Approximation methods to vehicle routing problem for a drone fleet management

Setyawan Ajie Sukarno

Université Polytechnique HAUTS-DE-FRANCE

Thèse de doctorat

Pour obtenir le grade de Docteur de

l'UNIVERSITE POLYTECHNIQUE HAUTS-DE-FRANCE

Discipline, spécialité selon la liste des spécialités pour lesquelles l'Ecole Doctorale est accréditée :

INFORMATIQUE

Présentée et soutenue par Setyawan Ajie, SUKARNO.

Le 04/12/2019, à l'Université Polytechnique Hauts-de-France

Ecole doctorale :
Sciences Pour l'Ingénieur (ED SPI 072)

Equipe de recherche, Laboratoire :
Laboratoire d'Automatique, de Mécanique et d'Informatique Industrielles et Humaines (LAMIH - UMR 8201)

Méthodes d'Approximation au Problème de Routage de Véhicule pour une Gestion de Flotte de Drones

JURY

Président du jury

- Djemai, Mohamed. Professeur. LAMIH, Université Polytechnique Hauts-de-France.

Rapporteurs

- Ben Othman, Jalel. Professeur. Institut Galilée, Université Paris 13.
- Chu, Feng. Professeur. IBISC, Université Paris-Saclay.

Examinateurs

- Sehab, Rabia. MdC. Pôle S2ET, ESTACA.
- Djemai, Mohamed. Professeur. LAMIH, Université Polytechnique Hauts-de-France.

Directeur de thèse : Ben Atitallah, Rabie. MdC, HdR. Computer Science Department, University of Galatasaray, Turquie
Co-directeur de thèse : Delot, Thierry. Professeur. LAMIH, Université Polytechnique Hauts-de-France.

Acknowledgements

First of all, I would like to thank my supervisors, DR. (MdC, HdR) Rabie Ben Atitallah and Prof. Thierry Delot, for giving me that opportunity to work with them. Especially to Mr. Rabie for his valuable support and continuous encouragement. And also for my particular advisor, Prof. Mohamed Djemai, for his important inspiration and immense patience. He was step by step beside me by his advice and assistance. His guidance helped me in writing of this thesis.

I would like to thank my thesis reviewers Prof. Jalel Ben Othman and Prof. Feng Chu for their time to review this manuscript and for their feedback. I would also like to thank Prof. Mohamed Djemai for presiding over my thesis defence jury as well as DR. Rabia Sehab.

I would very happy to send a special thanks for Prof. Elhadj Dogheche for offering me an opportunity to do my PhD study in Valenciennes, when he visited Politeknik Manufaktur Bandung in 2014. He always available for assisting me since 2012, when I did my master study in France, until I finished my PhD study. Another special thanks for Prof. Irwan Katili for his effort, support and passion in sending me to France in 2012. This achievement would never exist without his favor. And also for my supporter from Politeknik Manufaktur Bandung, Mr. Hadi Supriyanto and Mr. Aris Budiyarto.

This accomplishment doesn't just belong to me. I wouldn't be up here if it weren't for some very important people in my life who encouraged, sustained, blessed and inspired me, whom I call family and friends. So I would like to thank my colleagues, friends and brothers in LAMIH, Mousseron residence, Valenciennes and France for their help and love. Karim Ali for being by my side, literally, since my first day in LAMIH. You aren't just a friend for us, but also a part of our family. Pipit Anggraeni for struggling together in the beginning part of our PhD study. Placide Nduwayo for the unforgetable memories in Mousseron residence, especially when we were cooking and watching football together. Yazid for sharing happiness and misery in LAMIH. Marko and Is Mat for bringing laugh to our days in laboratory. And also for Aymen, Ayoub, Hanane, Lydia, Rania, Naila, Sophia, Zeineb, Valérie, Yun Fei, Walid, Ihsen, The 2 Mohameds, one from Lybia, and the other from Algerie, Danijela, Van Anh, Tareq, Tarek, and Mokhtar. Thanks to Wahyu, Anung, Fida, Bagus and Zahra for welcoming my guests on the defence day, and my Indonesian friends for becoming such a nice family, especially Mas Antok and Mbak Animun Farida.

My very special thanks for my mother, Setyati Widowati, for supporting me and praying for me every single day. And also for my children, Muhammad Rijal Firdaus, Hamzah Aynan Firdaus, and Marouane Askar Firdaus, who always become my oasis and my power charger. They are the best thing that ever happened in my life.

As I always safe the best for the last, I would like to grant this gratitude to one of the most important person in my life, my wife, Surniahyati, for standing by me in whatever the seasons, conditions and situations.

Setyawan Ajie Sukarno
December 2019

Abstract

Nowadays, drone plays a big role in civilian purposes, and it will getting bigger and more important in the future. Because of it's flexibility and versatility, the application of drone is more extensive. In certain fields of implementation, applying a team of drones will improve the effectiveness and efficiency of the application, such as in search and rescue, military purpose, agriculture and surveillance. Recently, there is a challenging issue to manage a team of drones in order to achieve the given mission. This issue opens many research opportunities, and our project is made to answer this challenge, to develop a platform for managing a fleet of drones. Among several approaches, vehicle routing problem (VRP) is one of a considered study to answer this challenge, in order to allocate the tasks and find the best path for each drone, with several constraints to be considered. There are many methods to solve VRP, and could be categorized into two groups i.e. exact and approximation method. But since VRP is classified as an NP-hard optimization problem, an approximation method is considered to be implemented in this project. Genetic Algorithm (GA), an approximation method which designed by an inspiration to the evolutionary ideas of genetic and natural selection, is applied in this project, since it is one of most used algorithm to solve VRP, among several approximation method. We observed that GA is suitable to be implemented in this project, but when the number of to-be-visited-points is hugely augmented, the number of iterations to get a satisfactory result would be extremely increased. This issue led us to hybridize GA with Clarke and Wright's saving algorithm (SA) in order to generate the initial population, so that it is no longer randomly generated as usually done. Eventually, this proposed hybrid method can improve the performance of the algorithm very satisfactorily, and reduce the number of iteration more than 90%.

Furthermore, a dynamic scenario in VRP is taken into account in this work i.e. an emerge of one or several new points that appear once the mission is already launched, and require a visit by a single drone. To deal with this dynamic scenarios, a Reverse Open Vehicle Routing Problem (ROVRP) is considered to be implemented. We decide to choose a heuristic method in solving the ROVRP, as it is classified as an NP-hard optimization problem, and we prefer to apply Clarke and Wright's Saving Algorithm (SA) in this project, due to their speed and simplicity. In our point of view, speed is one most considerable thing in choosing algorithm to solve dynamic scenario in VRP. Our proposed method is devided into two phases i.e. clustering and routing. And the experimental results show that our proposed method can give more than 95% accuracy.

In order to simulate and investigate the proposed methods, a Graphical User Interface (GUI) is developed. There are some available framework to develop this tool, and Netlogo is considered as the chosen framework.

Keywords Vehicle Routing Problem - Genetic Algorithm - Saving Algorithm - Dynamic Scenario - Reverse Open Vehicle Routing Problem - Graphical User Interface.

Résumé

Aujourd'hui, le drone joue un rôle important dans les activités civiles et deviendra de plus en plus important à l'avenir. Récemment, de nouvelles tendances se dirigent vers la gestion d'une flotte de drones afin de réaliser les missions donné Ce problème ouvre de nombreuses idée de recherche, et notre projet est fait pour répondre au défi, développer une plateforme de gestion de flotte de drones. Entre plusieurs approches, le problème de routage de véhicule (VRP) est une étude parfaite pour relever ce défi, afin de répartir les tâches et de trouver le meilleur chemin pour chaque drone, en tenant compte de plusieurs contraintes. Comme les VRP sont classés comme un problème d'optimisation NP-hard, une méthode d'approximation est considérée comme mise en œuvre dans ce projet. L'algorithme génétique (GA), est appliqué dans ce projet, puisqu'il s'agit de l'un des algorithmes les plus utilisés pour résoudre le VRP parmi plusieurs méthodes d'approximation. Nous avons observé que l'AG convient pour être utilisé dans ce projet, mais lorsque le nombre de points à traiter serait considérablement accru, le nombre d'itérations pour obtenir un résultat satisfaisant sera extrêmement augmenté. Ce problème nous a amenés à hybrider GA avec l'algorithme de sauvegarde (SA) afin de générer la population initiale, pour qu'elle ne soit plus générée aléatoirement comme d'habitude. Comme nous l'avons testé, cette méthode proposée peut améliorer les performances de l'algorithme de manière très satisfaisante et réduire le nombre d'itérations de plus de 90%.

