that the decision variable M_{ijt}^k is going to be triggered (i.e., $M_{ijt}^k = 1$) when both the USV and PB move along the same arc (i.e., $x_{ijt}^k = y_{ijt} = 1$). This will be used in the battery consumption equations.

$$M_{ijt}^k \geq x_{ijt}^k + y_{ijt} - 1 \qquad \forall t \in T, (i,j) \in A_p \cup A_s, k \in K \tag{3.7}$$

$$M_{ijt}^k \leq (x_{ijt}^k + y_{ijt})/2 \qquad \forall t \in T, (i,j) \in A_p \cup A_s, k \in K \tag{3.8}$$

Constraints (3.9)-(3.10) are used to set $R_{ijt}^k = x_{ijt}^k z_{it}^k$ to be equal to 1 when USV *k* and PB meet (i.e., z_{it}^k = 1) and moves along edge (*i*,*j*) (i.e., x_{ijt}^k = 1). This constraint is needed for the battery recharge when USV is in the PB path.

$$R_{ijt}^k \geq x_{ijt}^k + z_{it}^k - 1 \qquad \forall t \in T, (i,j) \in A_p \cup A_s, k \in K \tag{3.9}$$

$$R_{ijt}^k \leq (x_{ijt}^k + z_{it}^k)/2 \qquad \forall t \in T, (i,j) \in A_p \cup A_s, k \in K \tag{3.10}$$

To limit the battery capacity in each arc, constraints (3.11)-(3.13) are used to set the starting battery level (α_{ijt}^k) associated with an arc $(i,j) \in A$ to be no more than the maximum capacity (B) anytime, and become full in the beginning (i.e., x_{ij0}^k=1) or while carried by the PB (i.e., R_{ijt}^k=1).

$$\alpha_{ijt}^k \leq B\, x_{ijt}^k \qquad \forall t \in T, (i,j) \in A, k \in K \tag{3.11}$$

$$\alpha_{ij0}^k = B\, x_{ij0}^k \qquad \forall t \in T, (i,j) \in A_p \cup A_s \cup A_c, k \in K \tag{3.12}$$

$$\alpha_{ijt}^k \geq B\, R_{ijt}^k \qquad \forall t \in T, (i,j) \in A_p \cup A_s \cup A_c, k \in K \tag{3.13}$$

doi:10.6844/NCKU202000986

Constraint (3.14) sets the battery consumption along an arc $(i,j) \in A_u$, and (3.15) ignores that change when the USV is moving with the PB path (i.e., , $x_{ijt}^k = M_{ijt}^k = 1$).

$$\beta_{ijt}^k = \alpha_{ijt}^k - b_{ij} x_{ijt}^k \qquad \forall t \in T, (i,j) \in A_u, k \in K \tag{3.14}$$

$$\beta_{ijt}^k = \alpha_{ijt}^k - b_{ij} (x_{ijt}^k - M_{ijt}^k) \qquad \forall t \in T, (i,j) \in A_p \cup A_s, k \in K \tag{3.15}$$

For each USV node *i* passed by a USV *k*, its entering battery level equals its leaving battery level, as shown in (16). On the other hand, if a USV *k* passes a parent node, it may be recharged (i.e., $R_{ijt}^k = 1$) if necessary by (17).

$$\sum_{(i,j) \in A_u \cup A_c \cup A_s} \alpha_{ijt}^k - \sum_{(j,i) \in A_u \cup A_c \cup A_s} \beta_{ji(t-\Delta_{ji})}^k = 0 \qquad \forall t \in T, i \in V_u, k \in K \tag{3.16}$$

$$\sum_{(i,j) \in A_p \cup A_c \cup A_s} (\alpha_{ijt}^k - BR_{ijt}^k) - \sum_{(j,i) \in A_p \cup A_c \cup A_s} \beta_{ji(t-\Delta_{ji})}^k \le 0 \quad \forall t \in T, i \in V_p, k \in K \tag{3.17}$$

Both USV *k* and the PB are set to go to the sink node, meaning the operation is finished, by the end of the planning horizon, as represented in (3.18)-(3.19)

$$\sum_{t \in T} \sum_{(i,\psi) \in A_{sink}} y_{i\psi t} = 1 \qquad \forall k \in K \tag{3.18}$$

$$\sum_{t \in T} \sum_{(i,\psi) \in A_{sink}} x_{i\psi t}^k = 1 \qquad \forall k \in K \tag{3.19}$$

The constraints shown in equations (3.20) - (3.24) define the domain for each variable.

$$x_{ijt}^k \in \{0,1\} \qquad \forall t \in T, (i,j) \in A_u, k \in K \tag{3.20}$$

$$y_{ijt}, M_{ijt}^k, R_{ijt}^k \in \{0,1\} \qquad \forall t \in T, (i,j) \in A_p, k \in K \tag{3.21}$$

$$R_{ijt}^k \in \{0,1\} \qquad \forall t \in T, (i,j) \in A_p \cup A_s \cup A_c, k \in K \tag{3.22}$$

$$z_{it}^k \in \{0,1\} \qquad \forall t \in T, i \in V_p, k \in K \tag{3.23}$$

$$w_i \in \{0,1\} \qquad \forall t \in T, i \in V_T, k \in K \tag{3.24}$$

For better understanding, there are two sets of cases that will be analyzed in this thesis.

doi:10.6844/NCKU202000986

The first case is when the parent boat path is already known beforehand (C1), and the second case is when only the area for the parent boat is known beforehand (C2).

3.3.3. Case of Defined Parent Boat Path (C1)

In this case, the path for the parent boat is known beforehand. While the exact movement, time for the PB to stop, is not known. Unlike the subsection 3.3.4., in this case, the path for the PB will be used for the movement of the PB, and therefore, the PB doesn't have to search for a feasible path for the PB. The illustration for this case is shown in Figure 3.9.

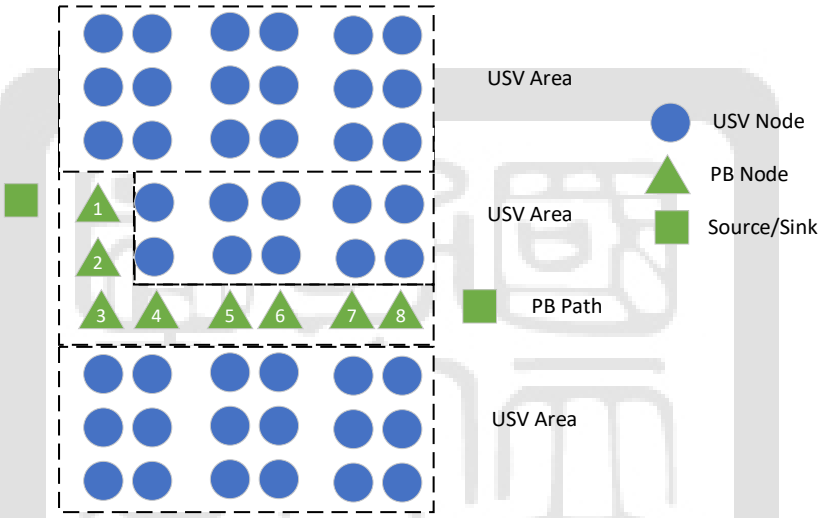


Figure 3.9.Given PB Path in a Network

By defining the path for the PB, there are several parameter changes in the Integer Programming (IP) mathematical formulations that are affected. For the parameter used in the IP model, the variable that is affected are V_p, and V_u. As shown in Figure 3.9., the PB path is defined as a feasible path from an origin node/area (the nodes with numbers) to a destination node/area.

3.3.4. Case of Defined Parent Boat Area (C2)

In this case, only the parent boat area known beforehand. Therefore the model will have to find the movement for its area, and the parent boat also will have to find its feasible path along with the joint operation. Presumably, by having this case and subsection 3.3.3., we can understand how much the computational time for the model can be affected if the parent boat path is not known beforehand. The illustration for this case is shown in Figure 3.10.

doi:10.6844/NCKU202000986

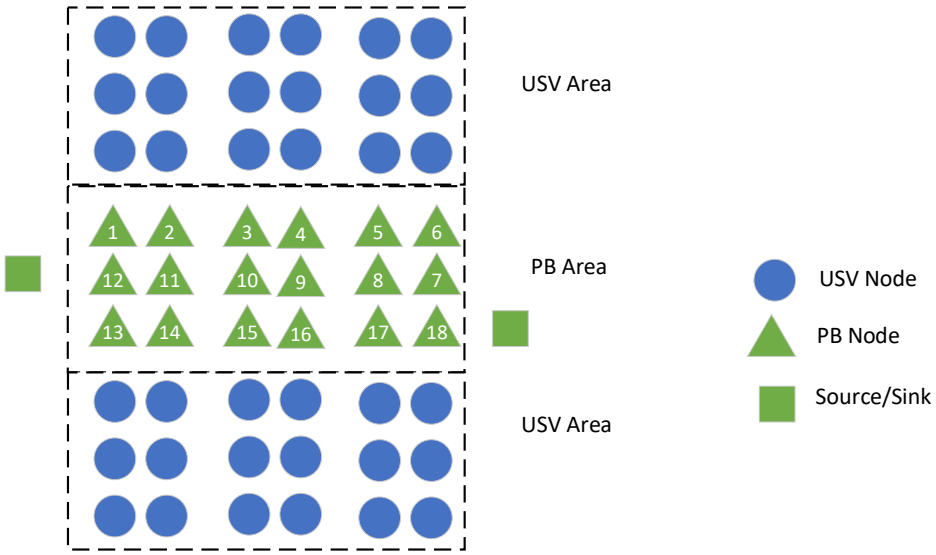


Figure 3.10. Given PB Area in a Network

There are several parameter changes in the Integer Programming (IP) mathematical formulations that are affected in this case. For the parameter used in the IP model, the variable that is affected are V_p, and V_u. As shown in Figure 3.10., the PB path is not yet defined. The only information known is the origin point (V_{source}) and destination point (V_{sink}).

doi:10.6844/NCKU202000986

Chapter 4

HEURISTICS AND ALGORITHM PROCEDURES

This chapter describes the procedures for the heuristics and algorithms that have been developed from the IP model. Section 4.1 describes the Iterative Clustering Heuristic (ICH) procedures. Section 4.2 describes the Iterative Local Search Algorithm procedures (ILS). Section 4.3 describes the Sequential Segmentation Heuristic (SSH).

4.1. Iterative Clustering Heuristic

Since we are dealing with an NP-hard problem, the IP formulation in Section 3 requires long computational time for large instances, even by the state-of-the-art commercial software such as GUROBI. To calculate a good solution within a shorter time, we propose a heuristic approach for the joint operation model in this Section. There are three main steps in our proposed heuristic. The first step is to apply a *k*-means algorithm to cluster the workspace for each USV. We determine a good sequence for those clusters based on the distance of the geometric center to the depot. Finally, we solve the simplified IP formulation associated with each cluster to determine optimal USV routing for target covering and rendezvousing with PB by GUROBI.

The procedures for the Iterative Clustering Heuristic is explained in Figure 4.1. After the algorithm reads the problem data (i.e., all the information associated with nodes and arcs), we conduct a quick preliminary experiment to determine a good number of clusters (a multiple of *k*). Then we apply a *k*-means algorithm to cluster USV nodes so that each cluster forms a connected subnetwork of similar size (i.e., number of USV nodes including targets) and is assigned to a USV. A good cluster visiting sequence is determined by the increasing order of the distance between the geometric cluster center to the depot. For each cluster, we formulate a smaller IP to solve by GUROBI for the USV routes as well as its rendezvous with PB. Finally, we connect PB nodes between clusters, as well as the USV routes inside each cluster, which defines all the USV routes and PB route. The entire procedure contains iterative runs of node clustering and routings for USVs. Thus we name it as an Iterative Clustering Heuristic (ICH).

doi:10.6844/NCKU202000986

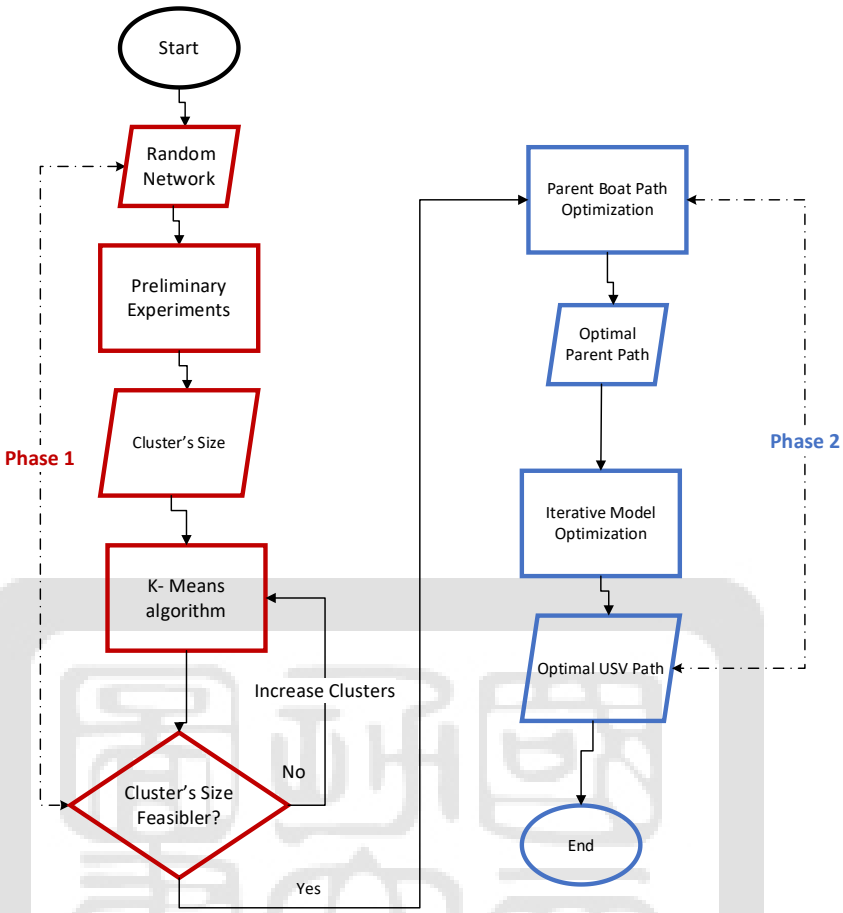


Figure 4.1. Iterative Clustering Heuristic Framework

The two phases in the Iterative Clustering Heuristic is explained in subsection 4.1.1 for phase 1 procedures, and subsection 4.1.2 for phase 2 procedures.

4.1.1. Heuristic Approach Phase 1

With the main objective of phase 1 of the heuristic approach to find the optimal parent path that can be used as an input for phase 2. The first step in the phase 1 is the preliminary experiments to calculate the number of clusters. The parameter for the number of nodes in each cluster for the ICH is calculated using the preliminary experiments. There are several assumptions used for the preliminary experiments. First, it is required that the total number of clusters to be around the multiple of the number of USVs used. This value is used to provide a balanced workspace for each USV. If 2 USVs are used for the operation, then the total number of clusters is from 2 up to 4 (e. g 2*2), and so on. The second assumption is that each cluster should take into consideration the USV's battery capacity in order to

doi:10.6844/NCKU202000986

minimize the recharging frequency. Applying each cluster to limit the movement of a USV has advantages related to speeding up the computational time but also can be a hindrance if the workspace is too small because it requires returning to the PB to move to another cluster. To calculate the size of a cluster, a greedy algorithm is used to calculate the route and the range within the USV will need to recharge with the PB. The purpose of our preliminary experiments is to seek good parameter settings for ICH to attain a good compromise between efficiency and effectiveness. The most time-consuming procedure in ICH is to solve routing for each USV cluster by GUROBI. The larger a USV cluster is, the better solution we can get but with longer computational time. Here we simulate sample grid networks, and try out the relation between the computational time and time and the size of USV nodes in solving the orienteering problems by GUROBI, to determine an appropriate size (e.g., number of USV nodes) of a USV cluster. We then use the test result to determine the number of clusters (in this thesis, a multiple of *k*) for ICH.

For example, we may simulate a set of 6x10 random networks containing 1x10 PB nodes in the bottom layer and 5x10 USV nodes in upper layers, randomly select some (e.g., 30 among 50) USV nodes as targets where each target has its given service time and value. With a given battery capacity (e.g., B=100) for each USV and the battery consumption on each target and along each edge, we construct an orienteering problem IP formulation to test the best routing for a USV within its range (i.e., before its battery level reaches 0). Based on the tests on simulated networks, we can observe the average number of USV nodes (e.g., 35) to be traversed by a USV without recharging. Suppose there are a total 300 USV nodes, and there are 5 USVs, then we may divide the USV nodes into 10 clusters. This is because among the 5, 10, and 15 clusters, 10 clusters give the best compromise (10*35=350 is closer to 300 than 5*35 and 15*35). Figure 4.2 illustrates a clustering example.

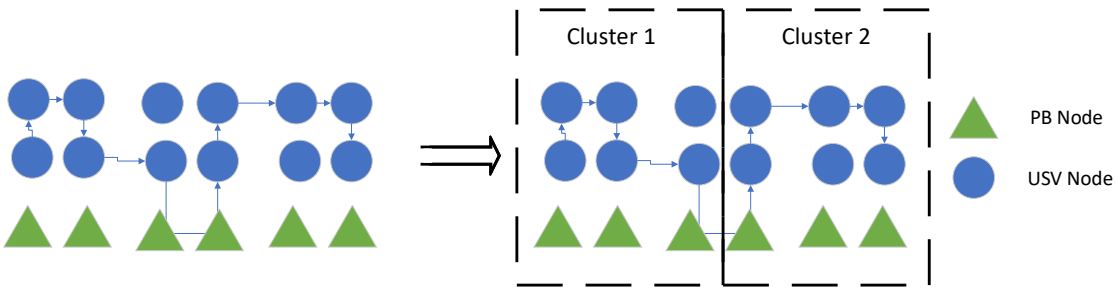


Figure 4.2. USV Clustering and Parent Boat Area

doi:10.6844/NCKU202000986

Each size of the network is tested on the commercial solver. The assumptions for the preliminary tests are applied. Figure 4.2 explains the possibilities that depending on the battery level capacity for the USV movement is an important factor for the recharging decision. Since recharging requires time, each cluster is constructed to minimize the number of recharging within each cluster.

Our ICH algorithm is a divide-and-conquer heuristic. Based on the result of preliminary experiments, we have calculated a good cluster number (k^*). We then divide the entire workspace into k^* clusters, so that we can evenly assign k USVs for these clusters, and solve the corresponding simplified IP model associated with each cluster for its best USV routing by GUROBI. Since the size of each IP model is much smaller than the original IP model, as shown in Section 3, the entire procedure requires less computational time. After the number of clusters has been decided, the next step will be to find the parent boat path. During this development, the information needed is the center of gravity for each cluster, the feasible time needed for each USV to operate as well as the feasible area for the parent boat to work around. This step is very important for the case of C2 to find the optimal PB path that will be used to find the route for PB. An illustrative example of the random network file is shown in Figure 4.3.