De plus, un scénario dynamique dans le VRP est pris en compte dans ce travail, c'est-à-dire l'émergence d'un ou plusieurs nouveaux points qui apparaissent quand la mission a déjà été lancée et qui nécessitent une visite d'un seul drone. Pour faire face à ce ce scénario dynamique, un problème de routage de véhicule ouvert en sens inverse (ROVRP) est considéré. Le ROVRP est utilisé pour définir un ensemble d'itinéraires de véhicules de retour au dépôt, lors de la construction de nouveaux chemins en raison d'un scénario dynamique. Nous décidons de choisir une méthode heuristique pour résoudre le ROVRP, puis q'il s'agit d'un problème d'optimisation NP-hard, et nous préférons appliquer l'algorithme de sauvegarde (SA) à ce projet, en raison de leur rapidité et de leur simplicité. A notre point de vue, la vitesse est l'un des aspects les plus importants du choix d'un algorithme pour résoudre un scénario dynamique dans un VRP. Notre méthode proposée est divisée en deux phases : regroupement et routage. Et nos résultats expérimentaux montrent que notre méthode proposée peut donner plus de 95% de précision.

Afin de simuler et d'examiner les méthodes proposées, une interface utilisateur graphique (GUI) est développée. Il existe un cadre disponible pour développer cet outil, et Netlogo est considéré comme le cadre choisi.

Mots clés Problème de Routage de Véhicule - Algorithme Génétique - Algorithme de Sauvegarde - Scénario Dynamique - Problème de Routage de Véhicule Ouverte en Reverse - Interface Utilisateur Graphique.

List of Abbreviation

— **AS** : Ant System

— **CVRP** : Capacitated Vehicle Routing Problem

— **DA** : Deterministic Annealing

— **DARP** : Dial-A-Ride Problem

— **DVRP** : Dynamic Vehicle Routing Problem

— **EA** : Evolutionary Algorithm

— **GA** : Genetic Algorithm

— **GASA** : Genetic Algorithm and Saving Algorithm Hybrid

— **GUI** : Graphical User Interface

— **NN** : Neural Network

— **NP** : Non-deterministic Polynomial-time

— **OVRP** : Open Vehicle Routing Problem

— **ROVRP** : Reverse Open Vehicle Routing Problem

— **SA** : Saving Algorithm

— **SiA** : Simulated Annealing

— **TS** : Tabu Search

— **TSP** : Traveling Salesman Problem

— **VRP** : Vehicle Routing Problem

— **VRPTW** : Vehicle Routing Problem with Time Windows

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The context of the work

It was August 29, 2005, when Hurricane Katrina hit the Gulf Coast in United States. 2 days after, several drones with cameras, microphones and sensors, were involved in searched of victims and also assessed damage and sent back images from places rescuers couldn't get.

Technology of drones, and how humans interact with it, has improved. This technology is changing the way we handle. In the case of disaster relief and recovery, this means more effective ways to save lives and begin the arduous process of rebuilding after catastrophe, like a massive flooding after abundant rainfall in Bosnia and Herzegovina in Spring 2014 [30]. The fundamental advantages of drones are their cost effective, lightness, autonomous, and ability to perform dangerous in fields that human must avoid, such an assessment at Fukushima nuclear plant after it's accident [113], or a surveillance for maintenance in very difficult areas such as in hydroelectric power plant or subway tunnel. Now, 14 years after the Hurricane Katrina, drones are widely used today to play the big role in the civilian and also military world.

The growth and advancements in the studies and researches on drones are very vast in recent decades, e.g. in terms of its capability to autonomous navigation and mission planning [7] [20], its ability to simultaneous localization and mapping [70] [66], trajectory generation of fleet of droneseev [81], coordination of networked drones [121] [122] [114], etc. Because of their flexibility and versatility of use, the drones application field is more extensive, covering surveillance [30], intelligent logistics, maintenance, search and rescue [30], agriculture [32], scientific studies, etc. We believe, the study of drones will be a very potential and extensive study in the future, because it acts like a plane but smarter than a plane since it's got all sorts of on-board electronics to let it do preprogrammed functionality. There are a lot of research studies in drones. Most of the studies are related to the development its ability in self-navigation, localization and mapping whether in indoor or outdoor areas, with or without GPS [7].

But nowadays, the study of drones that is also already being done by several researchers are about a team of drones that do some cooperation and coordination. A drone team can accomplish a given task more quickly than a single drone can, by dividing the task into sub-tasks and executing them concurrently. A team can also make effective use of specialists designed for a single purpose (e.g., scouting an area, picking up objects, hauling payload), rather than requiring that a single drone be a generalist, capable of performing all tasks but expert at no tasks. In the studies on multi drones coordination, the discussions about how to make a cooperative assignment and task scheduling system for each drone under an a-priori known area has been widely explored, e.g. by [13] [2], as the examples. Recently, there are some motivating studies in the coordination of multi drones over a dynamic environment.

This study could be structurized in 3 levels of application, as shown in Fig. 1.1, such as : strategic level, tactical level and operational level. The strategic level is dealing with the

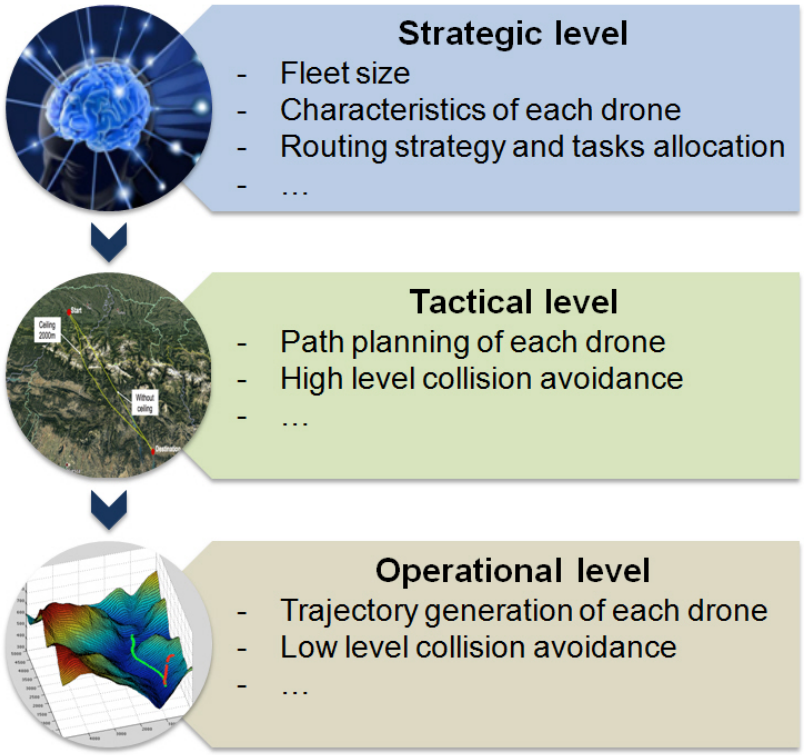

FIGURE 1.1 – Architecture of The Decision Making System

fleet size, characteristics of each drone, routing strategy, tasks allocation, etc. The tactical level is dealing with the path planning of each drone, high level collision avoidance, etc. And operational level is dealing with trajectory generation, low level collision avoidance, driver system, onboard computer, etc.

This thesis is dealing with the strategic level of application. The challenges arise in coordinating all of these drones to perform a single and global mission, to achieve the goals of each concurrent assignment, and also to generate the path planning for each and all drones.

The greater challenges emerge in adding the dynamic aspect in the mission. The word dynamic means each time new requests appear during the mission execution, our system must decide which drone will visit that new requests and along which route. Hence, the challenges is to design algorithms that enable dynamic task allocation and vehicle routing.

This thesis presents approximation algorithms to design an algorithm of a fleet of drones in a dynamic environment. The approach allows groups of agents to complete tasks in uncertain and dynamically changing environments, where new task requests are generated during the mission execution. Applications may include surveillance and monitoring missions, as well as distribution networks and delivery process. We consider the following scenario as an inspiring example : a team of drones perform a regular surveillance mission around an erupted volcano,

each of them is equipped with close-range high-resolution on-board sensors. Several sensors are deployed in certain area around the crater, in order to detect suspicious activity in a region of interest. Whenever a sensor detects a potential action, a request for close-range observation by one of the drones is generated. In response to this request, a drone visits the location to gather close-range information and investigate the cause of the alarm.

The coming Fig. 1.2 gives a clear view for the scenarios mentioned above. Fig. 1.2(a) a team of 3 drones is required to service a set of 12 points in a 3-dimensional space. The set of drone $\{D1, D2, D3\}$ are set to visit a set of points $\{P3 - P4 - P5 - P6\}$, $\{P10 - P9 - P8 - P7\}$ and $\{P11 - P0 - P1 - P2\}$ respectively. In Fig. 1.2(b) the mission is already started, and some drones has already visit some points. Then, in Fig. 1.2(c) few new set of points $\{P12, P13, P14\}$ appear and require to be visited by a drone. New paths are reconstructed, as seen in Fig. 1.2(d), since the appearance of few new points. Accordingly, this thesis presents task allocation and routing algorithms in a static and dynamic environment for a team of drones. This thesis applies the Vehicle Routing Problem (VRP) to deal with the task allocation and routing algorithms for a fleet of drones in accomplishing the given mission. The problem of planning routes through service demands that arrive during a mission execution is known as the Dynamic Vehicle Routing Problem (abbreviated as the DVRP in the operations research literature).

1.2 The Challenges

1.2.1 Task Allocation and Routing Problem

The foundation and well-discussed routing problem is the Traveling Salesman Problem (TSP) as presented by [45] in Journal of Operations Research, in which a salesman is required to visit a set of costumers and return to the point he started in, which is known as depot. The goal for the TSP is to minimize the total distance traveled by the salesman. The Vehicle Routing Problem (VRP) is a development of the TSP which consists in defining N-number vehicle routes, where the route is a tour that launchs at the starting point named depot, visit a set of customers, and returns to the depot. All customers must be visited exactly once and only by a single vehicle. Also, the total customers demands in a route should not exceed the vehicle capacity. The goal of the VRP is to minimize the total implementation costs.

In reality, the application of VRP meets number of additional constraints that complicate the model. These additional constraints could, for instance, be time constraints on the total route time and time windows within which the service must begin and terminate. Furthermore, another constraints that complicates the further models are applied in several studies such as adding multiple depots and commodities. There are many methods to solve VRP, including exact methods such as mathematical programming, and also custom designed heuristics and meta heuristics such as Saving Algorithm and Genetic Algorithm have also been applied to

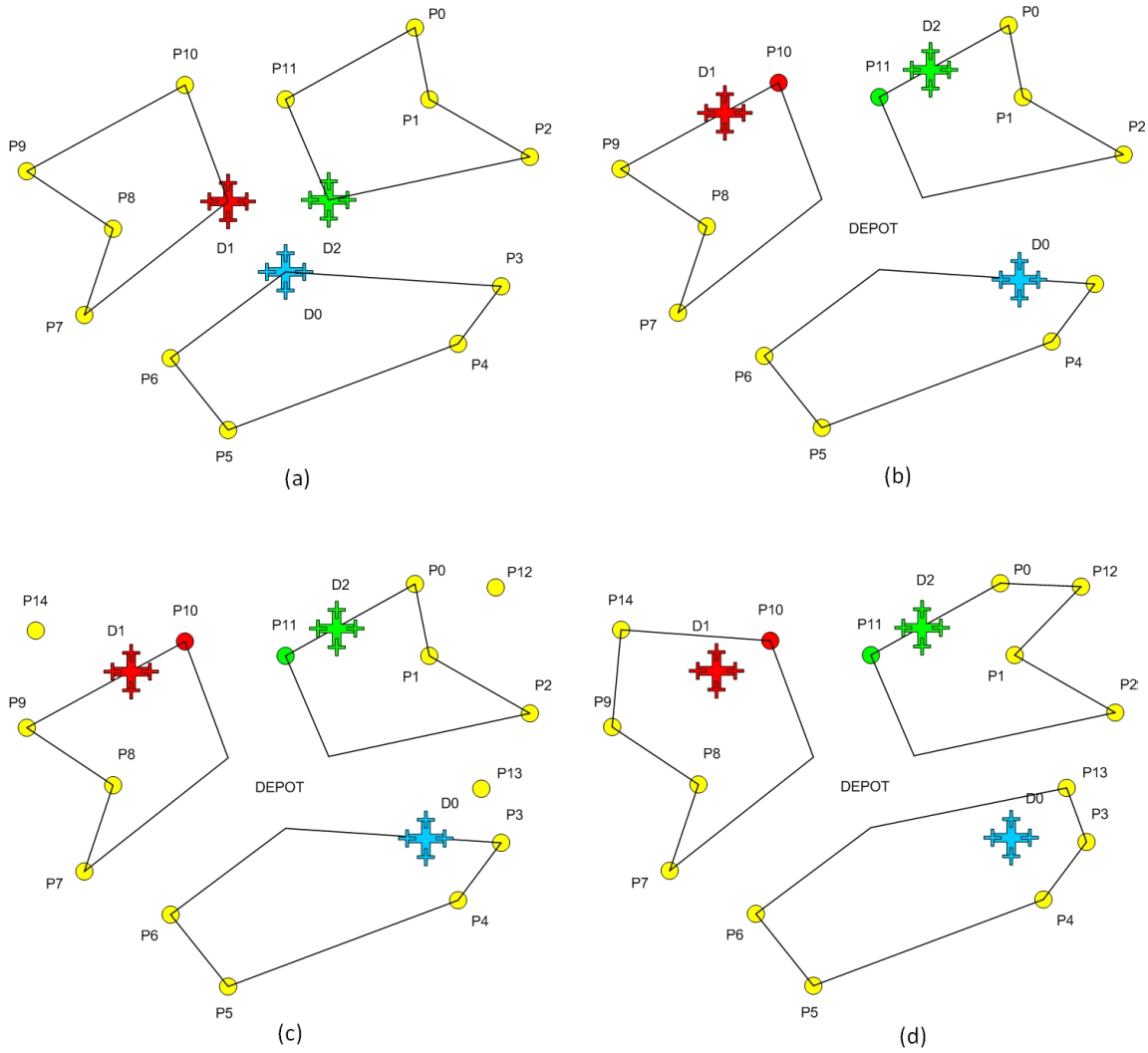

FIGURE 1.2 – The Proposed Scenario

the VRP. Several researchers has provided a survey on VRP such as [34] [41] [42] [51] and [71].

Reference [42] provides examples of VRP and several issues that emerge in its application, and also presents the examples of simple heuristics, math-programming based heuristics, optimization algorithms and various heuristic approaches. Reference [34] reviews various types in VRP and TSP such as inserting time windows, or complicates the problem with pick-up and delivery problems. Reference [71] exposes various exact methods and approximation methods, including heuristics meta-heuristics, to solve the VRP and its variants. The meta-heuristics methods that it exposed are the Ant Colony Optimization and Genetic Algorithm. Reference [41] presents a review some exacts, heuristics and metaheuristics methods for the

Vehicle Routing Problem with Time Windows (VRPTW). Reference [41] also reviews several approximation methods such as Simulated Annealing, Genetic Algorithm and Tabu Search. Also, reference [51] provides a categorized bibliography of metaheuristics for solving VRP ans its extensions. The categories are based on various types of metaheuristics like Ant Colony, Genetic Algorithm, Greedy Randomized Adaptive Search Procedure, Simulated Annealing, Tabu Search, Variable Neighborhood Search, and hybrid methods.