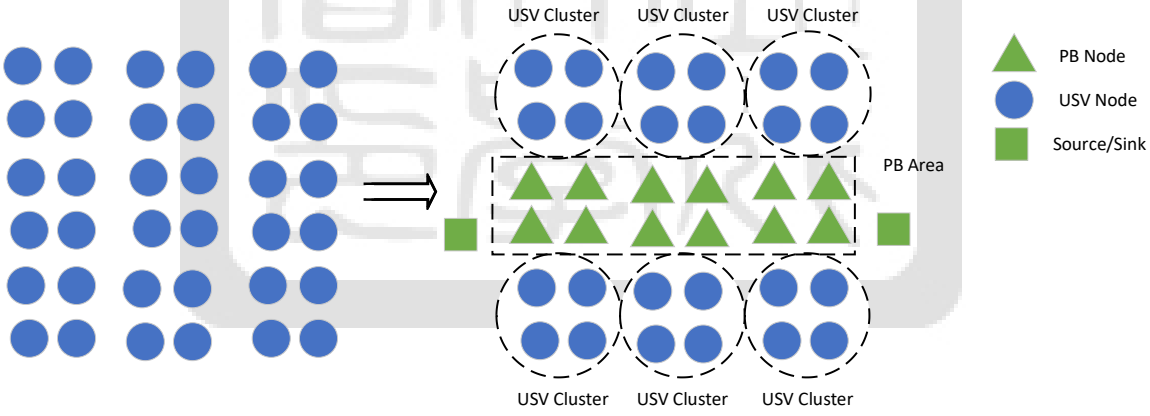


Figure 4.3. USV Clustering and Parent Boat Area

From the original network, each node is going to be computed into the number of clusters that the user defines. For each of these clusters, the center of gravity algorithm is used to find the center in each cluster. Here we suggest using the K-means algorithm to determine the composition of k^* clusters, since we assume the targets are somewhat fairly

doi:10.6844/NCKU202000986

distributed on the USV workspace. If the targets are not evenly distributed, one needs to design better clustering methods. After the clusters are formed, we can estimate a good visiting sequence for the PB, by the increasing order of the distance of the cluster center to the depot. This algorithm requires several pieces of information, including the set of nodes (N) required number of clusters ($n\_clusters$) and each node coordinate ($n_{coordinate}$). From this algorithm, we can obtain the cluster for each node as well as its label information that will be the main domain for each USV. The next step will be to measure the center of gravity in each cluster and calculate the feasible time each USV will have to operate in each cluster.

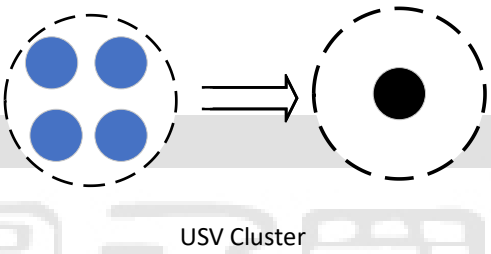


Figure 4.4. Center of Gravity in Each USV Cluster

In each cluster of USV areas, the center of gravity will be calculated to obtain the feasible result that will allow each USV to serve that position and thus become a constrain for the parent boat to follow. Based on these clusters, each USV is assigned to work in a cluster. Therefore in each USV will only have to deal in its own workspace and to save the time for the searching process. Since we do not know the total time each USV will have to spend in each cluster, a feasible way to calculate it is by using an approximation algorithm. Approximation algorithms are efficient algorithms that can find approximate solutions to NP-hard optimization problems with provable guarantees on the distance of the returned solution to the optimal one. The algorithm that we choose is a *Greedy Randomized Adaptive Search* (GRASP).

The objective of the Greedy Randomized Adaptive Search Procedure is to sample stochastically greedy solutions repeatedly, and then use a local search procedure to refine them to a local optimum. The strategy of the procedure is centered on the stochastic and greedy step-wise construction mechanism that constrains the selection and order-of-inclusion of the components of a solution based on the value they are expected to provide. GRASP algorithm is used to solve the *Travelling Salesman Problem* (TSP) problem with each target node as the destination.

doi:10.6844/NCKU202000986

Table 4.1. Greedy Randomized Adaptive Search Procedure

Greedy Randomized Adaptive Search
Input Data: n
Initialization: *maxIterations* = 100, *maxNoImprove* = 50, $\alpha = 0.3$, *currentIteration* =0
1: $S_{best} \leftarrow$ Constructrandomsolution()
2: **While** *currentIteration* < *maxIterations*:
3: $S_{candidate} \leftarrow$ GreedyRandomizedConstruction(n, α)
4.: $S_{candidate} \leftarrow$local_search($S_{candidate}$, n, *maxNoImprove*)
5.: **if** cost($S_{candidate}$) ≤ cost(S_{best}):
6: $S_{best} \leftarrow S_{candidate}$
7: *currentIteration = currentIteration +1*
Result : S_{best}

As shown in Table 4.1, the input needed in this algorithm is the current clusters dataset, and the nodes coordinate (nodes). During the initialization phase, several variables are defined, such as maximum iterations, greediness factor (α), and maximum iterations in the local search. The α here is the parameter that ranges from 0 to 1 with 0 to be more greedy and 1 to be more generalized. The most important part of the GRASP algorithm is within the greedy randomized construction procedure.

With time in each center of gravity obtained, the next procedure is to solve the model with the solver. The implementation of the algorithm results are shown in Figure 4.5, with each cluster has 1 target that is considered by the model. In each cluster, it now has an approximation of time spent in each cluster as a result obtained from the GRASP algorithm.

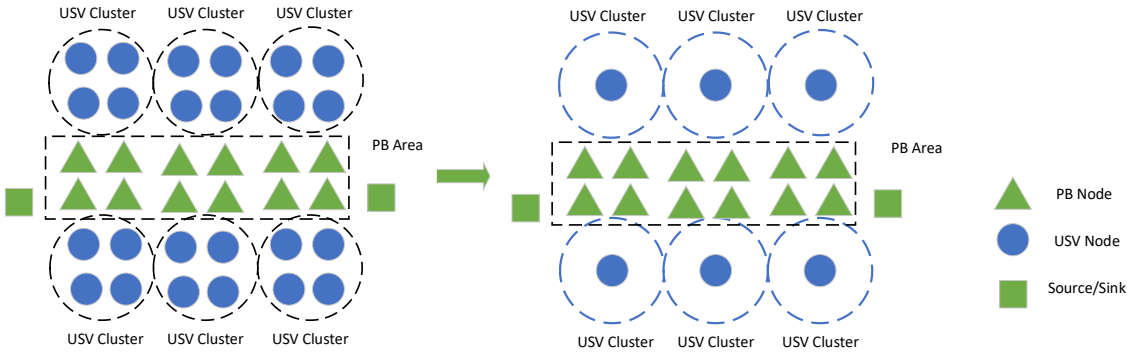


Figure 4.5. Center of Gravity in Each Cluster Mechanism

doi:10.6844/NCKU202000986

The algorithm in Table 4.2. provides the pseudocode of the Greedy Randomized Construction function. The function involves the step-wise construction of a candidate solution using a stochastically greedy construction process. The function works by building a Restricted Candidate List (RCL) that constraints the components of a solution that may be selected from each cycle.

Table 4.2. Greedy Randomized Construction Procedure

Greedy Randomized Construction
Input Data: nodes, α
Initialization: $S_{candidate}$ =∅
1: **While** $S_{candidate}$ ≠nodes.size
2: $F_{costs} \leftarrow \emptyset$
3: **For** each F_i *in* $S_{candidate}$
4.: $F_{costs} \leftarrow$ cost_of_adding_candidate($S_{candidate}$, F_i)
5: $RCL \leftarrow \emptyset$
6: $F_{costsmin} \leftarrow MinCost(F_{costs})$
7: $F_{costsmax} \leftarrow MaxCost(F_{costs})$
8: **For** each j in F_{costs}:
9.: ***if*** $j \leq F_{costsmin} + \alpha\ (F_{costsmax} - F_{costsmin})$
10: $RCL \leftarrow F_i$
11: $S_{candidate} \leftarrow SelectRandom(RCL)$
Result : $S_{candidate}$

With the total time obtained, the next procedure is to solve the model with the solver. The network that will be used in the solver can be seen from Figure 4.6. with 6 potential USV nodes in the USV area and parent boat area.

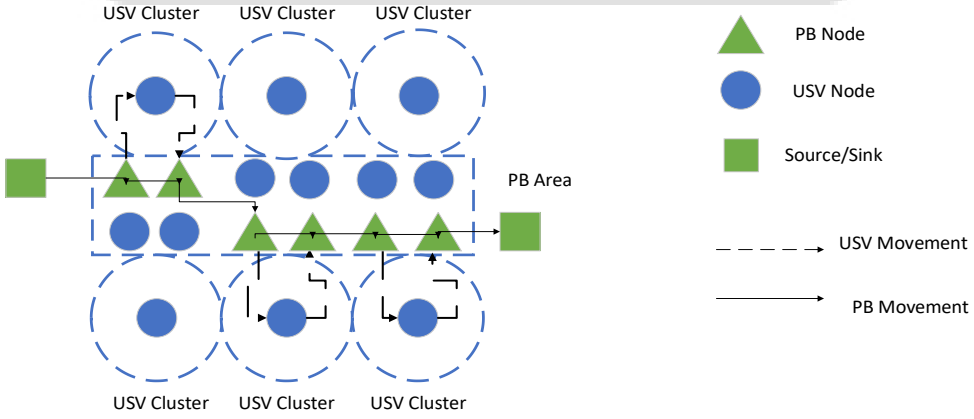


Figure 4.6. GRASP-based Routing

doi:10.6844/NCKU202000986

The result obtained from this model is the optimal path for the parent boat movement but not for each USV movement in each cluster. The movement for both vehicles may not be optimal and have the optimal total operation time since the target in each cluster is simplified. Therefore the result for the PB area here is an optimal path for the PB, but the detail of the movement will need to be analyzed in phase 2. As shown in Figure 4.6, not all center of gravity target will be rescued by the USV due to the orienteering nature of the problem. Therefore with the obtained optimal parent boat path, we move to phase 2 of the heuristic.

4.1.2. Heuristic Approach Phase 2

The main objective of phase 2 is to obtain USV optimal path with the parent path obtained in phase 1 as a constraint. This phase can be applied for both cases C1 and C2 as this is a necessary procedure to find the route for both PB and USVs. With the clustering of USV's cluster, in each iteration, the model will use different datasets to solve each cluster individually. To understand this mechanism better, Figure 4.7, describes a closer look in each USV cluster from Figure 4.6. Assuming that each cluster has its identification number (cluster 1, cluster 2, etc.). Now each cluster will function as a self-contained problem with a parent boat route as well as USV workspace with its target.

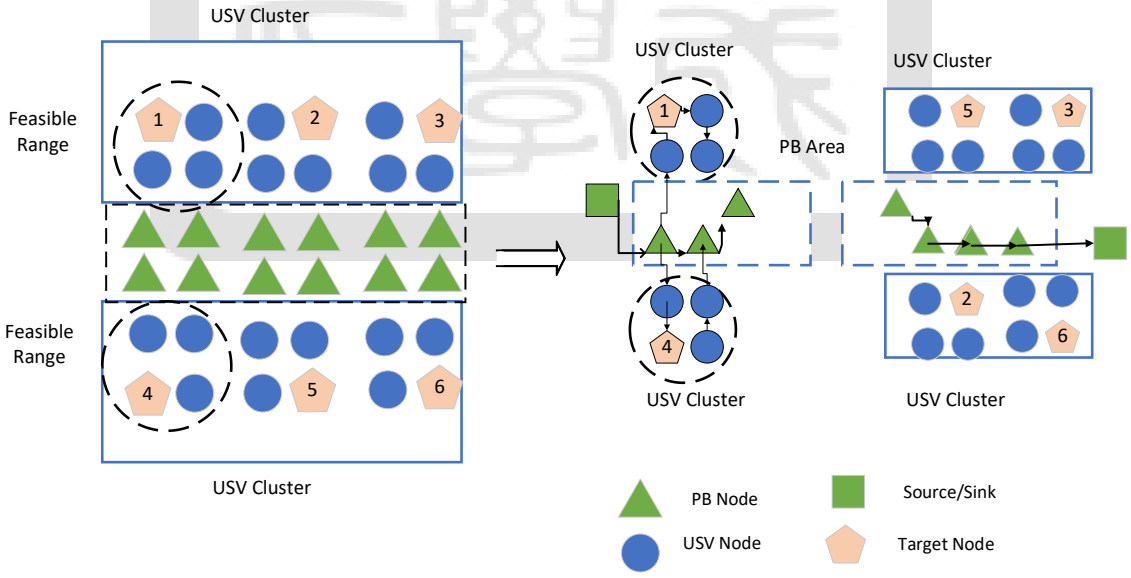


Figure 4.7. Cluster Iterations in Phase 2

doi:10.6844/NCKU202000986

The main difference compared to phase 1 is that in each USV workspace, the model was previously simplified for the parent boat to solve easily, but in phase 2 now that the original model has been broken down into smaller cases. The problem should operate faster compared to the original model, so that the newly acquired target in phase 1 will be changed to match the original model. Aside from that, with the new parent boat path obtained in phase 1, the area for the parent boat to operate except the optimal path will have to be traverse using each USV. Thus the model that will be used in phase 2 is represented in Figure 4.8.

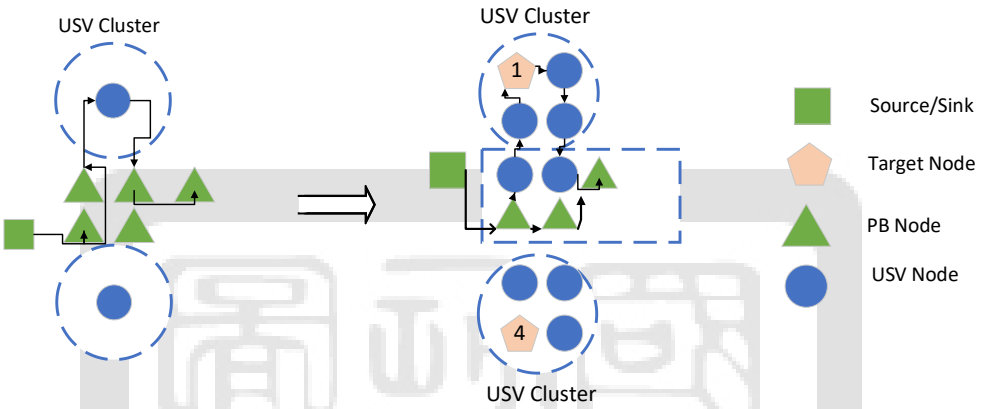


Figure 4.8. Model Layout conversion from phase 1 to phase 2

As can be seen from Figure 4.8, although we already have the optimal path for the parent boat, each USV may require battery swapping. Thus the constraints used in phase 2 is only for the area of the parent boat, and the parent boat can have different result compared to phase 1. In each iteration, the cluster will be solved using the model with the battery consumption consideration as well as the time needed in each cluster will be carried on to the next iteration and become a new constrain for the model to solve the operation within the given time limit. With the procedure's completion, we obtain an optimal path in each USV cluster as well as the parent path.

4.2. Iterated Local Search Algorithm

This section describes our proposed algorithm, Iterated Local Search (ILS). We first introduce a greedy algorithm for generating an initial solution. ILS further improves the initial solution.

doi:10.6844/NCKU202000986

4.2.1. Algorithm Framework

Our algorithm divided into three parts, finding the USV path, finding the PB path, and improving the solution. The idea of this algorithm is based on the iterative local search for team orienteering problem used by Gunawan et al. (2018). However, there are a few key differences that will be explained in section 4.2.2.

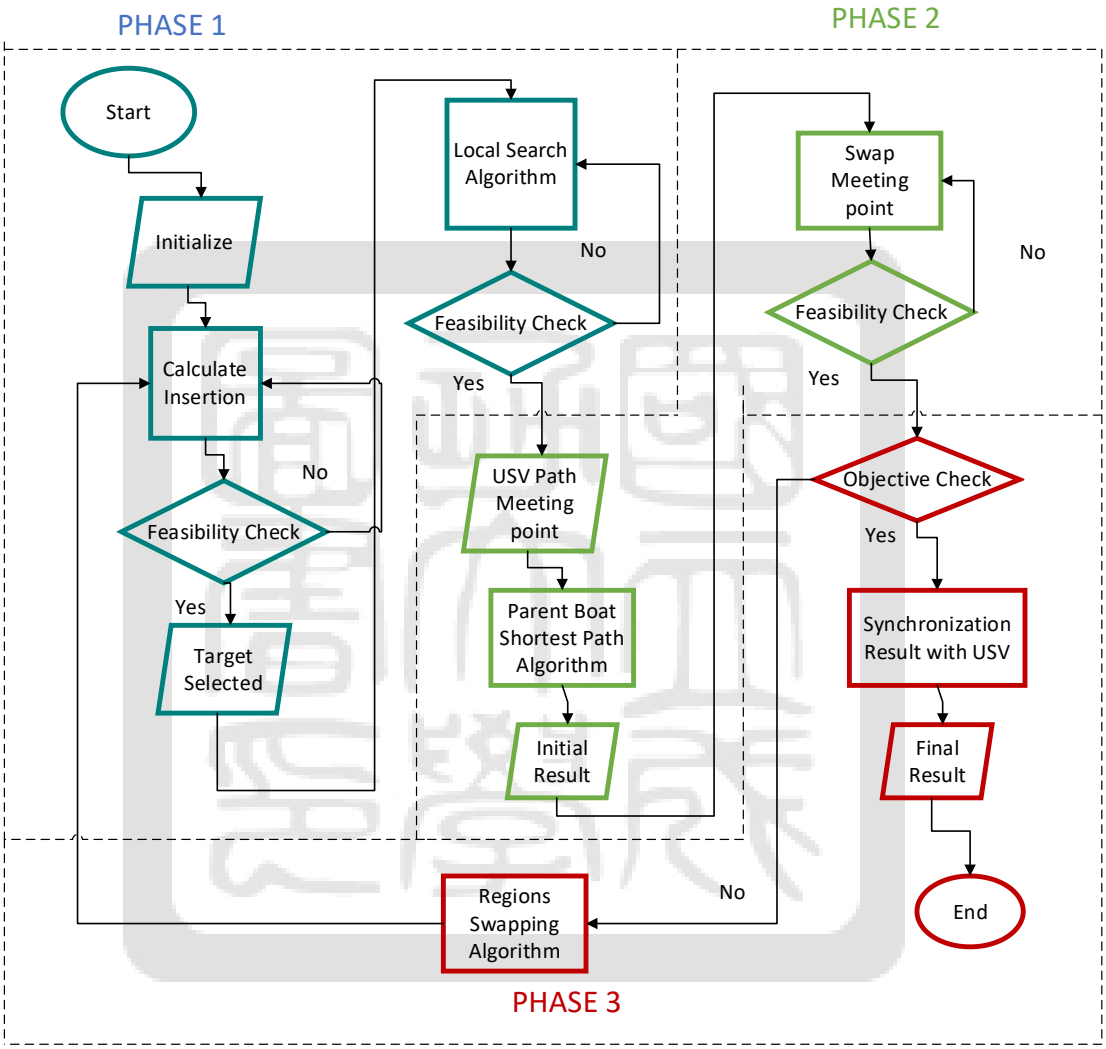


Figure 4.9. Iterated Local Search Algorithm Framework

To better understand the greedy algorithm, this section is divided into 3 phases, as described in Figure 4.9. The focus of phase 1 is to obtain the optimal USV path based on the initial information. During phase 2, the goal is to obtain the optimal parent boat path that can satisfy the meeting points generated in phase 1 and phase 3 to improve the result obtained and redo the mechanism to obtain a higher objective result. Both phase 1 and phase 2 can be

doi:10.6844/NCKU202000986

used to solve the cases (C1 and C2). For the implementation of phase 3, since in the case of C1, the PB path is already defined, this algorithm is not needed, however, this mechanism is very important for case C2.

The pseudocode in Table 4.3 shows the procedure of phase 1 to phase 3 in the algorithm function. The inputs needed in this algorithm are generated from the network with $V = V_u \cup V_p$ as a set of nodes, $A = A_u \cup A_p \cup A_c \cup A_s$ as a set of edges, T_{limit} as the parameter for the time limit of the operation, B as the battery level capacity, D_i as the importance value for each target with i ($\forall(i) \in V_T$), and K as the number of USVs.