This thesis applies the VRP to deal with the task allocation and routing algorithms for a fleet of drones in accomplishing the given mission. The mission mentioned before is not just take into account the static scenaios, but also the dynamic one.

1.2.2 Dynamic Vehicle Routing

Mainly, most of the available literature on routing for drones focuses on static environments and does not properly account for dynamic scenarios. In several decades, Vehicle Routing Problem (VRP) has been studied in a very large scale, and most of them focused on the static cases of vehicle routing in which at the time of the planning of the routes, all information is known.

But recently, dynamic scenarios have become more familiar in transportation models, and are seem to become even more in the future. In vehicle routing, they extend a various applications, such as the delivery of petroleum products or industrial gases [19] [37] [43], courier services [79], intermodal services [29] [85], tramp ship operations [48] [102] [112], pickup and delivery services [84] [117], management of container terminals [4], production floor robots [105], etc.

In the last decade, the number of publication which discuss the dynamic models for vehicle routing has been growing. Psaraftis [95] studies the fundamental topics in this field and contributes a survey for various dynamic vehicle routing problems (DVRP). In [95], he remarks that just a few results for some simple variants of the DVRP have been retrieved. This indicates the difficulties in obtaining other results for more advanced DVRP.

According to [93], the first reference to a dynamic vehicle routing problem is due to Wilson and Colvin [124] . They studied a single vehicle DARP, in which customer requests are trips from an origin to a destination that appear dynamically. Their approach uses insertion heuristics able to perform well with low computational effort. Later, [96] introduced the concept of immediate request : a customer requesting service always wants to be serviced as early as possible, requiring immediate replanning of the current vehicle route.

As stated in [95], a VRP termed as a "dynamic" (or sometimes named as "real-time", or "on-line" by several published papers) if the information on the problem is synchronously updated with the execution of the mission. Otherwise, if all inputs are informed before the execution of the mission and do not change there after, the problem is named as "static".

Larsen [75] made definition that differs the static VRP with the dynamic one. In the static

VRP :

1. All relevant informations relevant to the planning of the routes are assumed to be known by the planner before the routing process begins.
2. Information relevant to the routing does not change after the routes have been constructed.

While, in the dynamic VRP :

1. All relevant informations to the planning of the routes are known by the planner when the routing process begins.
2. Information can change after the initial routes have been constructed.

Obviously, the dynamic VRP is more complex compared to the static VRP. The dynamic VRP demands for real-time algorithms that run right away since the immediate requests should be handled. As static VRP are NP hard, it is not always possible to find optimal solutions to the problems. This indicates that the dynamic VRP is also classed as an NP hard problems, since a static VRP should be solved each time a new immediate request is received.

In Figure 1.3 a simple example of a dynamic vehicle routing scenario is shown. In the example, three vehicles must visit both advance and actual request points. The advance request points are illustrated by yellow nodes, while those that are actual requests are illustrated by black nodes. The solid lines show the three routes that has been planned prior to the drones leaving the depot. The three drones indicate their positions at the time the actual requests are received. Then, the new points should be inserted into the routes, and this insertion demand a path-reconstruction. The dashed lines indicate the cancelled path due to a path-reconstruction. However, in practice, the insertion of new points will be a complex task and will imply a re-planning of the non-achieved targets in the system.

1.3 Objectives and Contributions

To deal with the challenges we defined in the previous section, we propose these contributions :

1. **Approximation Algorithms for 3-Dimensional Vehicle Routing Problem for a Fleet of Drones.**

   We implement vehicle routing problem (VRP) to address the challenge of task allocating and routing for a fleet of drones which doing a surveillance mission above a certain terrain. The fleet must perform route to visit a set of points while respecting constraints. We apply an approximation method to solve VRP since it is classified as an NP-hard optimization problem. Among the existing approximation methods, we use Genetic Algorithm (GA). GA is one of the most used algorithm in solving VRP. This topic has

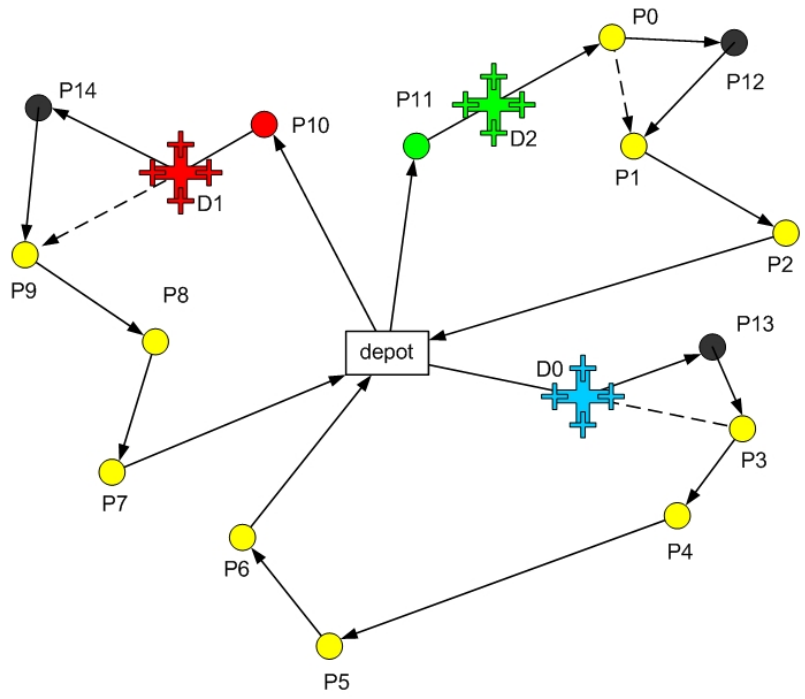

FIGURE 1.3 – A dynamic vehicle routing scenario with 12 known points and 3 new emerge points. Solid lines show the arranged routes. Dashed lines show the deleted routes due to the emerge of the new points.

already been intensively researched by a lot of papers in decades. At least, thousands of paper with this topic are published yearly.

To boost-up the performance of GA, we consider to find a better initial population constructor. We don't want to generate the initial population in a random manner as usual. So, we decide to implement Saving Algorithm (SA) in constructing the initial population of GA.

Hybridizing two or more algorithms is not a new concept in this field of study. A hybrid of GA and SA is also done by several research. Most of them implement the SA in order to generate new generation in GA, but we implement SA in order to generate initial population, so that the initial population generation no longer done in a random manner.

Hybridizing GA and SA is perfect for each other. GA can play a role as improvement phase heuristic for SA, since SA is a construction heuristic. For this reason, SA could be the perfect initial population constructor for GA, since one of the basic items that need to be carefully considered for GA to work as effective as possible is : a good initial population constructor.

As the tool for experiment and simulation as well, we use Netlogo. By using Netlogo,

we can create an interface window as the control panel of the algorithm, which consist of interfaces to control the parameters of experimental and simulation.

The experimental result shows that GA can give us a satisfactory result for finding the optimal flyable route in the 3D environment. But later, we will find that it's hybrid with SA give a very encouraging result in this project.

2. **Reverse Open Vehicle Routing Problem in a Dynamic Scenario of a Fleet of Drones.**

   The core of this contribution is the Dynamic Vehicle Routing Problem (DVRP). During the decades, number of papers have issued in many journals and books discussing DVRP. The most common source of dynamism in vehicle routing is the online arrival of customer requests during the operation. More specifically, requests can be a demand for goods, services and travel time. DVRP force making decisions in an online manner, which often compromises reactiveness with decision quality. In other words, the time invested for searching better decisions, comes at the price of a lower reactiveness to input changes. This aspect requires a good decision, at the time when new requests appear, to be made as fast as possible.

   We implement Reverse Open Vehicle Routing Problem (ROVRP) as a technique to deal with the dynamic scenario which could occurs during the mission execution. According to our knowledge, the ROVRP has not been much discussed. As named, ROVRP is the reverse version of open vehicle routing problem (OVRP), which, the difference with the classic VRP is that the vehicles are not demanded to return to the depot. That is why we call it open. ROVRP is applied to define a set of to-depot returning vehicle routes from any position of the vehicles, in constructing a new paths due to a dynamic scenario in a fleet of drones. Since VRP is classified as an NP-hard optimization problem, then ROVRP is also classified as the same. For that reason, we propose an heuristic algorithm to find a solution to this combinatorial optimization problem.