Table 4. 3. Iterated Local Search Algorithm

Iterated Local Search Algorithm
Input Data: V , A, T_{limit}, B, D_i, K
Initialization: $V_{removed} = \emptyset$
1: $graph \leftarrow$ constructgraph($V_{removed}$, V, A)
2: $S_{initial}, T_{initial} \leftarrow Reroute(graph, T_{limit}, B, D_i, K)$
3.: ***For*** *each j in* $S_{initial}$:
4.: $resetgraph(V, A)$
5: $graph \leftarrow$ constructgraph(j, V, A)
6: $S_{candidate}, T_{candidate} \leftarrow Reroute(graph, T_{limit}, B, D_i, K)$
7: ***if*** $S_{candidate} \geq S_{initial}$ && $T_{candidate} \leq T_{initial}$:
8.: $V_{removed} \leftarrow add$(j)
9: $S_{initial}, T_{initial} \leftarrow T_{candidate}, S_{candidate}$
10: $resetgraph(V, A)$
11: $graph \leftarrow$ constructgraph($V_{removed}$, V, A)
12: $S_{solution}, T_{solution} \leftarrow Reroute(graph, T_{limit}, B, D_i, K)$
Result : $S_{solution}, T_{solution}$

With the input, the first step of the algorithm will be to construct the network based on this initial graph. The initial solution is constructed using the steps stated in Figure 4.9. by rerouting the USVs and the parent boat. From the initial parent boat area, the area swapping mechanism is conducted by removing the node in parent boat initial route solution respectively to search for a better solution and thus constructing the new solution with the

doi:10.6844/NCKU202000986

nodes that are chosen to be removed. To understand better how each *Reroute* procedure is conducted. The following procedures are the steps taken to obtain the solution.

4.2.2 Iterated Local Search Algorithm Phase 1

The goal of phase 1 is to obtain a feasible USV path, using the network generated from the random network generator. There are several assumptions that we applied during this phase. First, the parent boat is expected to arrive at the meeting point between the parent boat and the USV if the parent boat is not able to meet at the designated point, then the USV will have to wait and thus postpone its operation. The priority of the target chosen by the USV will follow the equation below:

$$C_i = \frac{D_i}{dist_{(i-1)i}} \qquad \forall i \in V_T \tag{4.1}$$

Equation 4.1 requires D_i as the important points in each target i, $\forall(i) \in V_T$ with the value given from the network. Variable $dist_{(i-1)i}$ is the distance from the previous node to the chosen target. This variable is dynamically decided during the algorithm, depending on the current location of the USV.

The value C_i will be used to choose the priority for each target during the insertion algorithm. From the set of targets selected (MOVE and SORT), a greedy algorithm is used to find the feasible path for each USV with battery consideration if the USV cannot finish the operation and thus return to the nearest parent boat path. The selected nodes, therefore, are improved for the algorithm to find a feasible path with higher total values using a local search algorithm (SWAP, REPLACE, and INSERT). The result generated from phase 1 is the path for the USV, as well as the meeting points required for phase 2. There are a set of abbreviations that are described below in this algorithm:

1. MOVE: Moving one node from one USV to another USV. The procedure of this mechanism in our model is conducted by moving each node in 1 USV to another USV and performing a local search to find a better solution.

doi:10.6844/NCKU202000986

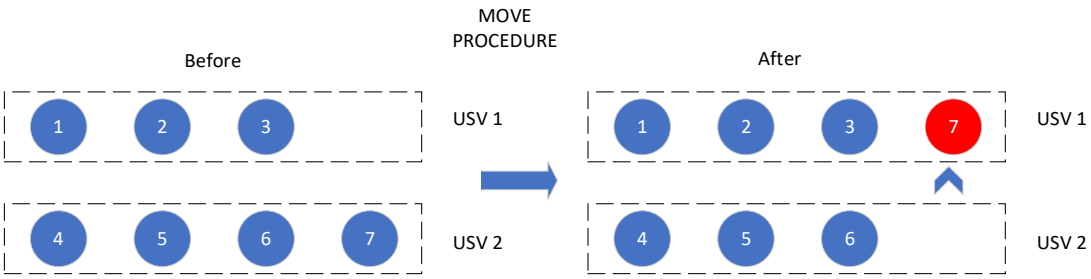


Figure 4.10. MOVE Procedure

The pseudocode in Table 4.4 shows the procedure of MOVE mechanism in *greedyUSV* function. The inputs needed in this algorithm are the graph generated from the network with $graph$ as the network file, $V_{selected}^{k}$ as the nodes selected or the scheduled nodes, T_{limit} as the parameter for the time limit of the operation, K as the number of USVs, B as the battery level capacity, $S_{candidate}$ as the previous routing result, and $T_{candidate}$ as the previous highest time-bound of operation found.

Table 4.4. MOVE Local Search Algorithm

MOVE Local Search Algorithm
Input Data: $T_{limit}, B, graph, V_{selected}^{k}, S_{candidate}, T_{candidate}, D_i, K$
Initialization:
1: $S_0, T_0 \leftarrow S_{candidate}, T_{candidate}$
2: ***For*** *each* k *in* K:
3: ***For*** *each* $candidate1$ *in* $V_{selected}^{k}$:
4: ***For*** *each* $candidate2$ *in* $V_{selected}^{k+1}$:
5: ***if*** $D_{candidate1} \leq D_{candidate2}$:
6: $V_{selected}^{k} \leftarrow shuffle(V_{selected}^{k+1}, V_{selected}^{k})$
7: $S_{candidate}, T_{candidate} \leftarrow greedyUSV(graph, T_{limit}, B, V_{selected}^{k})$
8.: ***if*** $S_{candidate} \geq S_0$ && ***if*** $T_{candidate} \leq T_0$:
9: $S_0, T_0 \leftarrow S_{candidate}, T_{candidate}$
10: $V_{candidate}^{k} \leftarrow V_{selected}^{k}$
Result : $V_{candidate}^{k}$

With the input, as stated previously, the procedures of the algorithm will be used to

doi:10.6844/NCKU202000986

to swap the selected node from USV *k* to another USV *k+1* that has a higher value and then recalculated. For each result that can give a higher or equal total objective value with lower or equal total time-bound of operation will be selected as the selected nodes from the MOVE operation.

2. SORT: Rearranging the sequence of the scheduled target visit. The SORT mechanism is conducted by evaluating the distance between each node to the location of the USV. Therefore the priority for each target in this scenario may be changed to find higher total objective values.

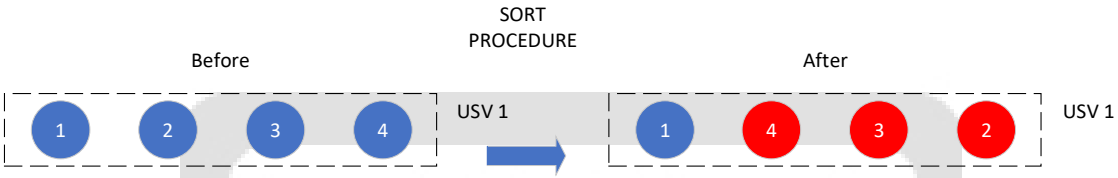


Figure 4.11. SORT Procedure

The pseudocode in Table 4.5 shows the procedure of SORT mechanism in *greedyUSV* function. The inputs needed in this algorithm are $graph$ as the network file, $V_{selected}^{k}$ as the nodes selected or the scheduled nodes, K as the number of USVs, T_{limit} as the parameter for the time limit of the operation, B as the battery level capacity and $S_{candidate}$ as the previous routing result from each USV.

Table 4.5. SORT Local Search Algorithm

SORT Local Search Algorithm

Input Data: $graph, T_{limit}, B,, V_{selected}^{k}, S_{candidate}, K$

Initialization: $V_{origin} = V_p[0], sortedresult = \emptyset, sorted = \emptyset$

1: ***For*** *each k in K*:

2: ***For*** *each j in* $V_{selected}^{k}$:

3: $reach \leftarrow shortestpath\_greedy(graph, V_{origin}, j)$

4.: $sorted \leftarrow add([j, reach])$

5: $sortedresult \leftarrow sorted.sort(key = val[1])$

6: $V_{candidate}^{k} \leftarrow sortedresult$

Result : $V_{candidate}^{k}$

doi:10.6844/NCKU202000986

For each USV, the distance between the position of the USV to the target is calculated. This result is then reordered based on the sequence from the minimum to the maximum distance. Therefore the order of the sequence will be set from the nearest target to the furthest target.

3. INSERT: Insert nodes into a USV. Depending on the variable T_{limit} (time limit of operation) each node that is inserted into the scheduled target will be evaluated. If more time is left before the operation end, then it is possible for the highest multiple or single nodes left to be inserted and evaluated.

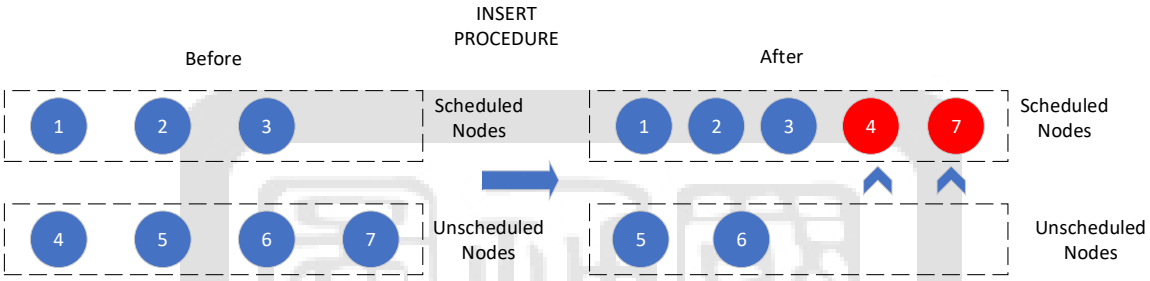


Figure 4.12. INSERT Procedure

The pseudocode in Table 4.6, shows the procedure of INSERT mechanism in *greedyUSV* function. The inputs needed in this algorithm are $graph$ as the network file, $V_{selected}^{k}$ as the nodes selected or the scheduled nodes, T_{limit} as the parameter for the time limit of the operation, K as the number of USVs, B as the battery level capacity and D_i as the importance value for each target with $i \in V_T$,

Table 4.6. INSERT Local Search Algorithm

INSERT Local Search Algorithm
Input Data:$graph, T_{limit}, B, V_{selected}^{k}, D_i, K$
Initialization: $C_{max} = 0$, V_{origin}=$V_p[0]$, $T_0 = 1000$, B =1000
1: ***For*** *each* k *in* K:
2: ***For*** *each* i *in* $V_{selected}^{k}$:
3: $reach, B \leftarrow shortestpath(graph, V_{origin}, i, \mathrm{B}, T_0)$
4: $B_2 \leftarrow returnshortestpath(graph, (i+1))$

doi:10.6844/NCKU202000986

Table 4.7. INSERT Local Search Algorithm (Cont.)

5.:	$C_i \leftarrow D_i/reach$
6:	$\boldsymbol{if}\ C_i \geq C_{max}$ && $B > B_2$:
7:	$C_{max} \leftarrow C_i$
8:	$V^k_{candidate} \leftarrow add(\mathrm{i})$

Result : $V^k_{candidate}$

For each USV, the distance between the position of the USV to the target is calculated using equation (4.1). This distance is used as a variable within the priority function with the values of each target. If the traveled distance is feasible and has a high importance value, the target will be selected as a candidate target that will be inserted.

4. SWAP: Exchanging nodes within one USV. During the SWAP mechanism, each node is chosen according to the highest importance value to another higher importance value with hopes of finding less makespan to the initial solution.

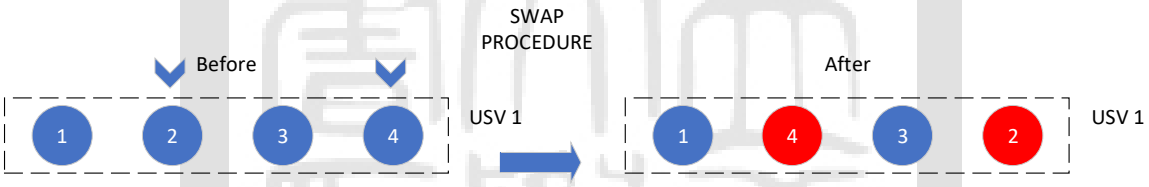


Figure 4.13. SWAP Procedure

The pseudocode in Table 4.7, shows the procedure of SWAP mechanism in *greedyUSV* function. The inputs needed in this algorithm are $graph$ as the network file, $V^k_{selected}$ as the nodes selected or the scheduled nodes, T_{limit} as the parameter for the time limit of the operation, B as the battery level capacity, D_i as the importance value for each target with i ($\forall(i) \in V_T$), K as the number of USVs, $S_{candidate}$ as the previous routing result, and $T_{candidate}$ as the previous highest time-bound of operation found.

doi:10.6844/NCKU202000986

Table 4.8. SWAP Local Search Algorithm

SWAP Local Search Algorithm

Input Data: $T_{limit}, B, graph, V^k_{selected}, S_{candidate}, T_{candidate}, D_i, K$

Initialization:

$S_0, T_0 \leftarrow S_{candidate}, T_{candidate}$

For *each* k *in* K:

    For *each* i *in* $V^k_{selected}$:

        For *each* j *in* $V^k_{selected}$:

            if $i \neq j$ && $D_i \leq D_j$:

                $V_{selected} \leftarrow shuffle(i,j)$

            $S_{candidate}, T_{candidate} \leftarrow greedyUSV(graph, T_{limit}, B, V^k_{selected})$

            if $S_{candidate} \geq S_0$ && ***if*** $T_{candidate} \leq T_0$:

                $S_0, T_0, V^k_{candidate} \leftarrow S_{candidate}, T_{candidate}, V^k_{selected}$

Result : $V^k_{candidate}$

For each USV, the target assigned in each USV is calculated and swapped to another target. The method is conducted by evaluating each node with another target that has a higher value compared to the selected node. For the result that can give a higher or equal total objective value with lower or equal total time-bound of operation will be selected as the selected nodes from the SWAP operation

5. REPLACE: Exchanging one scheduled node with an unscheduled node. Each node within the scheduled nodes is evaluated to the highest value within the unscheduled nodes and evaluated by using the feasibility check.

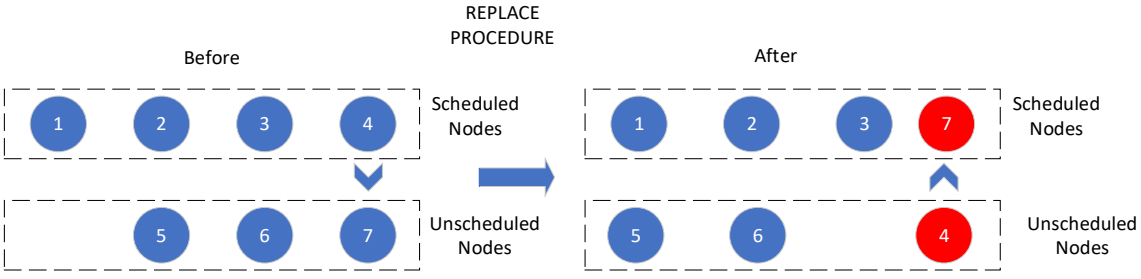


Figure 4.14. REPLACE Procedure

doi:10.6844/NCKU202000986

The pseudocode in Table 4.8, shows the procedure of REPLACE mechanism in *greedyUSV* function. The inputs needed in this algorithm are $graph$ as the network file, $V^k_{selected}$ as the nodes selected or the scheduled nodes, T_{limit} as the parameter for the time limit of the operation, B as the battery level capacity, K as the number of USVs, D_i as the importance value for each target with i ($\forall(i) \in V_T$),

Table 4.9. REPLACE Local Search Algorithm

REPLACE Local Search Algorithm

Input Data: $graph, B, T_{limit}, V^k_{selected}, D_i,\ V_T, K$

Initialization: $V^k_{visited} = \emptyset, S_0 = 0, T_0 = 10000$

1: ***For*** *each* k *in* K :

2: ***For*** *each* $Index_1$ *in* $V^k_{selected}$,:

3: ***For*** *each* $Index_2$ *in* V_T,:

4: ***if*** $Index_1$*not in* $(V^k_{selected} \cup V^k_{visited})$ && $D_{Index_1} \leq D_{Index_2}$:

5: $V^k_{selected} \leftarrow shuffle(Index_1, Index_2)$

6: $V^k_{visited} \leftarrow add(Index_1, Index_2)$

7: ***else:***

8: $V^k_{visited} \leftarrow add(Index_1, Index_2)$

9: $S_{candidate}, T_{candidate} \leftarrow greedyUSV(graph, T_{limit}, B, V^k_{selected})$

10.: ***if*** $S_{candidate} \geq S_0$ && ***if*** $T_{candidate} \leq T_0$:

11: $S_0, T_0 \leftarrow S_{candidate}, T_{candidate}$

12: $S_0, T_0, V^k_{candidate} \leftarrow S_{candidate}, T_{candidate}, V^k_{selected}$

Result : $V^k_{candidate}$

For each USV, the same mechanism as SWAP is applied to REPLACE mechanism, for target not in $V^k_{selected} \cup V^k_{visited}$, the target is calculated by evaluating the importance value higher to the selected target and then replaced. The target result ($V^k_{candidate}$) will then become a candidate set of the targets to be evaluated further.

These procedures are inspired by the capacitated team orienteering problem Iterative local search algorithm performed by Gunawan (2018). The main differences from our algorithm and Gunawan's algorithm are that Gunawan used 2-OPT (reverse the sequence of

doi:10.6844/NCKU202000986

certain nodes) instead of SORT mechanism.

The implementation of the local search procedure within the algorithm is described as the following steps:

1. The first step in Phase 1 of our algorithm will be to applies INSERT procedure to check the time limit given for the operation (T_{limit}) and to schedule the target with the highest priority based on constrain (4.1). For each iteration, the algorithm is going to check if the target inserted to the scheduled target is feasible using a greedy algorithm.

2. For each insertion procedure, the SORT procedure is then applied to sort the $V_{candidate}^{k}$ to minimize the makespan for each USV. For each target i, $\forall(i) \in V_{candidate}^{k}$ the total importance value is then calculated to obtain the sum of the selected set of targets. After each SORT procedure, the feasibility of the routing is then conducted along with the objective function comparisons to the initial solution.

3. The candidate from the set of the selected target is then evaluated using the SWAP procedure to minimize the makespan for each USV, and then the optimal solution will then go to the REPLACE procedure to increase the total objective values of the obtained set of targets.

4. The set of targets obtained from the previous procedures ($V_{candidate}^{k}$) will then be evaluated by MOVE procedure to balance the makespan for each USV. For each target selected, the same feasibility check is then applied by using the greedy algorithm.

The pseudocode in Table 4.9 shows the procedure of the USV routing mechanism in *Reroute* function. The inputs needed in this algorithm are generated from the network with are $graph$ as the network file, T_{limit} as the parameter for the time limit of the operation, and B as the battery level capacity.

doi:10.6844/NCKU202000986

Table 4.10. *Reroute* Local Search Algorithm

***Reroute* Local Search Algorithm**

Input Data: $graph, T_{limit}, B, D_i, K$

Initialization: $iteration = 0, S_0 = 0, T_0 = 10000, V^k_{selected} = \emptyset$

1: $\boldsymbol{While}\ iteration \leq length(V_{target})$:

2: $V^k_{candidate} \leftarrow INSERT(graph, T_{limit}, B, V^k_{selected}, D_i, K)$

3: $S_{candidate}, T_{candidate} \leftarrow greedyUSV(graph, T_{limit}, B, V^k_{candidate},)$

4: $\boldsymbol{if}\ S_{candidate} \geq S_0\ \&\&\ T_0 \geq T_{candidate}$:

5: $S_0, T_0\ , V^k_{selected} \leftarrow S_{candidate}, T_{candidate}, V^k_{candidate}$

6: $V^k_{candidate} \leftarrow SORT(graph, T_{limit}, B,, V^k_{selected}, S_{candidate}, K)$

7: $S_{candidate}, T_{candidate} \leftarrow greedyUSV(graph, T_{limit}, B, V^k_{candidate})$

8: $\boldsymbol{if}\ S_{candidate} \geq S_0\ \&\&\ T_0 \geq T_{candidate}$:

9: $S_0, T_0\ , V^k_{selected} \leftarrow S_{candidate}, T_{candidate}, V^k_{candidate}$

10: $iteration = iteration + 1$

11: $V^k_{selected} \leftarrow SWAP(graph, T_{limit}, B, V^k_{selected}, S_0, T_0, D_i, K)$

12: $V^k_{selected} \leftarrow REPLACE(graph, B, T_{limit}, V^k_{selected}, D_i, V_T, K)$

13: $S_{candidate}, T_{candidate} \leftarrow greedyUSV(graph, T_{limit}, B, V^k_{selected})$

13: $V^k_{selected} \leftarrow MOVE(T_{limit}, B, graph, V^k_{selected}, S_{candidate}, T_{candidate}, D_i, K)$

14: $S_{solution}, T_{solution} \leftarrow greedyUSV(graph, T_{limit}, B, V^k_{selected})$

15: $M_{required} \leftarrow Meetingpoints(S_{solution}, T_{solution})$

16: $S_p, T_p \leftarrow parentroute\ (graph, M_{required})$

Result : S_p, T_p

The most important solution generated from this procedure is the $M_{required}$ as a set of required meeting points and the time of requirements. These required locations to meet with parent boat are the meeting points that are guaranteed for the USV routing to achieve as many targets with the highest total objective values with minimum makespan. This marks the end of the phase 1 with $parentroute$ representing the PB route function of the phase 2.

doi:10.6844/NCKU202000986

4.2.3 Greedy USV Routing Algorithm

With the target selected in each cluster ($V_{selected}^{k}$), the main procedure in phase 1 is to find the route for each USV. The main mechanism for this procedure is that according to the priority of the selected targets, the USV is required to reach that target, perform the rescue mission, and continue or return.. For each trip that the USV makes, the feasibility check of the movement will be the battery capacity and the predicted battery consumptions to make the trip. For each USV, the inputs needed in this algorithm are the $graph$ as the network file, $V_{selected}^{k}$ as the nodes selected or the scheduled nodes, K as the number of USVs, T_{limit} as the parameter for the time limit of the operation, and B as the battery level.

Table 4.11. *Greedy* USV Algorithm

***Greedy USV* Algorithm**
Input Data: $graph, T_{limit}, B, V_{selected}^{k}, K$
Initialization: t=0, $init = V_p[0], S_{solution} = \emptyset, T_{candidate} = 0,$
1: ***For*** $each\ k\ in\ K$:
2: ***For*** $each\ d\ in\ V_{selected}^{k}$:
3: $T_{candidate} \leftarrow T_{candidate}$
4: $B_0 \leftarrow B$
5: $S_{candidate}, T_{candidate}, \mathrm{B} \leftarrow shortestpath(graph, init, d, \mathrm{B}, T_{candidate})$
6: $S_2, T_2, B_2 \leftarrow shortestpath(graph, d, d+1, \mathrm{B}, T_{candidate})$
7: ***if*** $B \leq B_2$ ::
8: $S_{candidate}, T_{candidate}, \mathrm{B} \leftarrow recharge(graph, init, B_0, T_{candidate})$
9: $S_{solution}, init \leftarrow S_{candidate}, S_{candidate}[n]$
10: ***else***:
11: $init = d$
12: $S_{solution}, T_{candidate}, \mathrm{B} \leftarrow returnshortestpath(graph, init, \mathrm{B}, T_{candidate})$
13: ***if*** $T_{candidate} \leq T_{limit}$:
14: $S_{solution} \leftarrow add\ (k, S_{solution})$
15: ***else***:
16: $S_{solution} \leftarrow \emptyset$
Result : $S_{solution}, T_{candidate}$

doi:10.6844/NCKU202000986

The goal of this algorithm is to obtain the route ($S_{solution}$) and time ($T_{candidate}$) for the preselected target. For each trip the calculated distance from the current location of the USV to the target will be calculated, if the battery after the trip is not enough for the next trip the USV is required to go back to the nearest parent boat route that will be converted to required meeting points ($M_{required}$). If its enough the path will be converted to the solution ($S_{solution}$). With all target trips calculated, the USV will return to the parent boat path and the total time will be evaluated with the time limit for the operation (T_{limit}). Thus this is all the necessary information needed to process the USV route and generating the required meeting points for phase 2.

4.2.4 Iterated Local Search Algorithm Phase 2

The goal of phase 2 is to obtain a feasible parent path based on the available parent boat area and the assigned meeting points from phase 1. As stated previously, the USV is expected to wait for the parent boat to arrive at the meeting point if both cannot meet on time. Based on these sets of constraints, we can obtain a new optimization problem, as shown in 4.1.

$$Minimize \qquad \sum_{(i,j)\in A_p} \sum_{t\in T} y_{ijt} \tag{4.1}$$

The goal of the objective function (4.1) is to minimize the parent path results with the following constraints:

$$\sum_{(i,j)\in A_p \cup A_{sink} \cup A_s} y_{ijt} - \sum_{(h,i)\in A_p \cup A_{source} \cup A_s} y_{hi(t-\Delta_{ij})} = 0 \qquad \forall i \in V_p, t \in T \tag{4.2}$$

The constraints shown in Equation (4.2) define the flow balance for the parent boat in the parent boat path for $i \in V_p$, where $t - \Delta_{ij}$ represents the previous movement in the network.

$$\sum_{t\in T} y_{iit} >= 1 \qquad \forall i \in V_{required} \tag{4.3}$$

$$y_{ijt} \in \{0,1\}, \qquad (i,j) \in A_p, t \in T \tag{4.4}$$

The constraints shown in Equation (4.3) are used to keep the parent boat inside the required arc from phase 1, while Equation (4.4) is the decision variable for parent boat movement.

doi:10.6844/NCKU202000986

The meeting points configuration in the PB area can be problematic since the order for the sequence of the meeting points and the limited resources of the parent boat may cause the USV to be idle, or vice versa. On the other hand, it is also possible that the USV may require the same meeting points or edges at the same time. The framework of this optimization problem is represented in Figure 4.15.

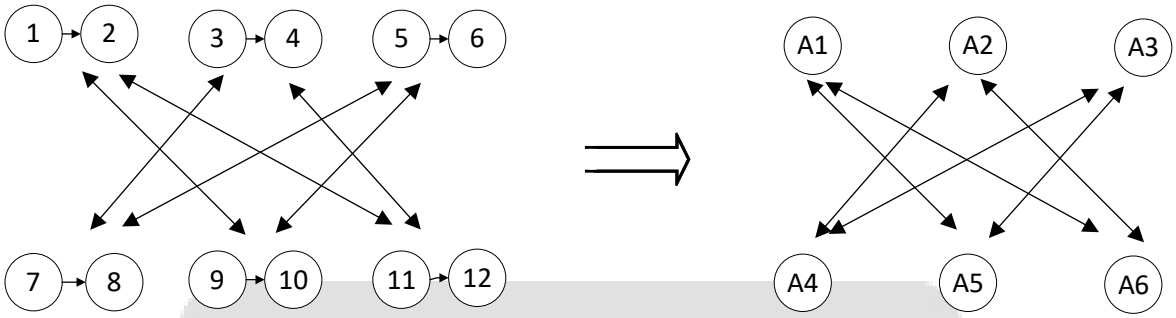


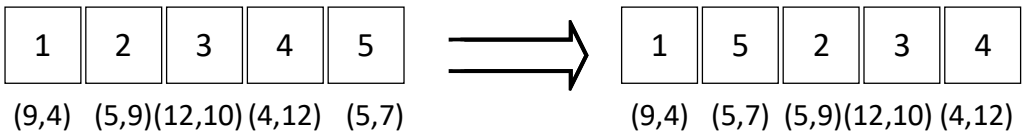
Figure 4.15. The framework of Parent boat Meeting Edges

Figure 4.15 describes the framework for this optimization problem, where it is assumed that each meeting point is meeting edges that require the parent boat to move along with the USV, so there are many possibilities of combinations. Assuming that meeting arc 1-2 is Arc 1 (A1), and all the arcs are treated the same, a feasible result will be for the parent boat to go to A1 and then to A4, A2-A5-A3-A6. Each time the parent boat transfers to another meeting edge, the USV will have to wait for the parent boat. Therefore, this will increase the downtime for each USV and cause the makespan to be higher. Finding an ideal feasible solution for these combinations can also lower the makespan and thus affect the total objective values for each USV. For the local search algorithm within the feasible path of the parent boat, there are a set of abbreviations that are described in the procedures below:

1. SORTTIME Procedure: Rearranging the sequence of the scheduled target visit according to the time. The SORTTIME mechanism is conducted by evaluating the nearest time between each meeting point. The parent boat will go to the nearest meeting point at the next time required within the parent boat area. According to this algorithm, time will be the priority in finding the feasible path.

doi:10.6844/NCKU202000986

SORTTIME Procedure



(N,T) Node position, Time required

Figure 4.16. SORTTIME Procedure Sequence

As described in Figure 4.16, the SORTTIME procedure will reorder the meeting sequence based on the nearest required time between each meeting point. Based on this example and assuming we have (N, T) as the information for each meeting point (1,2,..,5) where each meeting point has its own required location and required time to meet with the PB, the order for the meeting points based on the time will be 1-5-2-3-4 with the required time sequence of 4-7-9-10-12, for each meeting point respectively. This information does not guarantee that the parent boat will be able to meet with the USV on time. If the travel time from meeting point 1 to point 5 requires 5 sets of time, then, assuming the parent boat arrived at point 1 on time, it will arrive at point 5 at time 9, and thus, the USV will have to wait from time 7 to time 9 and postpone all the meeting points related to this USV by 2 sets of time. Therefore, this procedure aims to reorder the meeting points based on the time to minimize the downtime for each USV.

1. SORTSPACE Procedure: Rearranging the sequence of the scheduled target visit according to time. The SORTSPACE mechanism is conducted by evaluating the nearest space between each meeting point. The parent boat will go to the nearest space within the parent boat area. According to this algorithm, the distance between each meeting point will be the priority in terms of finding the most feasible path.

doi:10.6844/NCKU202000986

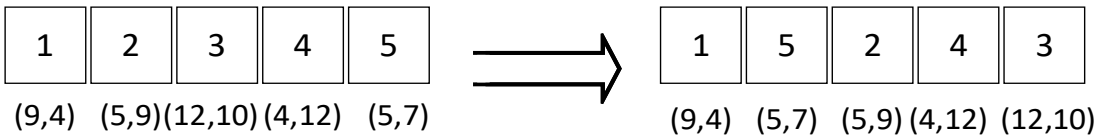


Figure 4.17. SORTSPACE Procedure Sequence

As described in Figure 4.17, the SORTSPACE procedure reorders the meeting sequence based on the nearest distance between each meeting point. Based on this example and assuming we have (N, T) as the information from each meeting point (1,2,..,5) with each meeting point having its own location and time set to meet with the PB, the order for the meeting points based on the distance will be 1-5-2-4-3. Compared to the SORTTIME mechanism, the main differences between the result generated is in meeting point 4 and meeting point 3. In SORTTIME mechanism, the PB is required to go to meeting point 2 (t = 9) and then meeting point 3 (t=10) since the nearest required meeting point based on time is meeting point 3 instead of 4 (t =12). Meanwhile, in the SORTSPACE mechanism, the PB is required to go to meeting point 4 (node 4) after meeting point 2 (node 5) since the location for the meeting is closer compared to meeting point 3 (node 12). This information does not guarantee the parent boat will be able to meet the USV on time. However, based on this reordering sequence, we can attempt to lower the total makespan of the parent boat, assuming that the result is feasible.

The implementation of the local search procedure within the algorithm is described as the following steps:

1. The first step in phase 2 of our algorithm will be to apply the SORTTIME procedure to rearrange the sequence meeting point i, $\forall(i,t) \in M_{required}$ in the parent boat area, followed by applying the greedy parent boat algorithm to calculate the total time needed for the routing of this operation ($S_{candidate}$) and adding this possible solution to the possible selected sequence.
2. The next step in our algorithm is similar to step 1, where the SORTSPACE procedure

doi:10.6844/NCKU202000986

is applied to rearrange the sequence meeting point i ,$\forall(i,t) \in M_{required}$ in the parent boat area, followed by applying a greedy parent boat algorithm to calculate the total time needed for the routing of this operation ($S_{candidate}$) and adding this possible solution to the possible selected sequence.

3. From the two sets of possible sequences, the solution the data obtained, the data from the solution is synchronized with the required time for the required meeting points from phase 1 to allow the possibility of the parent boat waiting in case the parent boat arrives faster than the parent boat
4. Both $T_{candidate}$ (operation time candidate) values from the two operators are then compared. The path with the least total operating time is then selected, as well as the route for the solution ($S_{solution}$) routing as the solution for phase 2.

The pseudocode in Table 4.11, shows the procedure of the USV routing mechanism in the *Parentroute* function. The inputs needed in this algorithm are generated from the network with $graph$ as the network file, and the original meeting point i $(\forall(i,t) \in M_{required})$.

Table 4.12. *Parentroute* Local Search Algorithm

***Parentroute* Local Search Algorithm**
Input Data: $graph, M_{required}$
Initialization: ,$S_{solution}$=∅
1: $S_0, T_0 \leftarrow parentgreedy(graph, M_{required})$
2: $M_{candidate} \leftarrow SORTTIME(M_{required})$
3: $S_{candidate}, T_{candidate} \leftarrow parentgreedy(graph, M_{candidate})$
4: ***if*** $S_{candidate} \geq S_0$ && ***if*** $T_{candidate} \leq T_0$:
5: $S_0, T_0 \leftarrow S_{candidate}, T_{candidate}$
6: $M_{candidate} \leftarrow SORTSPACE(M_{required})$
7: $S_{candidate2}, T_{candidate2} \leftarrow parentgreedy(graph, M_{candidate})$
8: ***if*** $S_{candidate} \geq S_0$ && ***if*** $T_{candidate} \leq T_0$:
9: $S_0, T_0 \leftarrow S_{candidate}, T_{candidate}$
10: $S_{solution}, T_{solution} \leftarrow synchronize\ (S_0, T_0, M_{required})$
Result : $S_{solution}, T_{solution}$

doi:10.6844/NCKU202000986

With the solution obtained from this local search algorithm ($S_{solution}, T_{solution}$), we now have found a feasible parent path route as well as its corresponding USV route and the total objective values. The next step of the algorithm is to move the procedure to phase 3 to either improve the total objective value or lower the total operation time.

4.2.5 Greedy Parent Boat Routing Algorithm

With the required meeting points obtained from the previous procedure ($M_{required}$), the main procedure in this step is to find the route for the parent boat. The main mechanism for this procedure is that according to the priority of the selected points, the parent boat is required to reach a given target, perform the recharging mission, and continue to the next point. The inputs needed in this algorithm are the $graph$ as the network file, $M_{required}$ as the nodes selected or the scheduled nodes with some initialized variables of V_{origin} as the source node, and V_{sink} as the sink node.

Table 4.13. *Greedy Parent Boat* Algorithm

ParentGreedy Function
Input Data: $M_{required}$, $graph$
Initialization: t=0, V_{origin}= $V_p[0]$,$V_{sink} = V_p[n]$, $dest$=$M_{required}$ [0][0], i =1
1: $S_{solution}, T_{candidate} \leftarrow shortestpath(graph, V_{origin}, dest, t)$
2: $Initial \leftarrow dest$
3: $V_{required} \leftarrow remove(initial)$
4: $\boldsymbol{For}$ $each\ dest\ in\ M_{required}$[i][0] :
5: $T_{candidate} \leftarrow T_{candidate}$
6: $S_{candidate}, T_{candidate} \leftarrow shortestpath(graph, initial, dest, T_{candidate})$
7: $S_{solution}$, $initial \leftarrow S_{candidate}, dest$
8: i+1
9: $S_{solution}, T_{solution} \leftarrow shortestpath(graph, initial, V_{sink}, T_{candidate})$
Result : $S_{solution}, T_{solution}$

The goal of this algorithm is to obtain the route ($S_{solution}$) and time ($T_{solution}$) for the preselected points. For each meeting point i,t ($\forall(i,t) \in M_{required}$), the calculated distance

doi:10.6844/NCKU202000986

will be calculated and added to the routing solution ($S_{solution}$). After all the points have been visited, the parent boat is required to return to the sink node (V_{sink}). Thus, this is all the necessary information needed to process the parent boat route.

4.2.6 Iterated Local Search Algorithm Phase 3

The aim of this phase is to find the optimal PB path and reroute the USV to find a better solution. This is accomplished by swapping the available areas and then rerouting both the USV and the parent boat to achieve a better total objective value or lower the total operating time needed. With the optimal parent boat route from phase 2, we can attempt to adjust the nearest parent boat meeting point to not be available for the parent boat. Therefore, the USV will have to find the second nearest point to its location. Since this new meeting point will have a different trip distance and battery consumption, this decision will affect the USV performance in terms of reaching as many targets as before for better or worse. On the other hand, now that there are new meeting points, it is also possible that all the meeting points after this meeting point will be affected in time and space. Therefore, the routing for the parent boat will also have to be reconstructed. Since in case C1, the PB path is already obtained, this mechanism is not needed and therefore is only needed in case C2.

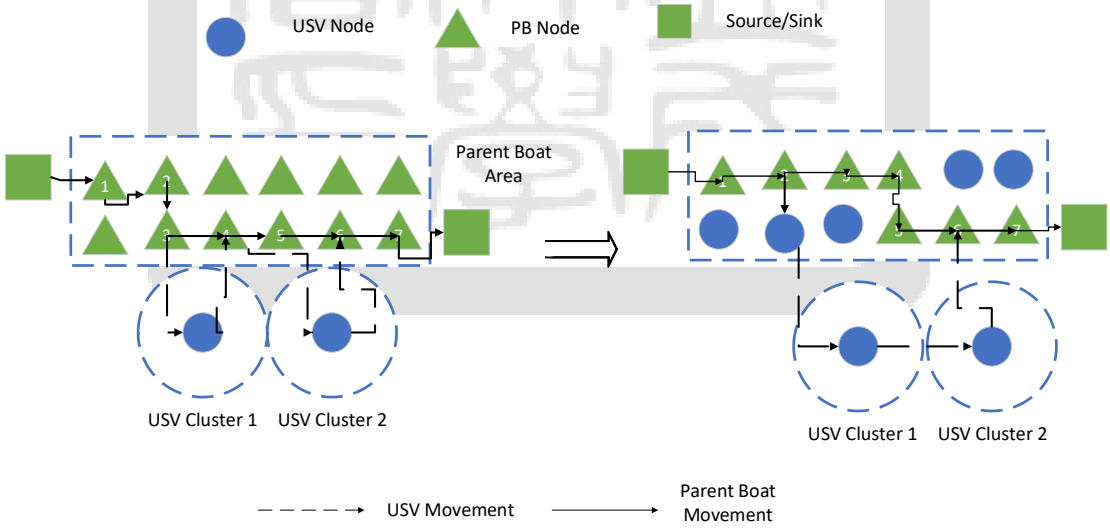


Figure 4.18. Phase 3 Procedure: before removal (left) after removal (right)

The example in Figure 4.18 shows how much the removal of nodes 2 and 3 on the left picture can affect the routing in the new graph (right). Now that nodes 2 and 3 are changed to be further from cluster 1 and cluster 2, the USV proceeds to go to node 5 (right)

doi:10.6844/NCKU202000986

to recharge instead of node 3 (left). This change will make the USV go faster to USV cluster 2 and may or may not increase the total objective values in USV routing. From the parent boat perspective, the number of available nodes in the parent boat area is now reduced. Therefore, the possible nodes for the meeting become a straight path (right) instead of altering its direction (left).

The steps to do this operation is described in the following procedure:

1. From the initial solution before the alteration, the routing from phase 2 will be set as the set of nodes to be altered ($V_{removed}$).
2. For each node i ($\forall i \in (V_{removed})$, the available parent boat area is altered, and thus, node i will not be available for the parent boat, and the USV has to find the second nearest parent boat area as the meeting point.
3. For each node that is removed, the functions for phase 1 and phase 2 are going to be performed ($Reroute(graph, T_{limit}, B, D_i, K)$) to recalculate the total operating time ($T_{candidate}$) and the total objective values ($S_{candidate}$) and reconstruct the graph.
4. For each node successfully removed with an improved solution, the algorithm will list those nodes and reconstruct the graph for the final solution.
5. For the final solution, phase 1 and phase 2 are conducted again to reconfirm the solution. If none of the nodes can provide an improved solution, the solution is set to the initial feasible solution.