   We divide our ROVRP into two phases : clustering and routing. In clustering phase, we simply determine it by seeking the nearest neighbor of each new points emerge. The new points emerge will be joined into their nearest neighbor's cluster. In routing phase, we implement the Saving Algorithm. This phase is divided into 2 steps : creating saving list and constructing routes.

   We implement this ROVRP because it works fast. The clustering can be done fast, and the routing is done by the fastest heuristic, SA. Dynamic VRP force making decisions in an online manner, which often compromises reactiveness with decision quality. In other words, the time invested searching for better decisions, comes at the price of a lower reactiveness to input changes. This aspect requires a good decision, at the time when new requests appear, to be made as fast as possible.

   For the experiment and simulation as well, we also use Netlogo as the tool.

1.4 Scientific Production

1. **S. A. Sukarno,** R. Ben Atitallah and M. Djemai, "Approximation Algorithm for 3-Dimensional Vehicle Routing Problem for Fleet of Multi-Agents," 6th International Conference on Control Engineering & Information Technology, Istanbul, Turkey, 2018.
2. **S. A. Sukarno,** R. B. Atitallah and M. Djemai, "Approximation Methods to Vehicle Routing Problem for a Drone Fleet Management," Transactions of the Institute of Measurement and Control. (Under Review)

1.5 Organization of the Thesis

This thesis is organized as follows :

— Chapter 1 starts with general introductions and the challenges of research to the problem that we studied in this thesis. In the general introduction, we present the context of our works in this thesis. The task allocation and routing problem, and also the dynamic vehicle routing problem are delivered as the challenges in this works. Afterwards, this chapter also provides the objectives and the contributions of this thesis, and the scientific productions that we have already produced.

— Chapter 2 presents the backgrounds and the related works to this thesis. Chapter 2 reviews the literature on vehicle routing problem (VRP) and the existing methods in solving VRP. The review includes the classification of methodologies in solving combinatorial optimization problems. In particular, literatures on classic heuristic, metaheuristic, and the hybrid algorithm are discussed for a deeper understanding of the problem determined in this thesis. In this chapter, we also discuss the dynamic VRP (DVRP) and another version of VRP called reverse open VRP (ROVRP), to deal with a dynamic scenario in a fleet of drones.

— Chapter 3 begins the discussion on the challenges in organizing a fleet of drones. Vehicle routing problem (VRP) is stated as a perfect study to answer this challenges. Among the existing methodologies to solve VRP, we choose approximation methods since VRP is an NP-hard problem. Among several algorithms in approximation methods, genetic alorithm (GA) is one of the most used and famous algorithm in solving VRP. Thousand of papers each year discuss GA to solve VRP. So, in this chapter we present the application of GA in solving the 3-dimensional VRP for a fleet of drones. In the second part of this chapter, we decide to add another algorithm to improve the performance of the previous proposition. We finally decide to hybridize GA with saving algorithm (SA), since SA is classified as the construction heuristic, and has a perfect score in speed and simplicity. Those attributes are perfect to be hybridized with GA. To meet the needs in experimental purpose, we've searched an experimental tool that can be programmed to

model a complex system, and also simulate and observe events in 3D environment, where user can give instructions at many agents to operate independently and concurrently. For that reason, we chose Netlogo. NetLogo is the next generation of the series of multi-agent modeling languages. It enables users to open simulations and "play" with them, exploring their behavior under various conditions. It is also an authoring environment that is simple enough to enable students and researchers to create their own models, even if they are not professional programmers. That makes Netlogo becomes suitable in education and research, and it is also used by many researchers worldwide. With Netlogo, we can create an interface window as the control panel of the algorithm, which consist of interfaces to control the parameters of experimental and simulation, buttons to control the process, and also monitors to see the solution visually.

— Chapter 4 proposes a dynamic scenario that might happen in the mission given. Many literatures denote this problem as a dynamic vehicle routing problem (DVRP). We use the reverse open vehicle routing problem (ROVRP) approach to deal with the dynamic scenario in the mission of a fleet of drones. To solve the ROVRP, we devide the implementation into two phases : clustering and routing. In the clustering phase, we simply search the nearest neighbor for each new points emerge to be clustered in to. Also, to create the new routes according to the emerge of one or more new points, we apply the saving algorithm (SA). The routing phase is divided into two steps : creating saving list and constructing routes. As implemented in chapter 3, in this chapter we also use Netlogo as the tool for programming and simulating.

— Finally, conclusions with some proposed future works are presented in Chapter 5. Here, we summarize the main contributions achieved in this work, and we will put our work in perspective with suggestions for future works. The growth and advancements in the studies and researches on managing a fleet of drones has lead the emerged of this thesis.

Chapter 2

Background and Related Works

2.1 Vehicle Routing Problem

In this chapter, we will review the discussion on Vehicle Routing Problem (VRP), the open version of it and also it's dynamic version. Then, we will introduce the heuristic and metaheuristic methods in designing an algorithm for the fleet of drones to accomplish the mission given.

Since Dantzig and Ramser [31] introduced it in modeling how a fleet of homogeneous trucks could serve the demand for oil of a number of gas stations from a central hub and with a minimum traveled distance in 1959, VRP became one of the most analyzed problem in the fields of transportation, distribution and logistics, and one of the most widely studied topics in the field of operation research [17]. It requests a solution of the optimal set of routes to be performed by a fleet of vehicles to serve a set of customers [119], subject to a set of constraints [71] [73]. The main objective is to minimize the cost in serving all costumers. It generalizes the well-known traveling salesman problem (TSP).

Although VRP is a very important combinatorial optimization problem, it is categorized as an NP-hard problem, thus it is hard to find the global solution in a large scale problem in VRP. In decades, so many extensive studies in finding algorithms to solve VRP. All of them could be divided into two categories : exact methods and approximation methods. The exact algorithms can be classified into three classifications : (1) direct tree search method ; (2) dynamic programming ; and (3) integer linear programming. Since VRP is categorized as an NP-hard problem, the exact algorithms can only solve the VRP with a small-scale. In [73] six representative examples of exact algorithms for VRP are provided, including two direct tree search methods based on different relaxations, a dynamic programming formulation and three integer programming algorithms.

The most effective exact algorithm for VRP is branch-and-cut (BC) algorithm based on a two-commodity network flow formulation of the problem [9]. Reference [47] proposed the method of a new branch-and cut-and-price (BCP) algorithm based on the two-index and the set partitioning (SP) formulations. The lower bound is computed with a column-and-cut generation method that uses k-cycle-free q-routes instead of feasible CVRP routes and the valid inequalities. The first exact algorithm for the VRPTW based on the SP formulation was the branch-and-price (BP) algorithm from reference [33]. In general, any exact algorithm for the VRPTW based on the SP model can be easily adapted to solve the CVRP by simply relaxing the time window constraints in the pricing algorithm.

2.2 Saving Heuristic for Vehicle Routing Problem

Reference [56] indicates that for VRP with more than 50 costumer, exact algorithm do not agree with the optimal solution. This circumstances led so many researches to design several algorithms with approximation methods. The approximation methods are divided into two

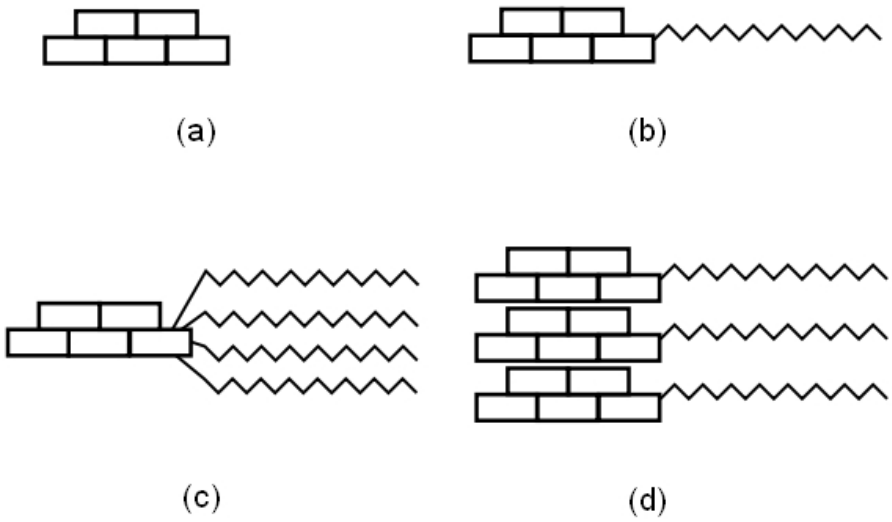

FIGURE 2.1 – Graphical representation of several construction and improvement heuristics. Bricks represent the construction phase. Ripples represent the improvement phase. Source : [27]

groups : classical heuristic and metaheuristics.