4.3. Sequential Segmentation Heuristic

Another method that we developed to shorten the computational time of the IP model is called Sequential Segmentation Heuristic (SSH). This method is aimed to divide the searching space by dividing the model by segments that the user defines. The reasons behind this process are due to the nature of the branch-and-bound searching process and the time-space path nature of our model. By dividing the model into multiple segments, the model will be able to search for a limited solution, which will shorten the computational time needed to solve each segment. This heuristic can be applied to both cases C1 and C2 as the PB path doesn't affect the procedure in this heuristic. However, this method has its own weaknesses that the number of segmentation is highly dependent on the networks and the problem itself. The illustration for this method is shown in Figure 4.19.

doi:10.6844/NCKU202000986

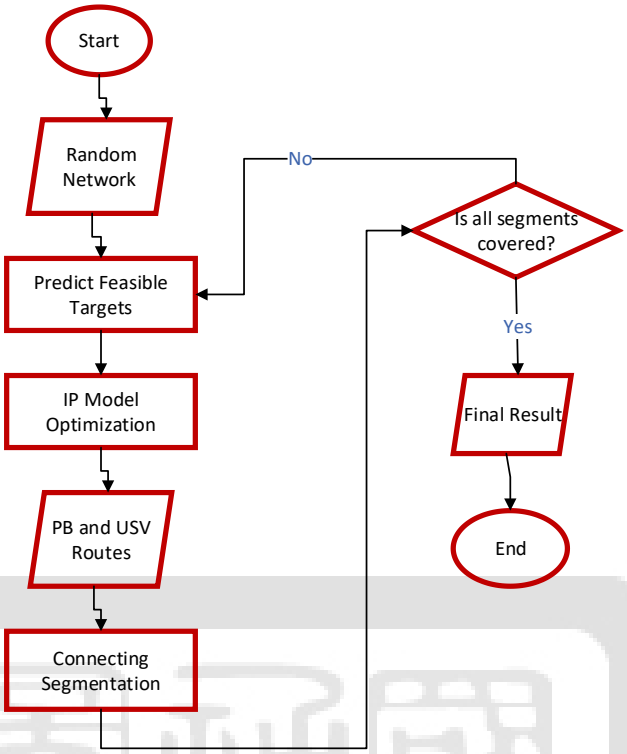


Figure 4.19. Segmentations Iteration Framework

This method is more straightforward compared to the Iterative Clustering Heuristic. The input that the model needed from the user is the time for each iteration and the network file. For example, in Figure 4.19, assuming we have defined the time for each iteration to be 20 sets of times with 60 as the total operation time. This method will implement 2 segmentations in each iteration with 20 total operation time, and each iteration is processed using the IP model. The network file itself is kept to the original network in each iteration. There are two major steps in this heuristic. The first step is at the start of each iteration the targets will be predicted to limit the solution workspace that will be explained in subsection 4.3.1., while the second step is subsection 4.3.2., that will explain the connection between mechanism to generate the final results.

4.3.1. Predicting Feasible Targets

Since we implement the IP model in each iteration, the more targets considered in the workspace can affect the computational time greatly since there will be more possibilities of movement for the USVs. To limit the searching process of IP model solver, the shortest path algorithm is implemented to calculate the targets that can be reached by the USVs within the

doi:10.6844/NCKU202000986

time-bound. Another constraint is that the battery level capacity should be higher or equal to the energy needed to return to the nearest parent boat area. For the targets outside of the feasible area. The targets will not be considered in the iteration and thus limit the possible movements for the USVs.

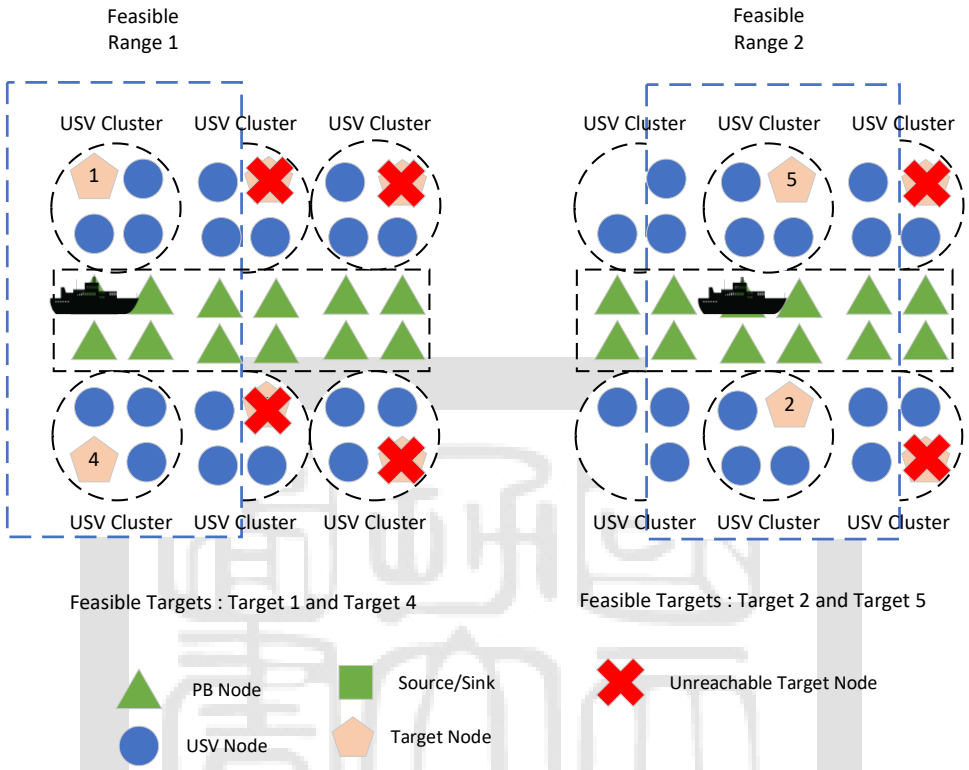


Figure 4.20. Predicting Targets Illustration

In Figure 4.20, assuming that we have obtained the feasible USV region within the time-bound in an iteration, we can use the shortest path algorithm to reach a target, when a target is reached within the time-bound, we include the target in the model. However, if based on the time bound a target is can not be reached then moved the target to the next iteration. By this mechanism each iteration will not have to calculate as many targets in each iteration.

4.3.2. Segmentation Connections

For the position of the USVs and the PB, we used the position at the end of the first segment of each iteration. As shown in Figure 4.21 started iteration 2 with segment 2 and segment 3 using the USVs and PB position at the end of segmentation 1. While in iteration 2, the segments calculated are segment 2 and segment 3.

doi:10.6844/NCKU202000986

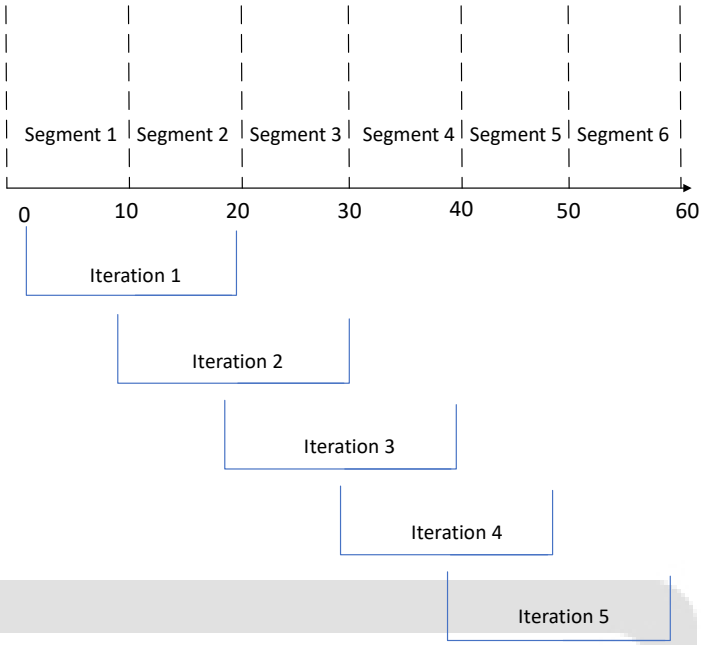


Figure 4.21. Segmentation Process

This mechanism is needed because if we start the next iteration from segment 3, it is possible that since the IP model didn't consider segment 3 in iteration 1 then the battery at the end of the operation is the most suitable for the segments considered in iteration 1. Therefore at the start of each iteration, we start the segmentation from the last segmentation in each iteration until all iterations are covered.

doi:10.6844/NCKU202000986

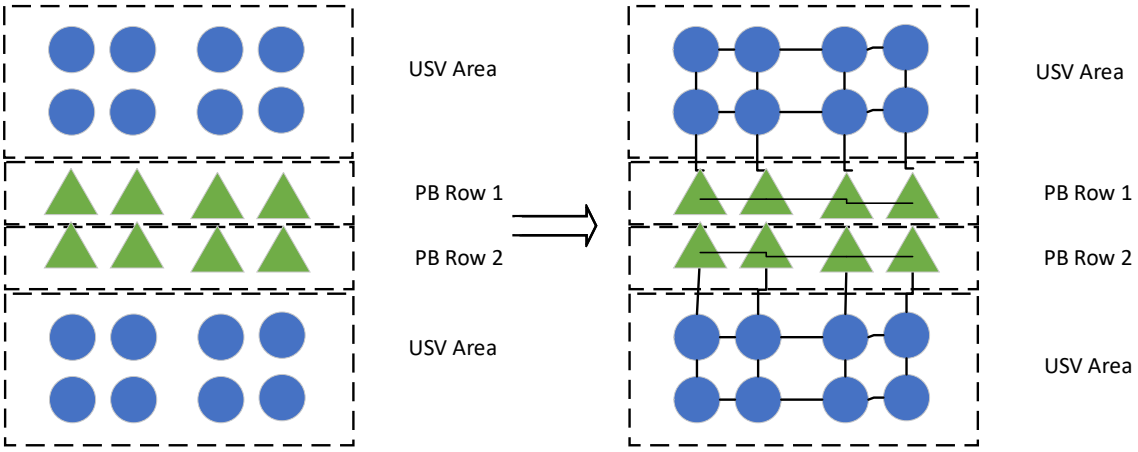
Chapter 5

COMPUTATIONAL EXPERIMENTS AND ANALYSIS

This chapter explains the random distribution networks for the computational test as well as discussing the computational comparison of the algorithms. Here, we implement a random network generator that produces the clusters for each node, the coordinate for each node, and the grid network. The network structure generator is described in Section 5.1, followed by the technical specifications for computational testing in Section 5.2, the settings for the C1 and C2 experimentations in Section 5.3, the experimental results in Section 5.4, an illustrative example with a real-world scenario setting in Section 5.5, and a summary of the results in Section 5.6.

5.1. Join Operation Random Network Generator

There are several parameters such as $S_u^k$with $k \in K$ (USV nodes area for each USV), S_p(size for the parent boat area), and T (required targets), $n_{clusters}$ (required clusters), that a user can specify to generate suitable networks. Depending on the size of the parent boat, these settings cannot be arbitrary. The example in Figure 5.1 shows that a simple network is formed according to the diagonal edges between each node. Since in this research, the network is evaluated through a grid-based network, the network formed to follow the grid rules that each node is connected to its adjacent neighbor nodes.



doi:10.6844/NCKU202000986

Figure.5.1. Grid-based network system

Each relationship between nodes consists of two edges, where assuming we have node a and node b, then the edges within these two nodes are A_{ab}, and A_{ba}. There are different types of edges used in this problem to model the connecting edges that cannot be put into the same category as the USV area edges since this category is related to the possibility of recharging USVs in the parent boat nodes. Therefore, the three types of edges are USV edges, connecting edges, and parent boat edges. According to the input data needed for the model, there are four main parts needed for the construction of the network: USV edges, parent boat edges, USV-Parent connection edges, and the target generator.

5.1.1. USV Edges Construction

Some procedures have to be followed to construct USV edges ($A_u = \{(i'', j'')|i'', j'' \in V_u$). The input needed for these procedures are the nodes for USV region (V_u), the parent boat nodes (V_p), and the number of rows within the parent boat nodes (row_p). The steps are described as follows:

1. Constructing Linear edges

   With the input that the user gives starting from node 0 (source), our generator generates a linear relationship until it reaches all USV area nodes. These linear edges are the incoming arc for the destination nodes. Suppose we have n set of nodes. The linear arc will be node n-5 into node n-4, node n-4 into node n-3, and so on, up to node n-1 to node n. Due to the nature of the grid network, the weight in each arc should be the same.

2. Constructing Horizontal edges

   For the horizontal edges, first depending on the nodes in row_p, the first procedure is to calculate the equivalent column nodes for each USV node to construct the horizontal edges. As described in Figure 5.2., there are patterns in each horizontal node. Suppose from n nodes in row_p, the next horizontal node is n+1, followed by node n-1 in a horizontal relationship to node n+1+2. For every -1 movement in the USV nodes, the horizontal nodes are the respective +2 counter plus the previous increment value.

doi:10.6844/NCKU202000986

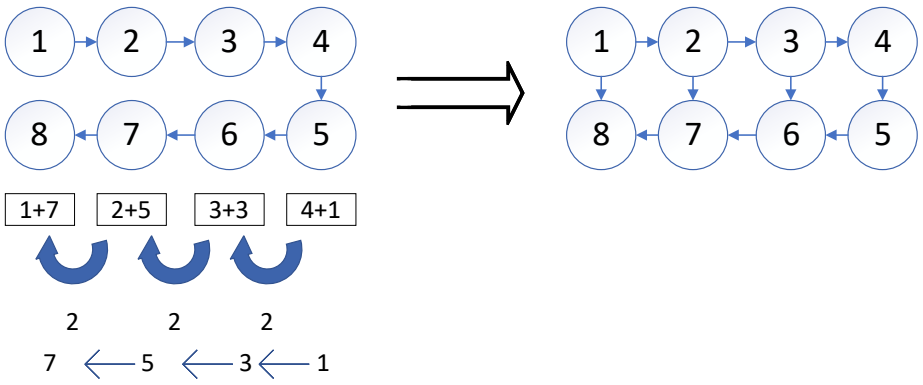


Figure 5.2. Construction of the Horizontal Edges

Based on these patterns and rules, the horizontal edges are constructed and assigned the same weight as the linear arc. Since each USV may go to the same node multiple times during the operation, for each arc (linear and horizontal), the reverse arcs are also assigned to each arc. With the parent boat area to be given in the middle of the network, the USV areas are constructed twice on each side of the parent boat area.

5.1.2. Construction of Parent Boat Edges

The input needed for the procedures used to construct the parent boat edges $A_p = \{(i', j') | i', j' \in V_p)$ are the nodes for USV region (V_u), the parent boat nodes (V_p), and the number of rows within the parent boat nodes (row_p). The steps are described as follows:

1. Constructing Linear edges

   The same rules apply for the parent boat linear edges. Our generator generates linear relationships until it reaches the all parent boat region nodes. These linear edges are the incoming arc for the destination nodes. Because the parent boat area is in the middle of the network, if we have n set of nodes for the parent boat, the linear arc will start from the maximum node in V_u +1. Thus, assuming the USV region is assigned 30 nodes, the parent boat's lowest value will start from 31 and go to the highest value of the parent boat nodes. Due to the nature of the grid network and time-space model for the network, the weight in each arc should be higher than the USV region to allow the movement in parent boat nodes to consume more value than the movement in the USV region.

2. Constructing Horizontal edges

   For the horizontal edges, first, depending on the nodes in row_p, the first

doi:10.6844/NCKU202000986

procedure is to calculate the equivalent column nodes for each parent boat node to construct the horizontal edges. As described in Figure 5.2., there are patterns in each horizontal node. Following the same mechanism, the horizontal arcs within the parent boat nodes will follow the same rules. Suppose from n nodes in row_p, the next horizontal nodes are n+1, followed by the horizontal relationship of node n-1 to node n+1+2. For every -1 movement in the USV nodes, the horizontal nodes are its respective +2 counter plus the value of the previous increment. The horizontal edges are constructed according to the rules based on these patterns and assigned the same weight as the linear arc. Since the parent boat may go to the same node multiple times during the operation, for each arc (linear and horizontal) the reverse arcs are also assigned to each arc.

5.1.3. USV-Parent Connection Edges Construction

For the procedures that must be followed to construct parent boat edges $A_c = \{(i', j'')|j'' \in R_{j'}\} \cup \{(i'', j')|j' \in R_{j''}\}$, the input for these procedures include the nodes for the USV region (V_u), the parent boat nodes (V_p), and the number of rows within the parent boat nodes (row_p). Since the connection for the connecting edges is all horizontal, the row_p (number of nodes in one row) plays an important role in this procedure.

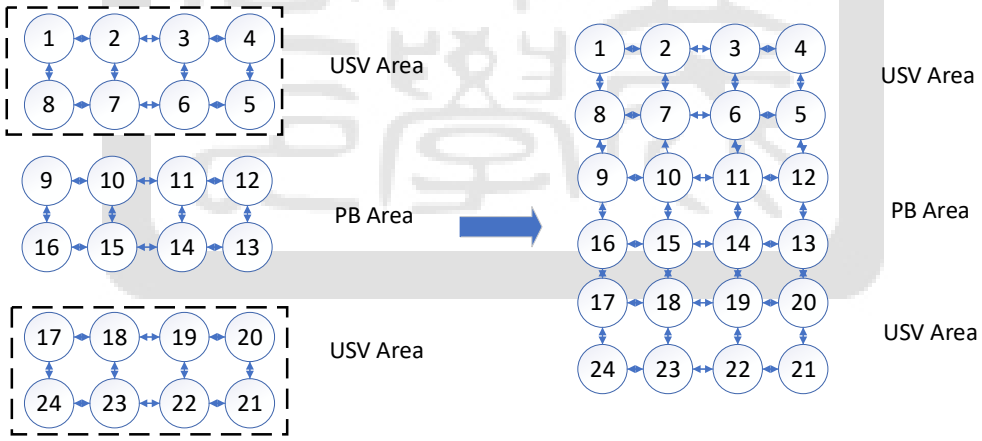


Figure 5.3. Construction of the USV-Parent Boat Connection Edges

The highest node value in USV area 1 is reduced by row_p to obtain n while the same value is then added to the row_p variable to obtain the row that will be connected to the USV area. As described in Figure 5.3., the same procedures are then applied to these rows to obtain the connections between nodes 5-8 and nodes 9-12. For the second USV region, the

doi:10.6844/NCKU202000986

lowest node value is added with row_p, while the way to obtain the row from the parent boat area is by reducing this value to row_p. From the example, this procedure is applied from nodes 17-20 and nodes 13-16.

5.1.4. Random Target generator

For the random target generator, the input needed for these procedures are the nodes for USV area (V_u), row_p as the number of rows in the PB area, V_p as the nodes for the PB area and the required targets ((i, $\forall(i) \in V_T$). For each iteration the target is randomized within the USV region with the value assigned is randomized between 1 as the lowest value and 10 as the highest importance value (D_i). The pseudocode of our random network generator is presented as follows.

Table 5.1. Random Network Generator Procedure

Random Network Construction
Input Data: V_T, V_u, row_p, V_p
Initialization: $parentnodes = V_p$
1: ***For*** *k in K:*
2: $S_u^k \leftarrow$ writeusvpath(V_u, V_p, row_p)
3: $V_p \leftarrow (V_p * row_p) + V_u$
4: $V \leftarrow V_u * 2 + (parentnodes * row_p)$
5.:$S_p \leftarrow$ writeparentpath(V_u, V_p, row_p)
6: $V_T \leftarrow$ writerandomtargets(V_u, V_p, row_p)
7: $Network.txt \leftarrow$ writeedges$(V_u, V_p, row_p, V_T, V)$
8: $Network.txt \leftarrow$ writeconnectionpath(S_u^k, S_p, row_p)
Result : $Network.txt$

The order for the pseudocode is to get the input data from the user for the total nodes of USV (V_u), and the total nodes for the parent boat area (V_p).The first procedure is to construct the USV area and the parent boat, with the data for the total nodes and the row for the parent boat area in a grid-network, then the connection edges can be constructed, the last step is to distribute the target nodes randomly (V_T). Thus, we now have all the information needed for the optimization problem.

doi:10.6844/NCKU202000986

5.2. Technical Specifications

With the proposed problem in Chapter 3, the model is constructed in optimization solver Gurobi. The computational experiments are conducted on a personal computer (PC). The specification for the PC and the software that we use to perform the computational experiments are shown in Table 5.2.

Table 5.2. PC Specifications

PC specifications	
Operating System	Microsoft Windows 10 1 x64-based
Processor	Intel ® Core ™ i7-6700HQ CPU @ 2.60 GHz
RAM	16.00 GB
Software specifications	
Python (Cython Compiler)	Version 3.7.6
Anaconda	Version Anaconda 2020.02 (64 Bit)
Gurobi	Version 9.0.3

5.3. Settings for C1 and C2 Experimental Cases

For the experimental cases, this section is mainly focused on the mathematical model computational testing and the algorithm performances. The scenarios are divided into four sections. In the first scenario (S1), the USV nodes were increased from 60, to 100, and to 200 problem sets for the computational experiments. In the second scenario (S2), the number of targets was increased from 10, to 30, and to 50 problem sets for the computational experiments. In the third scenario (S3), the total operating time was increased from 30, to 50, and to 65 problem sets for the computational experiments. The fourth scenario is conducted only for the case of C2 to see the effect of having more PB area nodes to the performance of the methods from 20, 40, and 60. There were several parameters used for the table representation, where $|V_u|$ represents the total USV nodes and $|V_T|$ represents the total target nodes; $|A|$ is the total edges. *T* represents the maximum period for the model. $|V_p|$ represents the total number of PB nodes in which the PB path will be set as a linear path from the origin node to the destination node, with 10 nodes in each row. *CPU* represents the

doi:10.6844/NCKU202000986

total computational time necessary to obtain the solution, which becomes 3600(s) if an optimal solution is not calculated within a 1-hour time limit. *Gap* represents the difference between the best-known solution and the best-bound value. We set the battery capacity at 100 (*B*= 100) for all cases, and the ICH clusters were set at 4. Each case was run 10 times to provide better representations.

Table 5.3.Comparison Model Case Information

Scenario	Problem Set	$\|V_u\|$	$\|V_p\|$	$\|V_T\|$	$\|A\|$	T
USV Nodes	S1-30	**60**	30	15	250	50
	S1-50	**100**	30	15	402	50
	S1-100	**200**	30	15	744	50
Target Nodes	S2-10	80	30	**10**	402	50
	S2-30	80	30	**30**	402	50
	S2-50	80	30	**50**	402	50
Operation Time	S3-30	80	30	20	402	**30**
	S3-50	80	30	20	402	**50**
	S3-65	80	30	20	402	**65**
PB Nodes Increment	S4-20	60	**20**	30	344	50
	S4-40	60	**40**	30	388	50
	S4-60	60	**60**	30	444	50

Aside from the setting shown in Table 5.3, for each method that we are using, the cluster in Iterative Clustering is kept to be below 35 nodes in each cluster to provide the same accuracy. While the segmentation in Sequential Segmentation Heuristic (SSH) is set to be 20 sets of time in each segment.

5.4. Experiments Results

This section is divided into four subsections detailing the evaluation made on the GAP (%) and the computational time for each case in each scenario.

5.4.1. Scenario USV Nodes for Cases C1 and C2

To evaluate the performance of the C1 and C2 mathematical models mentioned in Chapter 3, each case was run on each model with the same value of T and B. Each experiment was considered alone in an individual run for the first step. All of the cases were solved in a 1-hour time limit irrespective of whether a feasible or optimal solution was found. This time

doi:10.6844/NCKU202000986

limit was used either because increasing the 1-hour time limit would not lead to any significant improvement or because it would take a long time to obtain any significant improvement. All indicator values in the table were obtained using the Gurobi Solver.

Table 5.4.Scenario USV Nodes Computational Result for the Case of C1

C1	Problem Set			IP Model		Iterative Cluster Heuristic		ILS Algorithm		Sequential Segmentation Heuristic		Best Upper Bound Value
		|A|	$\|V_u\|$	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU (s)	GAP(%)	CPU(s)	GAP(%)	
	S1-30	250	60	3600.52	52.94	532.97	0.90	1.41	0.00	720.58	6.6	74.25
	S1-50	402	100	3600.23	-	1988.43	4.80	1.91	14.19	1132.17	6.23	78.63
	S1-100	744	200	3600.10	-	2855.12	10.64	3.47	22.05	1311.44	25.02	80

For the case comparisons of the three methods, the fastest to solve the problem was by using the ILS algorithm although the result was not an optimal solution for S1-50 and S1-100. Since in the case of C1, it is not necessary to find the path for the PB, the ILS algorithm and Iterative Cluster Heuristic (ICH) can be performed faster. We can conclude that due to the clustering technique, the workspace for the USV in the ICH is simplified and thus managed to lower the computational time significantly since in the case of S1-50 and S1-100, the IP model was not able to achieve a feasible solution within the 1-hour time limit. Adding more USV nodes increases the workspace for each USV, the positions for each target, and the battery consumption flow balance. Since there are multiple USVs in our methods, each USV is affected significantly by these constraints, which was therefore the main reason the IP model could not find a feasible solution in S1-50 and S1-100.

Table 5.5. Computational Results for the Scenario USV Nodes for Case C2

C2	Problem Set			IP Model		Iterative Cluster Heuristic		ILS Algorithm		Sequential Segmentation Heuristic		Best Upper Bound Value
		|A|	$\|V_u\|$	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU (s)	GAP(%)	CPU(s)	GAP(%)	
	S1-30	250	60	3600.52	4.41	736.84	3,34	67.25	0.43	146.17	2.8	69.75
	S1-50	402	100	3600.00	-	2851.76	3,62	75.14	17.51	424.32	3.58	80
	S1-100	744	200	3600.09	-	3050.28	15.34	139.9	30.02	792.01	14.60	81.88

In case C2, compared to the same size network in C1, the SSH performed better compared to the ICH based on the GAP (%) and computational time. However, the ILS algorithm and IHC performance was worse than that of C1 based on both the Gap (%) and the computational time. This was to be expected because of the addition of phase 3 in the

doi:10.6844/NCKU202000986

ILS algorithm and phase 1 in the ICH to calculate the PB path. The same behavior could be seen in the IP model, where, due to the size of the network, in a smaller network (S1-30), the IP model was able to find a feasible solution. Aside from the USV nodes increments, since the PB path was not yet fixed, the flow balance constraint for the PB was affected as well as the meeting time for both the PB and USVs.

Another interesting finding was that by dividing the problem workspaces by clustering or time segmentation, it was possible to find an optimal solution using the solver. Due to the branch-and-bound nature of the solver, increasing the workspace size did not affect the SSH greatly compared to the ICH since the operating time was the same in the SSH. On the other hand, since the time was not divided in the ICH, the computational time remained higher due to having more time-space workspace.

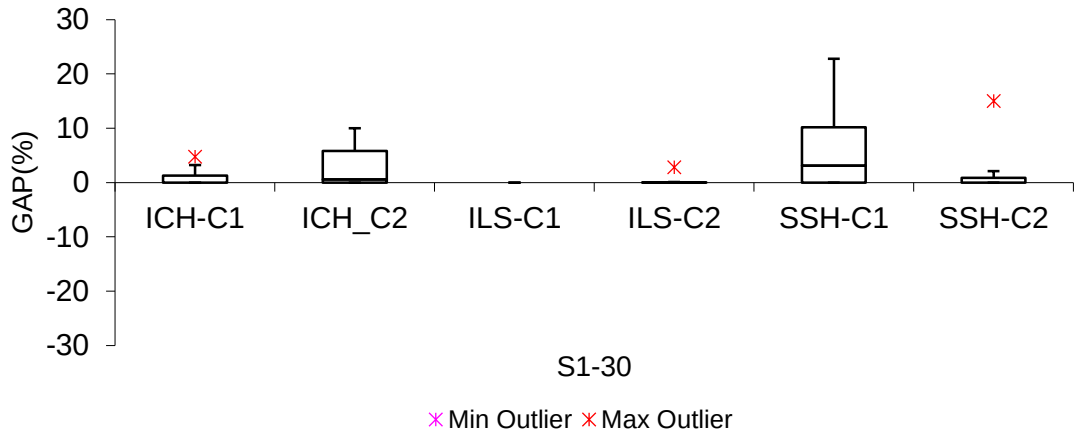


Figure 5.4. GAP (%) comparisons in S1-30 (15 Targets and 50 Sets of Time)

To better understand each algorithm, the GAP (%) in S1-30 is compared to the same figure for all methods. Figure 5.4 shows that even when the optimality may not be reached in some methods, the result for both C1 and C2 is closer to 0. This means that in the network size and the settings provided, these methods are able to perform almost as well as the IP model and are also faster.

doi:10.6844/NCKU202000986

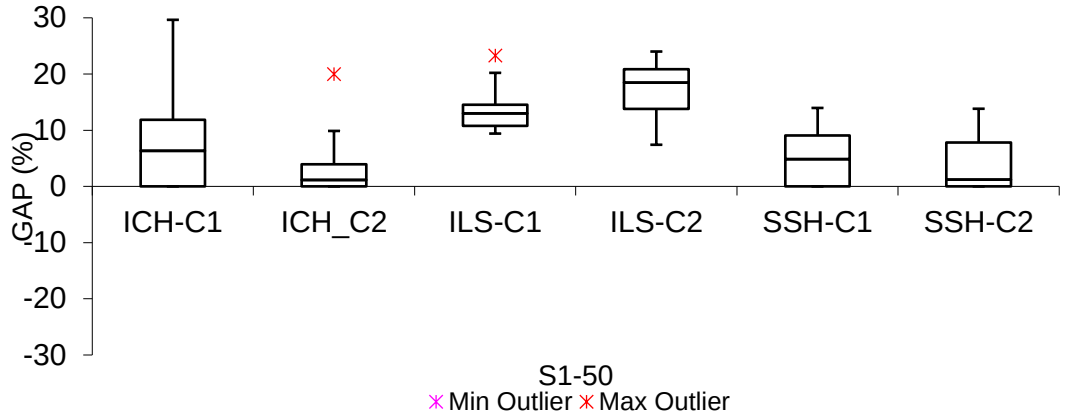


Figure 5.5. GAP (%) comparisons in S1-50 (15 Targets and 50 Sets of Time)

In the S1-50 problem set, the C2 GAP (%) in the ICH and SSH is less than the corresponding network setting in C1. Based on these results, we can conclude that based on the S1-50, SSH, and ICH settings, it is possible to reach a better result faster than is the case for C1. A possible explanation for this is that due to not yet having a PB path, there were more possibilities by which to reach a solution in the case of C2, so it was possible to arrive at a better solution within the given time.

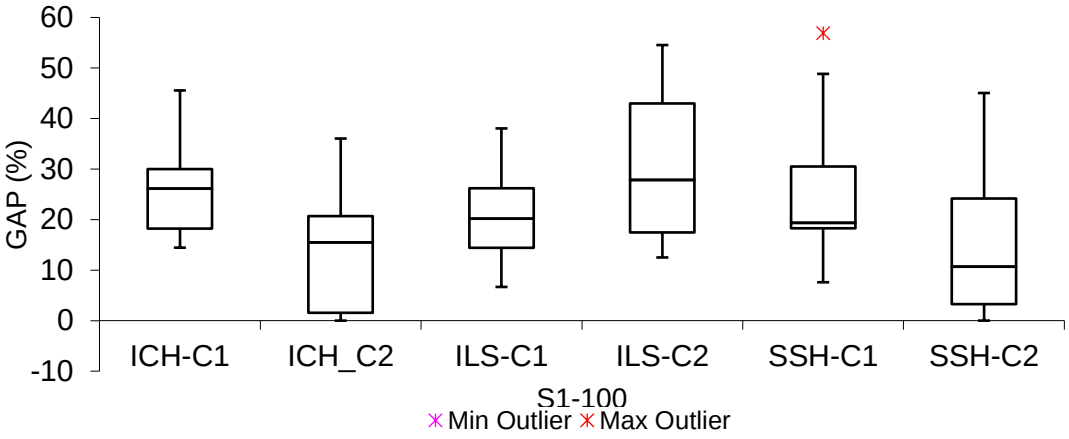


Figure 5.6. GAP (%) comparisons in S1-100 (15 Targets and 50 Sets of Time)

In the problem set S1-100, the GAP (%) for C2 in the ICH and SSH was also less than the corresponding network setting in the case of C1. However, for the ILS algorithm, according to the previous results, in C2, the ILS performed worse than the corresponding network in C1. This means that when the PB path is not yet defined, the algorithm has to search more and that a better PB path local search can be a good direction for this algorithm.

doi:10.6844/NCKU202000986

5.4.2. Scenario Target Nodes for Cases C1 and C2

The same mechanism was applied to scenario 2. All the cases were solved in a 1-hour time limit irrespective of whether either a feasible or optimal solution was found. The results of the experiment are shown in Table 5.6.-5.7. All indicator values in the table were obtained using the Gurobi Solver.

Table 5.6. Scenario Target Nodes Computational Result for the Case of C1

	Problem Set			IP Model		Iterative Cluster Heuristic		ILS Algorithm		Sequential Segmentation Heuristic		Best Upper Bound Value
C1		|A|	$\|V_t\|$	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU (s)	GAP(%)	CPU(s)	GAP(%)	
	S2-10	402	10	2707.21	0.00	434.67	0.90	1.49	1.87	418.16	0.52	54.20
	S2-30	402	30	3600.00	31.75	1145.26	4.06	2.50	32.66	773.66	10.20	148.13
	S2-50	402	50	3600.00	-	3600.95	31.45	3.97	45.63	1851.93	22.46	207.40

For the scenario of the target nodes in C1, it was faster to solve the problem using the ILS algorithm. The SSH was the second fastest, and the ICH with the original solver was the slowest. We used a multiple local search algorithm in the ILS (MOVE, SORT, REPLACE, SWAP, and INSERT). Having more targets in a network requires more combinations that may not yet have been implemented in our algorithm. This is the main reason why increasing the targets increased the GAP (%) in the ILS. However, for the ICH and SSH, when more targets were considered, there were also multiple total objectives for each USV, and the USVs were required to search for all possible combinations. In addition, in the problem sets S2-50 and S2-30, almost all of the methods had a higher GAP (%). This was to be expected since the best upper bound value that was used as a comparison here was based on the upper bound value in the IP model. Therefore, it was possible that if the time limit for the computational result were to have been extended, these results would have been much lower. In conclusion, when larger targets are set in a network, for each method and the IP model, it will be more difficult to find the optimal solution.

Table 5.7. Scenario Target Nodes Computational Result for the Case of C2

	Problem Set			IP Model		Iterative Cluster Heuristic		ILS Algorithm		Sequential Segmentation Heuristic		Best Upper Bound Value
C2		|A|	$\|V_t\|$	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU (s)	GAP(%)	CPU(s)	GAP(%)	
	S2-10	402	10	2832.44	0.19	982.81	0.38	42.72	2.35	140.87	2.46	57.10
	S2-30	402	30	3600.00	46.00	2425.71	3.08	69.84	21.09	940.38	8.02	125.70
	S2-50	402	50	3600.00	-	3600.04	15.61	80.93	43.90	1815.76	20.87	187.80

doi:10.6844/NCKU202000986

The same pattern can be seen in C2 in this scenario compared to scenario 1. The SSH method performed faster in this scenario due to having both the ICH and ILS to find the PB path first. According to the computational results, the ILS was the fastest, followed by the SSH, and then the ICH. From the average GAP (%) results, the ICH performed the best in all cases, although it required more computational time. Aside from that, the average GAP (%) results were also lower compared to C1. This was probably because since in the case of C2, the PB path was not yet defined, there were many meeting points for both the PB and the USVs, thus allowing the model to reach an optimal solution easier at the cost of more computational time. Multiple meeting points affected the recharging points and, therefore, the battery consumption flow balance for each USV.

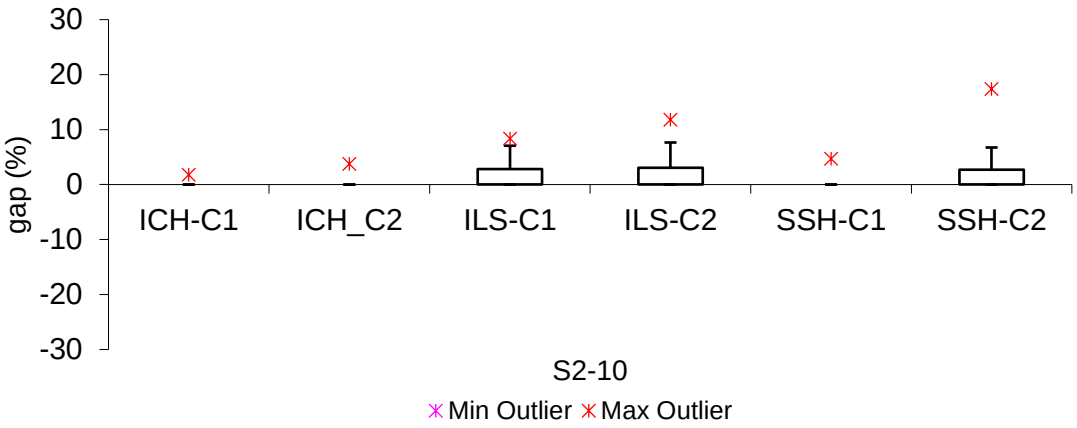


Figure 5.7. GAP (%) comparisons in S2-10 (100 USV Nodes and 50 Sets of Time)

To better understand each algorithm, the GAP (%) in S2-10 was compared to the same figure for all the methods. Figure 5.7 shows that even though the optimal solution may not be obtained in some methods, each method in C1 and C2 is also closer to 0 when the target is set to 10. This means that using the network size and the settings provided, these methods are able to perform almost as well as the IP model from the GAP (%), and they are faster.

doi:10.6844/NCKU202000986

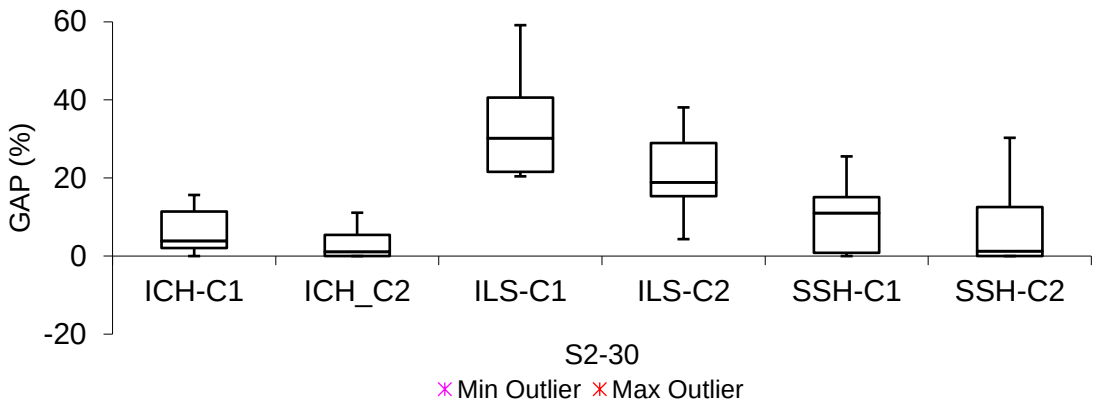


Figure 5.8. GAP (%) comparisons in S2-30 (100 USV Nodes and 50 Sets of Time)

According to Figure 5.8, when the target distance was increased, within the 1-hour time limit, none of the three methods could achieve optimality although the computational time was lower compared to the IP model. Compared to C1, in C2, it was easier to find an optimal solution in the same network set since the median for C2 was lower than that for C1.