Classical heuristics for the VRP are naturally divided into constructive heuristics and improvement heuristics. Laporte and Semet [74] documents and reviews these two types of heuristic. The descent heuristics always proceed from a solution to a better one in its neighbourhood until no further gain is possible. In contrast, metaheuristics allow the consideration of non-improving and even infeasible intermediate solutions.

Construction heuristics mainly includes Clarke and Wright's savings algorithm [26], route-first cluster-second [10], cluster-first route-second [44] and sweep algorithm [52] [106]. Two types of improvement algorithms can be applied to VRP solutions : intra-route heuristics and inter-route heuristics. Mostly, constructive procedures are followed by an improvement phase [27].

Fig. 2.1 illustrates a number of ways to design heuristics consisting of a construction phase followed by an improvement phase. Fig. 2.1(a) shows the constructive heuristic. Fig. 2.1(b) shows single construction improvement thread. The constructive phase followed by improvement in several ways (may be executed in parallel) is shown in fig. 2.1(c), and fig. 2.1(d) demonstrates several construction improvement threads (may be executed in parallel).

Savings heuristic was proposed by [26] to solve the problem for which the number of vehicles is a decision variable. It is perhaps the most widely known heuristic for the VRP.

This method is started by calculates saving amount for every pair of points. Those saving

amounts are then ranked from the biggest to the smallest. Starting from the biggest amount, the paired points are linked and added to the tour, and the other points are placed on routes into which they can be linked (otherwise a new route is started) until a constraint is reached.

Later, this algorithm is implemented in this thesis. It is easy to implement but is suitable mainly to the small scale optimization. According to [78], this algorithm does not work well when the scale of costumer increases. That is why, in the implementation on chapter 3, we hybridize it with an algorithm that can play the improvement role, since we know that saving algorithm is a constructive heuristic. Another constructive heuristics is listed in annex of this thesis.

2.3 Genetic Algorithm for Vehicle Routing Problem

In the application to combinatorial optimization problems, metaheuristics is a field of research that is growing rapidly in decades. Reference [14] surveys the metaheuristic for combinatorial optimization problem. [14] gives a definition of metaheuristic : In computer science and mathematical optimization, a metaheuristic is a higher level procedure or heuristic designed to find, generate or select a heuristic that may provide a sufficiently good solution to an optimization problem, especially with incomplete or imperfect information of limited computation capacity. Recently, this methods are rising as successful alternatives to more classical approaches also for solving optimization problems that include in their mathematical formulation uncertain, stochastic, and dynamic information.

Blum and Roli [16] lists the properties that characterize most metaheuristics : (1) Metaheuristics are strategies that guide the search process. (2) The goal is to efficiently explore the search space in order to find near optimal solutions. (3) Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes. (4) Metaheuristic algorithms are approximate and usually non-deterministic. (5) Metaheuristics are not problem-specific.

There are a wide variety of metaheuristics. Some properties are used to classify them, such as local search or global search, single-solution or population-based, hybridization or memetic algorithms, parallel metaheuristcs and mature-inspired metaheuristcs. The most common and studied metaheuristics include Ant Colony Optimization, Simulated Annealing, Tabu Search and Evolutionary Algorithm (AE). Genetic algorithm, which later is applied in this thesis, evolutionary programming and memetic algorithm belong to EA. The classification and main metaheuristics in each classification is showed in figure 2.2.

Furthermore, a good metaheuristic implementation can provide near-optimal solutions in reasonable computation times. VRP, in particular, exhibits an impressive record of successful metaheuristic implementations [51].

The most popular types of metaheuristics, according to [51], are explained in the annex of

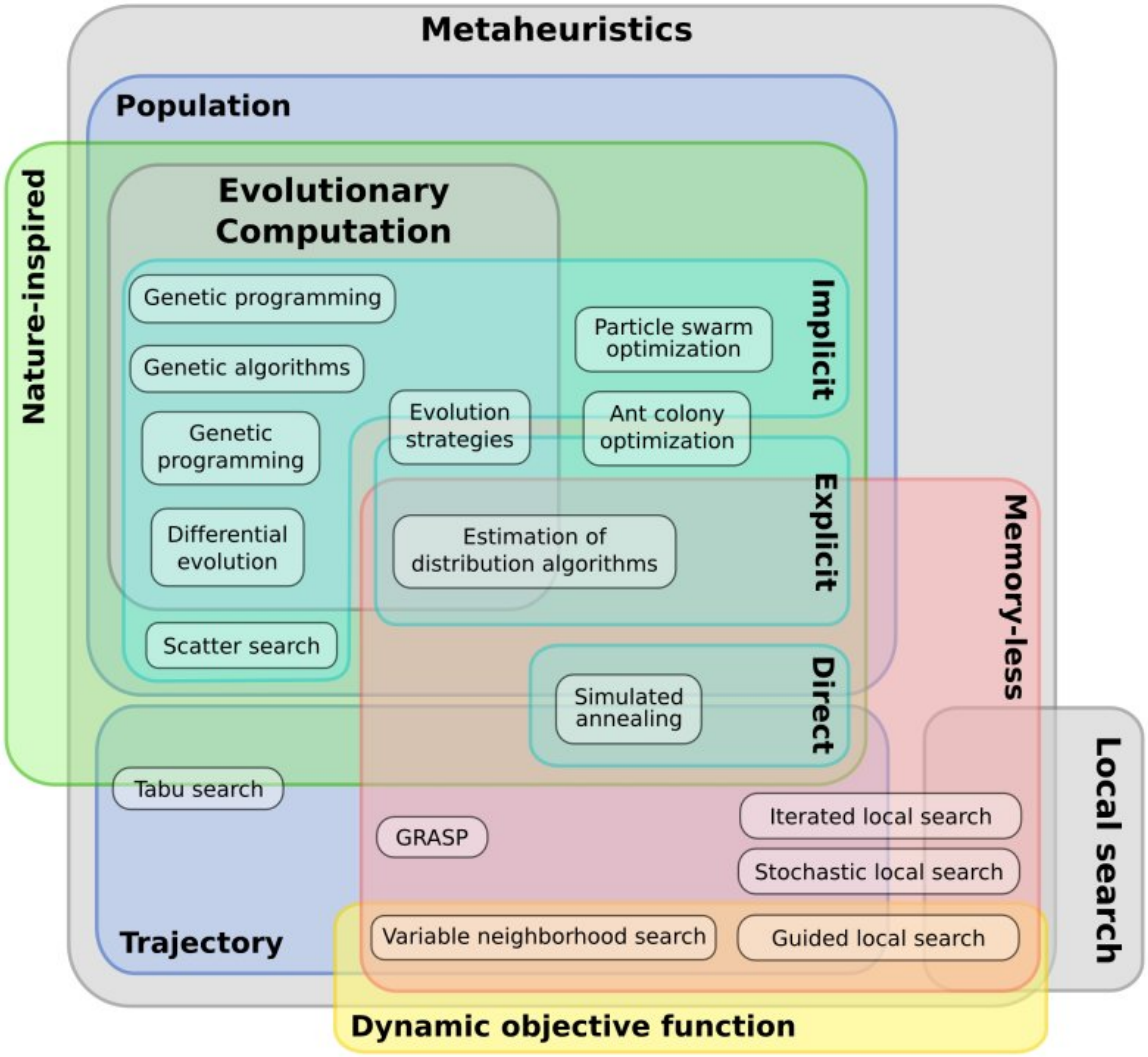

FIGURE 2.2 – Classification of Metaheuristics. Source : [36]

this thesis. One of them is evolutionary algorithm. Evolutionary algorithms are a large class of metaheuristics, also inspired from a natural metaphor, with Genetic Algorithms (GAs) [55] being one of the best known. Basically, they imitate the technique species evolve and adapt to their environment, according to the Darwinian principle of natural selection. Under this paradigm, a population of solutions (often encoded as bit or integer strings, known as chromosomes) evolves from one generation to the next through the application of operators that are similar to those found in nature, like selection of the fittest, genetic crossover and mutation. Through the selection process, only the best solutions are allowed to become

parents and to generate offspring. The mating process, known as crossover, then takes two selected parent solutions and combines their most desirable features to create one or two offspring solutions. This is repeated until a new population of offspring is obtained. Finally each offspring is randomly perturbed by a mutation operator. Starting from a randomly or heuristically generated initial population, this cycle is repeated for a number of generations, and the best solution found is returned at the end. When applied to vehicle routing problems, the classical GA solution scheme is often modified. In particular, the encoding of solutions into chromosomes is either completely ignored (by applying the various operators directly on the solutions) or designed in a very particular way to take advantage of specialized crossover and mutation operators. Fig. 2.3 shows the brief diagram of the GA's procedure in searching for a solution in a combinatorial optimization problems.

Among the Metaheuristic Algorithms, Genetic Algorithm (GA) is one of the most used in various types of transportation problems, such as facility location problem (FLP) and vehicle routing problem (VRP) [68]. There have been a lot of the researches done about solving VRP with GA such as [6] [8] [23] [65] and [127]. In [6] and [8], authors discuss the application of GA to the capacitated VRP and basic VRP in which customers of known demand are supplied from a single depot. Cheng et al. [23] shows that an ideal flight path for unmanned aerial vehicle (UAV) can be more quickly searched using Immune GA. Zhang et al. [127] introduces a finite automaton (FA) to produce individual population and proposes a new evolution way enlightened by hermaphrodites. Iwańkowicz and Sekulski [65] applies GA in planning the collection of wastes by one garbage truck from a certain number of collection points. The topic of GA in solving VRP has already been intensively researched by a lot of papers in decades, as shown in fig. 2.4

2.4 Hybrid Genetic and Saving Algorithm for Vehicle Routing Problem

The main reason of hybridizing two or more algorithms in solving the combinatorial optimization problem is to improving the result and the performance of the algorithm. We need to consider two criteria in determining which algorithm we use to solve the optimization problem : speed and accuracy [28]. Beside those main criteria, there are also another additional attributes that are also essential : simplicity dan flexiility [28]. But, the problem we frequently meet is not easy to get those criteria in a heuristic we apply. If it has speed, sometimes it is not accurate. Otherwise, if it has accuracy, it is not fast.

None of the classical heuristics fares very well on accuracy and flexibility. In this category, Saving algorithm (SA) has at least the distinct advantage of being very quick and simple to implement. SA is one of the best known and remains widely used in practice to this day. It is a heuristic with the best score in speed and simplicity as well [28]. SA contains no parameters

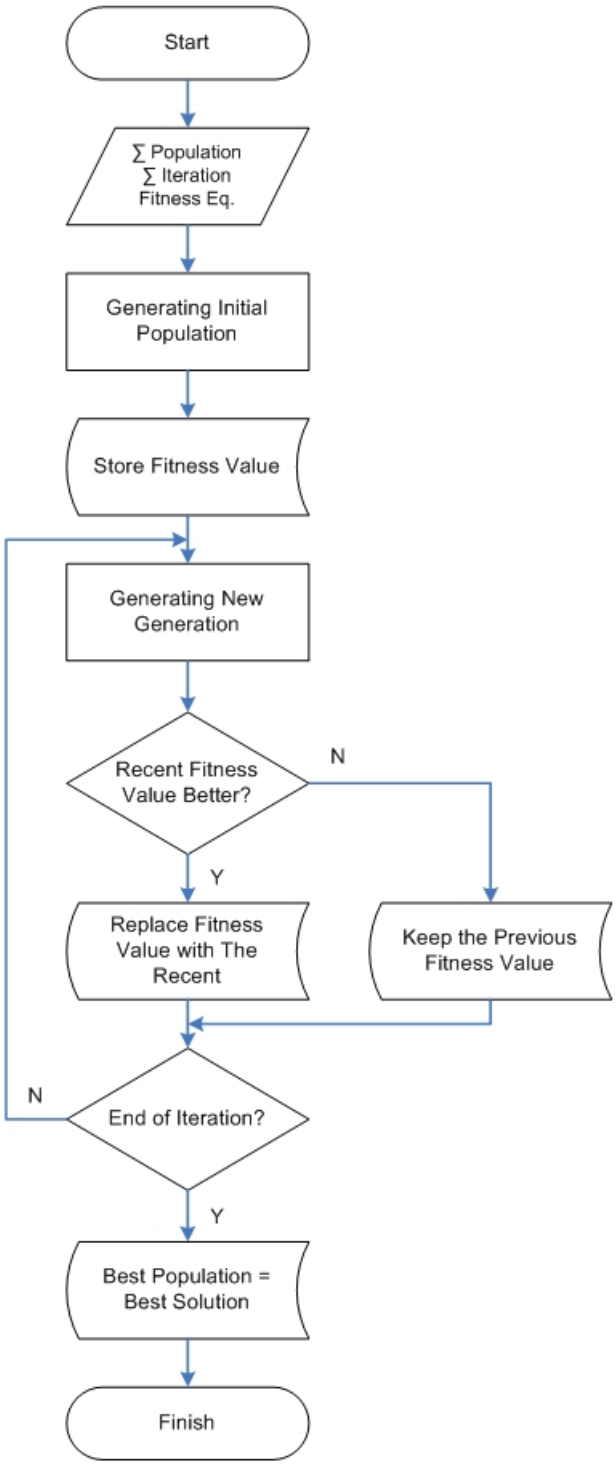

FIGURE 2.3 – Procedure of Genetic Algorithm

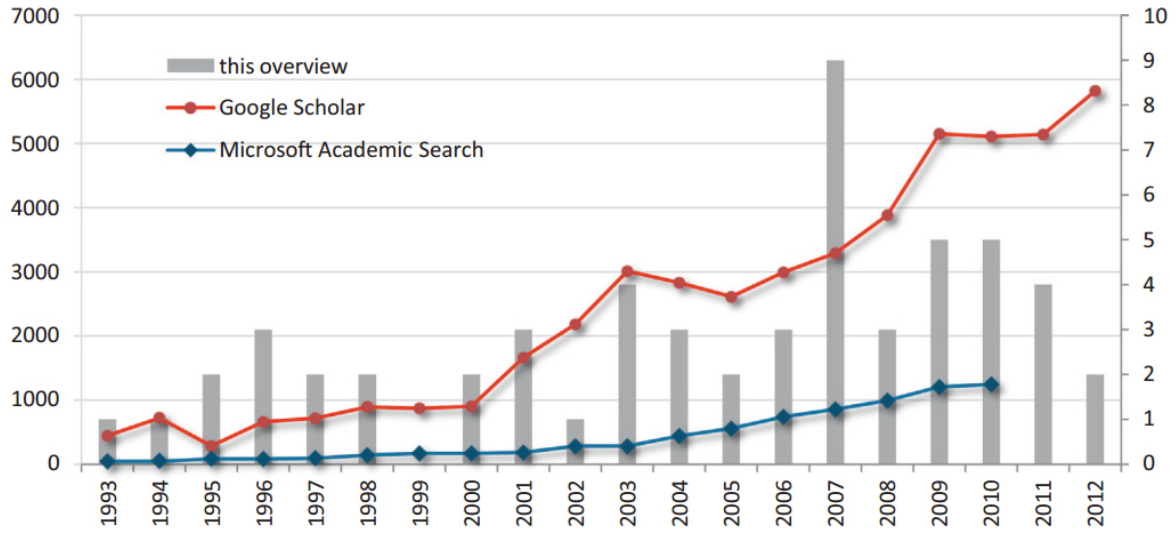

FIGURE 2.4 – The number of papers per year on using GA for solving VRP. Source : [68]

and easy to code, that is why it has a good score in simplicity. But, as stated by [78], saving algorithm does not work well when the scale of costumer increases. So, we need to apply another algorithm to improve and boost the lack from the weak attributes of SA. This is the main reason of hybridizing.