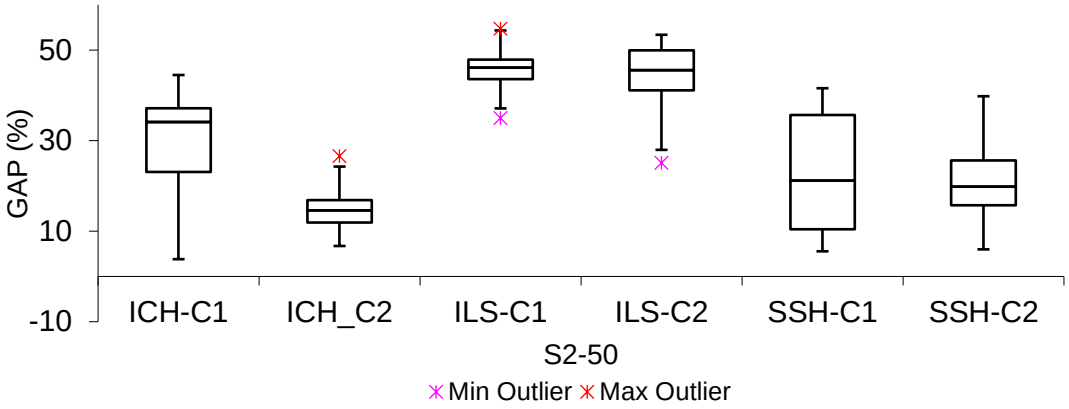


Figure 5.9.GAP (%) comparisons in S2-50 (100 USV Nodes and 50 Sets of Time)

According to Figure 5.9, when the target distance was increased, all three methods had an increased GAP (%). The GAP (%) for C2 in the ILS didn't change significantly because the meeting point swapping mechanism may not have led to the best combinations for the PB path. Therefore, this is a potentially good direction for the development of the swapping mechanism to consider the position of the target nodes.

5.4.3. Scenario Operation Time Nodes for Case C1 and C2

The same mechanism was applied to scenario 3. All the cases were solved in a 1-

doi:10.6844/NCKU202000986

hour time limit irrespective of whether either a feasible or optimal solution was found. The results of the experiment are shown in Table 5.8-5.9. All indicator values in the table were obtained using the Gurobi Solver. .

Table 5.8. Scenario Operation Time Computational Result for the Case of C1

				IP Model		Iterative Cluster Heuristic		ILS Algorithm		Sequential Segmentation Heuristic		Best Upper Bound Value
C1	Problem Set	|A|	T	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU (s)	GAP(%)	CPU(s)	GAP(%)	
	S3-30	402	30	2995.51	5.48	264.30	0.48	1.60	43.06	1307.80	2.44	85.40
	S3-50	402	50	3600.00	46.68	1167.60	1.76	2.31	19.06	1558.53	2.83	109.70
	S3-65	402	100	3600.00	-	3067.17	10.19	2.48	2.37	1647.99	2.67	105.50

For the operating time scenario, the fastest method by which to solve the problem was the ILS, followed by the SSH in the larger time-bound operation and ICH in the smaller time-bound operation. Based on the average GAP (%) results, the ICH performed the best in smaller time-bound operations, and the ILS performed the best in larger time-bound operations. The main reason for this was the nature of the branch-and-bound solver. When more operating time was included in the solver, the workspace in which the solver had to work increased exponentially. However, in the case of the SSH, since the workspace was kept at 20 sets of time in each segment, the additional operating time did not affect the computational time. In conclusion, the method that was affected the most by the branch-and-bound nature of the solver was the ICH since the solver was used in this method without segmentation.

Table 5.9. Scenario Operation Time Computational Result for the Case of C2

				IP Model		Iterative Cluster Heuristic		ILS Algorithm		Sequential Segmentation Heuristic		Best Upper Bound Value
C2	Problem Set	|A|	T	CPU(s)	GAP(%)	CPU(s)	GAP(%)	CPU (s)	GAP(%)	CPU(s)	GAP(%)	
	S3-30	402	30	1801	0.00	337.27	0.88	35.67	34.03	628.64	3.71	87.80
	S3-50	402	50	3600.00	78	1845.10	0.94	69.67	21.72	1101.49	1.71	104.40
	S3-65	402	100	3600.00	-	3600.00	7.55	88.74	2.78	1144.91	2.15	98.60

In the C2 case, all of the methods had similar performance. However, since the PB path was not yet defined, the ILS and ICH consumed more computational time. From the average GAP (%) perspective, since the ILS did not use the branch-and-bound design paradigm to solve the problem with more operating time, the optimal result could be

doi:10.6844/NCKU202000986

achieved in a short amount of time. Hence, we can conclude that if the user has more operating time, the ILS algorithm can be used to give an optimal solution faster as compared to the other methods under consideration.

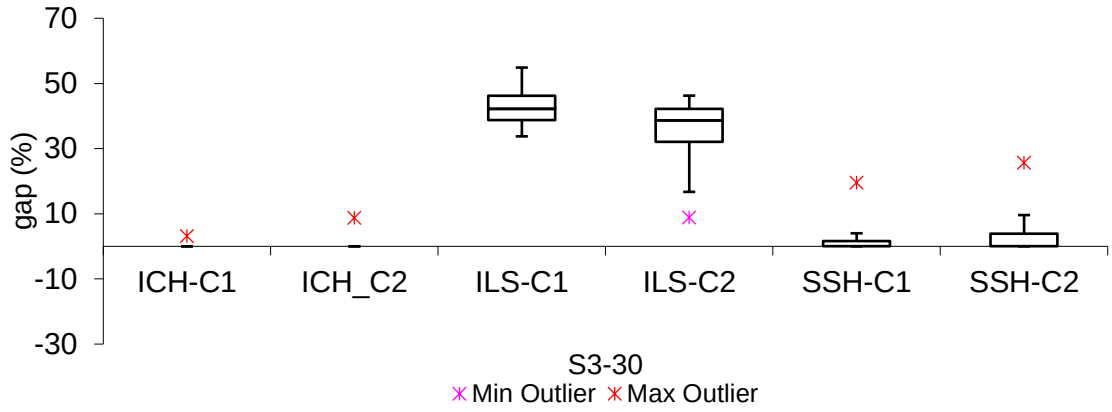


Figure 5.10.GAP (%) comparisons in S3-30 (100 USV Nodes and 20 Targets)

To better understand each algorithm, the GAP (%) in S3-10 was compared to the same figure for all the methods. Figure 5.10 shows that the ILS algorithm had the highest GAP (%) in a lower time-bound operation. It can be concluded that in a lower time-bound operation, this algorithm performs the worst out of the three methods although it is the fastest method.

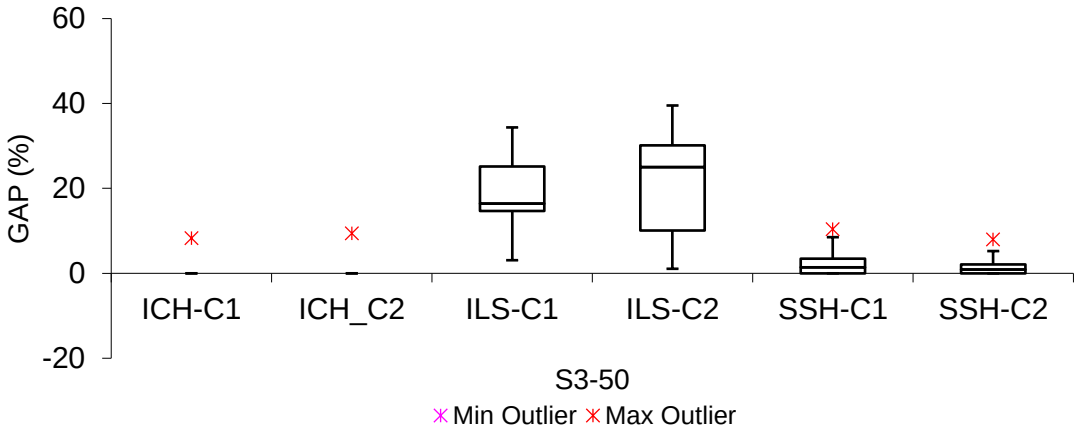


Figure 5. 11. GAP (%) comparisons in S3-50 (100 USV Nodes and 20 Targets)

According to Figure 5.11, when the time-bound was increased, within the 1-hour time limit, all three methods were affected. Based on the ILS algorithm, it was easier to reach optimality due to having more workspace to perform the local search. While SSH and ICH were able to stay within the optimality range due to setting the time-bound to be sufficient

doi:10.6844/NCKU202000986

for the problem set. Both C1 and C2 were able to obtain a lower GAP (%) when the time-bound setting was set at 50 sets of time.

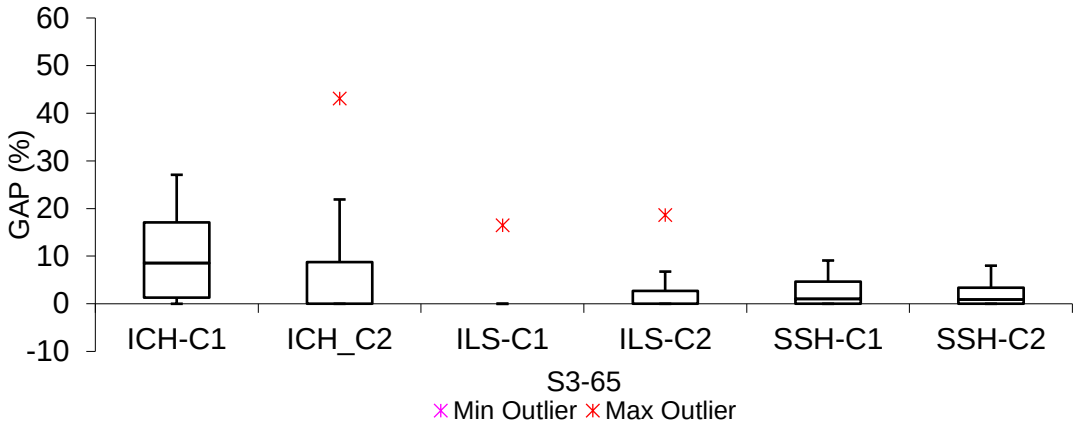


Figure 5. 12. GAP (%) comparisons in S3-65 (100 USV Nodes and 20 Targets)

According to Figure 5.12, when the time-bound was increased, the GAP (%) in all three methods was affected significantly. In the ICH method, due to the nature of the branch-and-bound algorithm, because the commercial solver (GUROBI) had more time-bound operation, this was a disadvantage. Having more time requires more searching space in a time-space network and therefore requires more computational time. Within the 1-hour time limit, there was not sufficient time for the model to find the optimal objective values. This behavior did not affect the SSH since the workspace was divided into time segmentation in this method. On the other side, having more time was an advantage for the ILS algorithm since having more time allowed more target insertion procedures and thus allowed a more local search procedure for the combinations of targets considered in the operation.

5.4.4. Scenario PB Path for Case C2

The same mechanism is applied to scenario 4. All the cases were solved in the 1-hour time limit irrespective of whether a feasible or optimal solution was found. The results of the experiment are shown in Table 5.10, all indicator values in the table were obtained using the Gurobi Solver.

doi:10.6844/NCKU202000986

Table 5.10.Scenario PB Area Computational Result for the Case of C2

			IP Model		Iterative Cluster Heuristic		ILS Algorithm		Sequential Segmentation Heuristic		Best Upper Bound Value
Problem Set	**|A|**	**|Vp|**	**CPU(s)**	**GAP(%)**	**CPU(s)**	**GAP(%)**	**CPU (s)**	**GAP(%)**	**CPU(s)**	**GAP(%)**	
S4-20	344	20	3600.24	16.12	1573.89	0.48	43.26	3.85	691,5	2.26	98.5
S4-40	388	40	3600.41	9.61	2829.70	2.50	120.34	0.93	842.58	9.78	98.2
S4-60	444	60	3600.00	--	3273.81	14.87	302.26	9.54	1029.85	13.28	104.2

In the case of C2, When the PB path was increased, there were more feasible solutions for the model to solve, thus explaining why the IP model, ICH, and SSH involved increased computational time in each increment. Another interesting finding was that due to having more PB paths to solve in ILS and ICH, both methods required a lot of computational time, especially the ILS algorithm. Having more PB path feasible solutions required multiple swapping node mechanisms in the ILS algorithm. In each swapping mechanism, both phase 1 and 2 were implemented to reconfirm the new USV solution and thus required a lot of computational time.

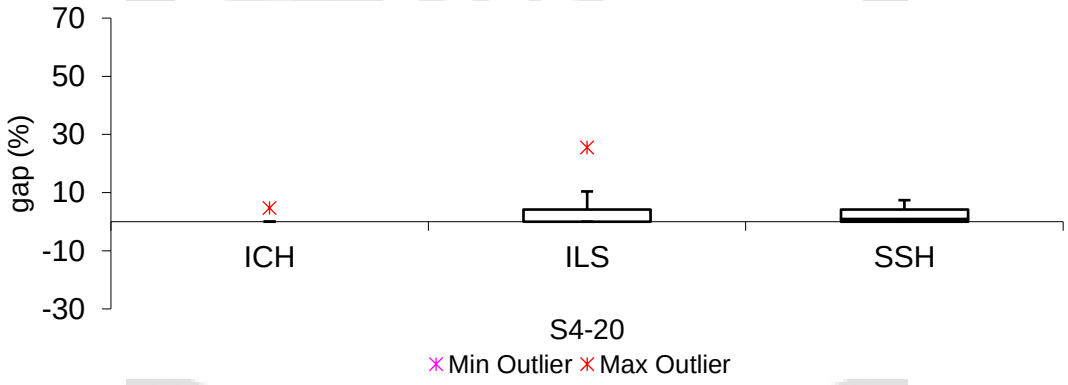


Figure 5.13. GAP (%) comparisons in S4-20 (30 Targets and 50 Sets of Time)

According to Figure 5.13, in a smaller PB area, the GAP (%) in all three methods was able to reach optimality within the given time-bound. Although the ICH had the highest computational time, it was the method with the lowest variance in terms of reaching the optimal solution, while both the SSH and ILS had higher variance. Having more PB area affected the SORTSPACE and SORTTIME local search procedures for the ILS and thus required more combinations to be considered in the development of this algorithm.

doi:10.6844/NCKU202000986

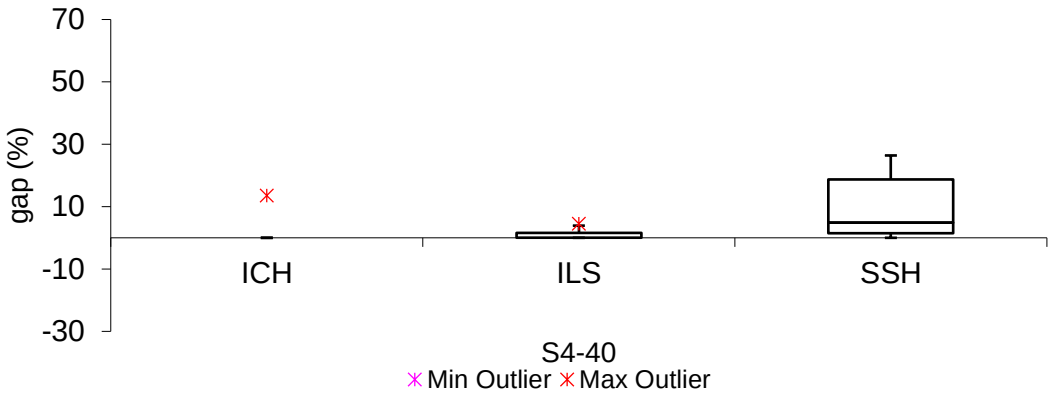


Figure 5.14. GAP (%) comparisons in S4-40 (30 Targets and 50 Sets of Time)

According to Figure 5.14, when the PB path was increased, the GAP (%) in all three methods reached optimality within the given time-bound. An interesting finding here is that in the SSH method, since the segmentation now was more likely to stop within the PB area, the variance within the results in this method also increased. There are multiple reasons for these results. First, this is because of possible solutions in the PB routes and also because more targets were assigned in the PB area instead of in the USV area, which may have caused the recharging points to be changed in each segment and in turn, affected the next segmentation. This means that when more segmentation is implemented in a wider PB area, the results generated from the SSH are more unreliable.

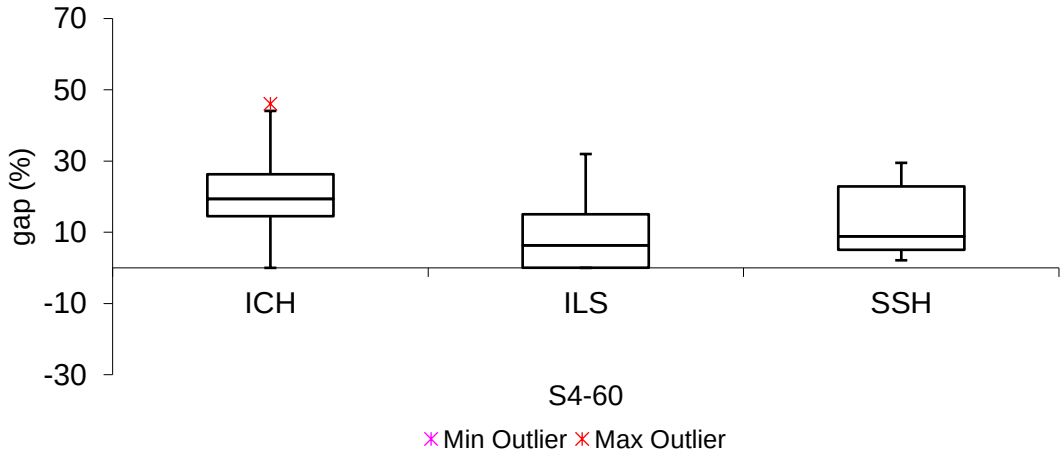


Figure 5.15. GAP (%) comparisons in S4-60 (30 Targets and 50 Sets of Time)

According to Figure 5.15, when the PB area is increased, the workspace for the ICH is increased as well. Unlike the previous figure, where when the PB area was increased, within the time-bound of the operation, the ICH was not able to reach the optimal solution.

doi:10.6844/NCKU202000986

For the other methods, aside from the computational time, adding more PB did not significantly affect SSH and ILS.

5.5. An Illustrative Example with a Real-world Scenario Setting

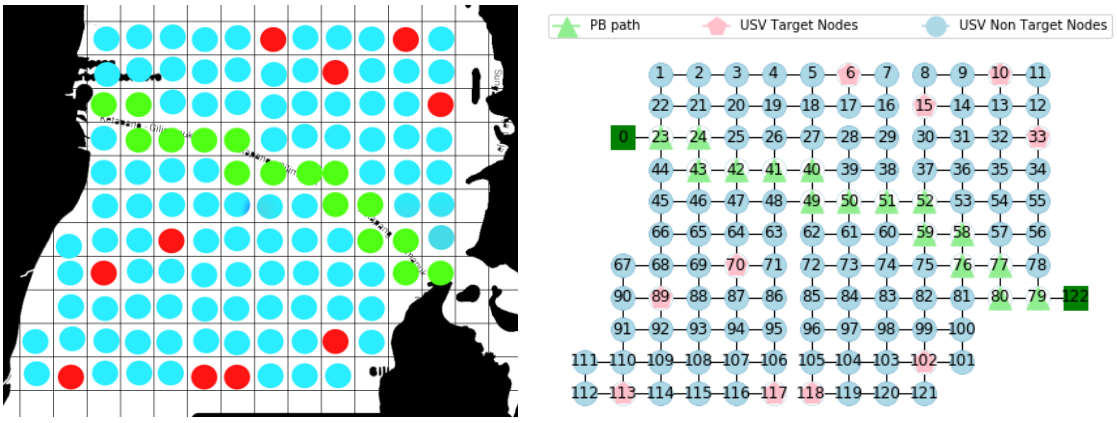
In this section, we report our experimental results using a scenario created based on an actual map to test our model, heuristics and algorithm. For the construction network of the real data, the grid network is taken from a real map located in Bali province, Indonesia. Since there are two main harbors connecting Bali island (Gilimanuk Harbour) and Java island (Ketapang Harbour) that can be used as the harbor for the PB, the route will follow the real-world data taken from google map and computed to the grid network with simulated targets.

(a) A Real-world Water Area in Bali, Indonesia (b) A Grid-Network Approximation

Figure 5.16. Google Map Data Conversion to Grid Network

Based on the real data and the distance measurement, the size of the grid is set to be 400 m x 400 m, with the total operation time of 35 to reflect the 35 minutes travel time of the transportation from Gilimanuk harbor to Ketapang harbor. There are a total of 10 randomized targets that are spread along with a total of 96 blue nodes (USV area nodes), green nodes represent the predetermined path of the PB (16 nodes), while the red nodes represent the targets.

doi:10.6844/NCKU202000986



(a) Representations of USV, PB, Target Nodes (b) A Converted Grid Network Approximation

Figure 5.17. Representation of Grid Network Nodes

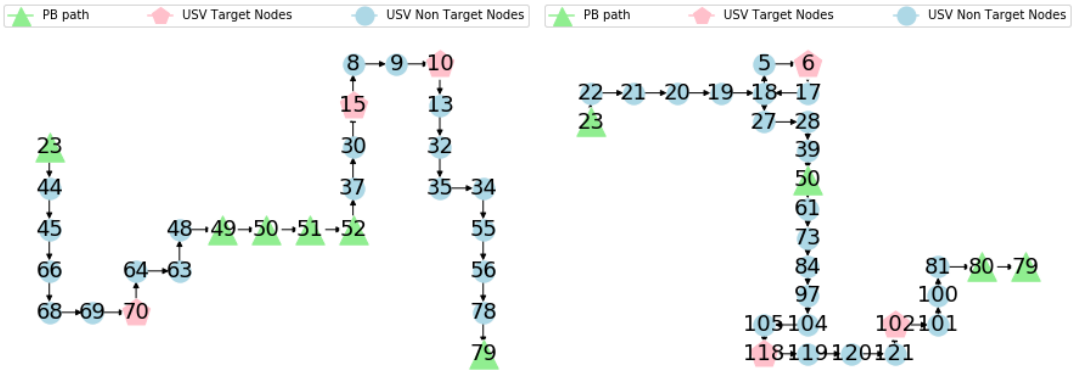
From this network, IP model, ICH, ILS and the SSH are implemented with the result obtained is shown in the table below. The computational results that we get are shown in Table 5.11. Each of the methods is limited to 3600 (s) maximum computational time to limit the time spent to solve the case.

Table 5. 11.Gilimanuk-Ketapang Experimental Results

Method	GAP (%)	Found Objective	CPU (s)	Upper Bound Objective
IP-model Commercial Solver	21.43	33	3601.95	42
Iterative Clustering Heuristic	4.76	40	351.66	42
Iterative Local Search Algorithm	26.19	31	1.76	42
Sequential Segmentation Heuristic	14.28	36	1732.15	42

Based on these results, the ICH method performs the best from GAP (%) perspective, and the ILS performs the best from CPU (%) perspective. From the commercial solver based result, the ICH performs the fastest, while for the SSH, with 10 sets of time for each segment, each iteration will require 20 sets of time and cause an unnecessary searching process that consumes time thus causing it to be outperformed by the ICH. These results prove that in real-world based data, all the methods can be implemented to solve the problem, and choosing the right method requires the right calculation of the resources that the user has.

doi:10.6844/NCKU202000986



(a) USV 1 Route (b) USV 2 Route

Figure 5 18. USVs Routing Result

From the computational results in Figure 5.18, in 35 sets of time, 6 numbers of 10 targets are saved. USV 1 retrieved target 70, 15, and 10 with total objective values of 23 while USV 2 retrieved target 6, 118, and 102 with total objective values of 17. The number of recharging is 4 times with drone 1 at time 12 at node 49 to node 50 and at time 14 at node 50 to node 51 and drone 2 at time 14 at node 50 to node 61 and at time 32 at node 80 to node 79. These computational results prove that the IP mathematical formulations is able to solve the problem with the detailed movement for each USV as well as the PB, the number of recharging time for each USV and the location the PB is required to move with the USVs, as well as which targets can give the highest total objective values for these targets retrieved.

5.6. Summary

Based on the results for C1 and C2, there were four scenarios used as the experiments. In the first scenario, we increased the USV nodes to understand the implications of adding USV nodes to the developed methods. In the case of C1, the ILS was the fastest method by which to solve each scenario, followed by the SSH and then the ICH. In the case of C2, the same pattern was observed. An interesting note here is that we could see the differences in not having a defined PB path in the case of C2 since the ICH and ILS had increased computational time due to having more phases during which to search for the optimal PB path. For the SSH and the ICH, adding more USV nodes increased the workspace for each USV, allowed more dynamic positions for each target, and increased the battery consumption flow balance, thus increasing the computational time with each addition.

doi:10.6844/NCKU202000986

In the second scenario, we increased the number of targets. We concluded that C1 did significantly increase the computational time for all three methods, with the fastest method being the ILS algorithm. From the GAP (%) perspective, the ICH and ILS performed the fastest, with the optimal solution in the first case (10 targets) in both C1 and C2. In the case of C2, aside from the same pattern in the computational time, the ICH method gave the best optimal solution out of the three methods, although this method consumed a lot of computational time. A possible reason behind this is because since in the case of C2, the PB path was not yet defined, there were many meeting points for both the PB and the USVs, thus allowing the model to reach an optimal solution easier at the cost of more computational time. An interesting finding from this experiment is that in a network where there are a lot of targets, ILS performs the worst to reach the optimal solution. This is because having more targets requires a lot of combinations for solutions that may not yet be considered in the current ILS algorithm. Therefore more local search procedures for the target's consideration can be a good consideration for this algorithm

In the third scenario, the time-bound for the operation time is increased in each scenario, due to the branch-and-bound nature of the solver, the IP model and the ICH is affected greatly. The ICH performs the best in smaller time-bound operations, and the ILS performs the best in larger time-bound operations. The main reason behind this is because due to the nature of branch-and-bound of the solver. However, in the case of SSH, since the workspace is kept the same in each segment, the addition of having more operational time does not affect the computational time. For the ILS, having more operational time does not affect ILS computational time and gave the ILS algorithm more opportunities to search for a better solution. According to our experiments, the ILS performs the best when there is more operational time in the model on both the case of C1 and C2.

In the fourth scenario for the case of C2, the ICH algorithm performs the best from the GAP (%) perspective due to having more options for feasible solutions while SSH performs the fastest with the ILS as the highest average GAP (%). Another interesting finding that can be observed from this scenario is that the more segmentations in the SSH method, the harder for the method to find the optimal solution. For the ILS, having more PB path feasible solution increases the GAP (%) the most. This is because of the addition of phase 3 in the ILS algorithm to search for the PB path, and since, in this scenario, the PB path is increased, the ILS will have to search for more possibilities that may not yet be

doi:10.6844/NCKU202000986

considered in the current local search mechanism. This is a good direction for the development of this algorithm to consider more combinations for PB path local search. In the real world scenario setting, all of our proposed heuristics and algorithm were able to solve the problem faster compared to the commercial solver with good results. Therefore, we can conclude that the proposed methods are able to be implemented in the proposed problem settings.

doi:10.6844/NCKU202000986

Chapter 6

CONCLUSIONS AND FUTURE RESEARCH

6.1. CONCLUSIONS

This thesis focused on the joint operation of a parent boat and unmanned surface vehicles (USVs) with limited battery capacity. This limitation was solved by providing moving recharging stations with a parent boat. The objective of this research was to maximize the total value of the targets rescued within a limited operating time and to calculate the optimal routes that satisfy the energy constraints along the process. There were two main cases tested in this research. The first case (C1) was a case where the parent boat (PB) path was already defined, while in the second case (C2), only the PB area was known. Therefore, the PB path in the case of C2 was not known. Since the original commercial solver for the IP model takes a lot of time, we constructed an Iterative Clustering Heuristic, a Sequential Segmentation Heuristic, and an Iterative Local Search (ILS) algorithm to solve the model faster.

We tested these methods based on four scenarios that comprised both experimental- and real-world-based data. In scenario 1 the USV nodes were increased; in scenario 2 the target nodes were increased; in scenario 3 the time-bound for the operation was increased, and scenario 4, which was the case of C2, the implications of having more PB area was examined for the three methods. The conclusions derived from our study are as follows:

1. The Iterative Clustering Heuristic (ICH) was created to limit the movement of the USVs by assigning each USV to a cluster, and then the solver was applied to solve the simplified problem.
2. The Iterative Local Search (ILS) algorithm was applied by performing multiple local search procedures (MOVE, INSERT, SWAP, REPLACE, SORT) for the targets that could be covered by each USV, followed by the local search procedures for the meeting points of the USVs and the PB.
3. The Sequential Segmentation Heuristic (SSH) was applied by dividing the problem into multiple segmentations by time and predicting the targets covered in each segment, after which the solver was applied to solve the simplified problem.
4. The structure for the network used played an important role in the model to solve the

doi:10.6844/NCKU202000986

problem. We tested C1 and C2 to understand the effects of having a defined PB path beforehand to the model. In the case of the ICH and ILS algorithms, the computational time for C2 was increased due to having more phases to search for the PB path. However, in the case of C1, the fastest method to obtain a result was the ILS algorithm in all scenarios, followed by the SSH.

5. When more targets were considered in the problem, a longer computational time was needed. In the comparison scenarios, within the 1-hour time limit, the original solver, the heuristics, and the ILS algorithm's *Gap (%)* were increased as targets were added to the model. This means that with more targets, the model was forced to find a global optimum that considered all targets.
6. Although each method has its own advantages, our experiments showed that each method has its own weaknesses as well. In the case of the Iterative Clustering Heuristic, the main weakness is the design for each clustering (*k *clusters*) and the time-bound for the operation due to the branch-and-bound nature of the solver. For the Sequential Segmentation, the main weakness is the time covered in each segment, as proven in scenario 1 (USV increment) and scenario 4 (PB area increment). The ILS algorithm, was highly dependent on the operational time covered and the number of total targets since having more time allows the algorithm to do more target insertion local searches.
7. In the case of C2, although most of the results showed that it required more computational time as compared to C1, the scenarios proved that in the same network, the GAP (%) was generally lower for C1. One of the reasons for this was that with the PB path not yet defined, there were more feasible meeting points for both vehicles, which thus allowed more feasible solutions, although it resulted in a need for more computational time. Another finding from the experiments was that the defined PB path might not be the optimal PB path for the problem set. Thus, the addition of a more comprehensive mechanism provided a better solution for the original PB path.
8. We tested each method to a real map located in Bali province, Indonesia. There were 10 targets to be retrieved from two main harbors connecting Bali (Gilimanuk Harbor) and Java (Ketapang Harbor) with a route that follows the real-world data taken from Google Maps and are computed to the grid network. The same patterns can be observed that the ILS provides the fastest result while the ICH provides the highest optimal result and the SSH to have a faster and higher total objective result compared

doi:10.6844/NCKU202000986

to the original IP model.

6.2. FUTURE RESEARCH SUGGESTIONS

There are some related issues worthy of future investigation. Here is a list of our suggestions based on our findings:

1. Range Communication between USV

   In our setting, we assume that USV can always communicate with the Parent boat at any time to arrange the meeting (e.g., through satellite connection). However, in practice, the connection between USV and Parent boat may require a certain distance to do the communication that will make the routing to be more difficult
2. USV speed and battery consumption

   To simplify our problem, we assumed that the movement in each node in USV nodes is faster than the parent boat nodes to provide the flexibility of the USV compared to the Parent boat. However, it is also possible that in certain nodes in USV nodes to consume more time and more battery consumption that will provide more realistic implications on the USV's battery consumption.
3. USV Area pressure and nodes movement

   We did not consider the possibility of the USV area movement (e.g., sea waves, or wind movement) in our model. However, the possibility of adding the movement to the node will make the model more realistic to match the real-world scenario.
4. Time Windows for each target

   Due to the nature of the sea, it is also possible that each target may move after a certain time or not be available at a certain time. Adding the possibility of a more realistic nature of the environment to the model would make the problem more practical.
5. Consideration of the use of several types of USVs

   In our model, we assumed that the fleet of USVs to be the same type. This assumption resulted in all of the USV to have the same battery capacity and speed. In a real situation, one may deploy different types of USV for different missions.
6. Develop better algorithm and math programming models

   Since the computational time to solve the mathematical model was quite time-

doi:10.6844/NCKU202000986

consuming, it is possible to develop the algorithm to provide the same solution to the solver model. Another opportunity is to design a better sequence in the ICH (k* *clusters*) to understand how the sequence of clusters can affect the performance of the model. Developing an algorithm for the multiple PB and multiple USVs case is also a good opportunity for future research since there has been little research around this topic.

doi:10.6844/NCKU202000986

Reference

Avellar, G., Pereira, G., Pimenta, L., & Iscold, P. (2015). Multi-UAV Routing for Area Coverage and Remote Sensing with Minimum Time. *Sensors*, *15*(11), 27783-27803. doi:10.3390/s151127783

Campbell, S., Naeem, W., & Irwin,G. (2012). A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance maneuvers. *Annual Reviews in Control*, vol. 36, No. 2, pp. 267-283.

Carlsson, J. G., & Song, S. (2017). Coordinated Logistics with a Truck and a Drone. *Management Science*, *64*(9), 4052-4069. doi:10.1287/mnsc.2017.2824

Cuda, R., Guastaroba, G., & Speranza, M. G. (2015). A survey on two-echelon routing problems. *Comput. Oper. Res.*, *55*(C), 185-199. doi:10.1016/j.cor.2014.06.008

Cruz Chávez, M., Rodríguez-León, A., Rivera-Lopez, R., & Cruz-Rosales, M. (2019). A Grid-Based Genetic Approach to Solving the Vehicle Routing Problem with Time Windows. *Applied Sciences*, *9*, 1. doi:10.3390/app9183656

Dallard, H., Lam, S. S., & Kulturel-Konak, S. (2007, 13-15 Aug. 2007). *Solving the Orienteering Problem Using Attractive and Repulsive Particle Swarm Optimization.* Paper presented at the 2007 IEEE International Conference on Information Reuse and Integration.

El-Hajj, R., Dang, D.-C., & Moukrim, A. (2016). Solving the Team Orienteering Problem with Cutting Planes. *Computers & Operations Research*, *74*. doi:10.1016/j.cor.2016.04.008

Ferrandez, S. M., Harbison, T., Weber, T., Sturges, R., & Rich, R. (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management*, *9*(2), 15. Retrieved from http://inis.iaea.org/search/search.aspx?orig_q=RN:48050833.

Fu, Z., Yu, J., Xie, G., Chen, Y., & Mao, Y. (2018). A Heuristic Evolutionary Algorithm of UAV Path Planning. *Wireless Communications and Mobile Computing*, *2018*, 1-11. doi:10.1155/2018/2851964

Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, *61*(12), 1258-1276. doi:10.1016/j.robot.2013.09.004

doi:10.6844/NCKU202000986

Garone, E., Naldi, R., Casavola, A., & Frazzoli, E. (2010). Cooperative mission planning for a class of carrier-vehicle systems. *In 49th IEEE Conference on Decision and Control (CDC).* Atlanta, Georgia, USA: IEEE

Gunawan A., Lau H.C., Lu K. (2015) An Iterated Local Search Algorithm for Solving the Orienteering Problem with Time Windows. In: Ochoa G., Chicano F. (eds) *Evolutionary Computation in Combinatorial Optimization.* EvoCOP 2015. Lecture Notes in Computer Science, vol 9026. Springer, Cham

Gunawan, A., Ng, K.M., Yu, V. F., Adiprasetyo, G., and Lau, H. C. Iterated local search algorithm for the capacitated team orienteering problem. (2018). *Proceedings of the 12th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2018), Vienna, Austria, August 28-31*. 493-496. Research Collection School of Information Systems.

Kareem, M. M., Ismail, M., Altahrawi, M. A., Arsad, N., Mansor, M. F., & Ali, A. H. (2018, 26-28 Nov. 2018). *Grid Based Clustering Technique in Wireless Sensor Network using Hierarchical Routing Protocol.* Paper presented at the 2018 IEEE 4th International Symposium on Telecommunication Technologies (ISTT).

Khare, N., & Singh, P. (2011). Modeling and optimization of a hybrid power system for an unmanned surface vehicle. *Journal of Power Sources*, ***198***, 368–377. doi:10.1016/j.jpowsour.2011.09.080

Khan, A., Noreen, I., & Habib, Z. (2017). On Complete Coverage Path Planning Algorithms for Non-holonomic Mobile Robots: Survey and Challenges. *Journal of Information Science and Engineering, 33*(1), 101-121. doi: 10.6688/JISE.2017.33.1.7

Labadie, N., Mansini, R., Melechovský, J., & Wolfler Calvo, R. (2012). The Team Orienteering Problem with Time Windows: An LP-based Granular Variable Neighborhood Search. *European Journal of Operational Research, 220*(1), 15-27. doi:https://doi.org/10.1016/j.ejor.2012.01.030

Luo, Z., Liu, Z., & Shi, J. (2017). A Two-Echelon Cooperated Routing Problem for a Ground Vehicle and Its Carried Unmanned Aerial Vehicle. *Sensors, 17*(5), 1144. doi:10.3390/s17051144

Mukhina, K. D., Visheratin, A. A., & Nasonov, D. (2019). Orienteering Problem with Functional Profits for multi-source dynamic path construction. *PLOS ONE, 14*(4), e0213777. doi:10.1371/journal.pone.0213777

doi:10.6844/NCKU202000986

Matos, A., Cubber, G.D., Doroftei, D., Balta, H., Silva, E., Serrano, D., Govindaraj, S., Roda, R., Lobo, V., Marques, M., et al. (2017). Operational Validation of Search and Rescue Robots. *In Search and Rescue Robotics; IntechOpen*: doi: 10.5772/intechopen.69497

Mathew, N., Smith, S. L., & Waslander, S. L. (2015). Multirobot Rendezvous Planning for Recharging in Persistent Tasks. *IEEE Transactions on Robotics, 31*(1), 128-142. doi:10.1109/TRO.2014.2380593

Moravec, H., & Elfes, A. (1985). High resolution maps from wide angle sonar. *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. doi:10.1109/robot.1985.1087316

Mirhedayatian, S. M., Crainic, T. G., Guajardo, M., & Wallace, S. W. (2019): A two-echelon location-routing problem with synchronization, *Journal of the Operational Research Society*, DOI: 10.1080/01605682.2019.1650625

Perez-Carabaza, S., Besada-Portas, E., Lopez-Orozco, J. A., & de la Cruz, J. M. (2018). Ant colony optimization for multi-UAV minimum time search in uncertain domains. *Applied Soft Computing, 62*, 789-806. doi:https://doi.org/10.1016/j.asoc.2017.09.009

Specht, C., Świtalski, E., & Specht, M. (2017). Application of an Autonomous/Unmanned Survey Vessel (ASV/USV) in Bathymetric Measurements. *Polish Maritime Research, 24*, 36-44. doi:10.1515/pomr-2017-0088

Vansteenwegen, P., Souffriau, W., Berghe, G. V., & Oudheusden, D. V. (2009). Iterated local search for the team orienteering problem with time windows. *Computation Operation Research, 36*(12), 3281-3290. doi:10.1016/j.cor.2009.03.008

Wang, X., & Wallace, S. (2016). Stochastic scheduled service network design in the presence of a spot market for excess capacity. *EURO Journal on Transportation and Logistics, 5*, 393–413. doi:10.1007/s13676-015-0089-1

Yahiaoui, A.-E., Moukrim, A., & Serairi, M. (2019). The clustered team orienteering problem. *Computers & Operations Research, 111*, 386-399. doi:https://doi.org/10.1016/j.cor.2019.07.008

Yuan, Y., & Tang, L. (2017). Novel time-space network flow formulation and approximate dynamic programming approach for the crane scheduling in a coil warehouse. *European Journal of Operational Research*. doi:10.1016/j.ejor.2017.03.007

Zhang, J., Jia, L.-m., Niu, S., Zhang, F., Tong, L., & Zhou, X. (2015). A Space-Time Network-Based Modeling Framework for Dynamic Unmanned Aerial Vehicle Routing

doi:10.6844/NCKU202000986

in Traffic Incident Monitoring Applications. *Sensors (Basel, Switzerland), 15,* 13874-13898. doi:10.3390/s150613874

Zhang, J., Zhang, F., Liu, Z., & Li, Y. (2019). Efficient Path Planning Method of USV for Intelligent Target Search. *Journal of Geovisualization and Spatial Analysis, 3*(2), 13. doi:10.1007/s41651-019-0035-0

doi:10.6844/NCKU202000986