In order to boost their performance, most constructive procedures are followed by an improvement phase [27]. Compared with classical heuristics, metaheuristics perform a much more thorough search of the solution space, allowing inferior and sometimes infeasible moves, as well as recombinations of solutions to create new ones. A metaheuristic that fit and compatible to play the role as an improvement heuristics methods is Genetic Algorithm (GA) [15].

In this thesis, we combine GA with saving algorithm (SA) in solving VRP for a fleet of drones. We found few papers that combine GA with SA in VRP, for example [60] and [89]. In [60], GA is hybridized with three heuristics. [60] A hybrid genetic algorithm for the multi-depot vehicle routing problem. The three heuristics hybridized in the algorithm are SA, the nearest neighbor heuristic, and the iterated swap procedure. Paper [89] also combined GA and SA in their work, but in different way with ours. They implemented the SA in order to generate new generation, but we implement SA in order to generate initial population as shown in the diagram of fig. 3.8.

Hybridizing GA and SA is perfect for each other. GA can play a role as improvement phase heuristic for SA, since SA is a construction heuristic, as shown in fig. 3.9. And SA could be the perfect initial population constructor for GA. Since one of the basic items that needs to be carefully considered for the algorithm to work as effective as possible is : a good

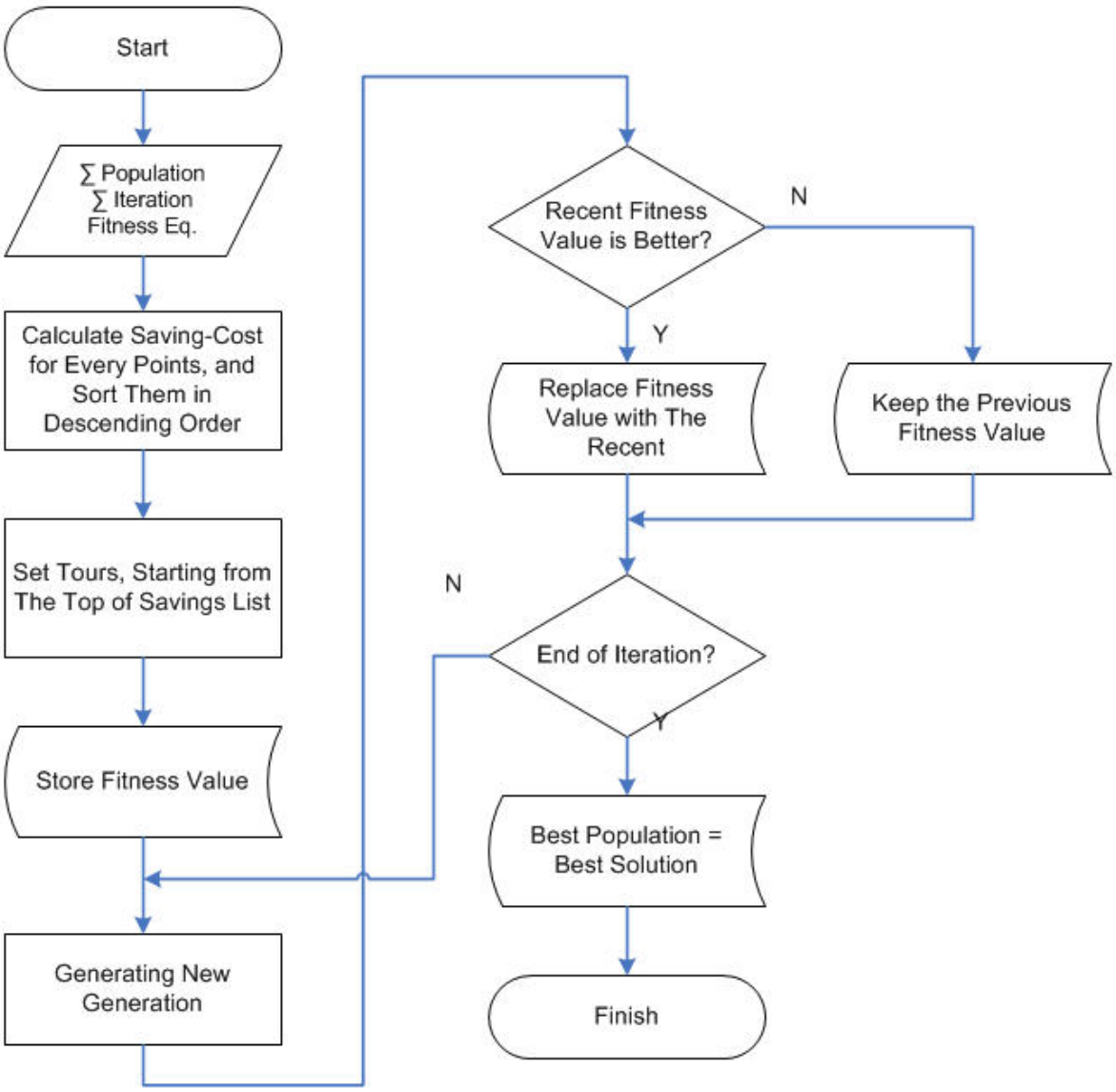

FIGURE 2.5 – Diagram of Combining Genetic Algorithm and Saving Algorithm.

initial population constructor.

2.5 Reverse Open Vehicle Routing Problem for Dynamic Scenarios in a Fleet of Drones

2.5.1 Dynamic Vehicle Routing Problem

According to Pillac et al. [93], the first scientific paper to a dynamic vehicle routing problem (DVRP) is due to Wilson and Colvin [124]. They studied a single vehicle dial-a-

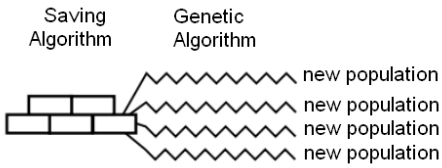

FIGURE 2.6 – Construction and Improvement Heuristic. SA as the Construction Heuristic, and GA as the Improvement Heuristic.

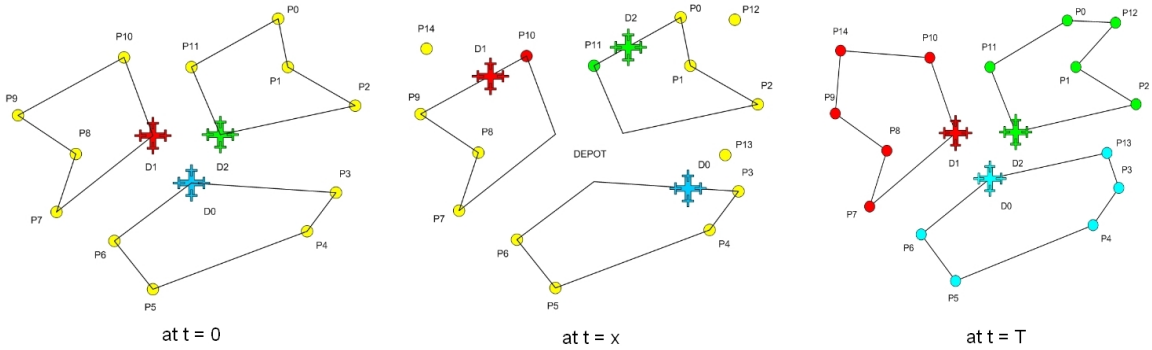

FIGURE 2.7 – Example application of DVRP

ride problem (DARP), in which customer requests are trips from an origin to a destination that appear dynamically. Their approach uses insertion heuristics able to perform well with low computational effort. Later, Psaraftis [96] proposes the concept of immediate request : a customer demanding service always wants to be serviced as early as possible, requiring immediate replanning of the current vehicle route. The differences between static or classic VRP and DVRP are explained in the annex of this thesis. The real-life application of DVRP is also explained.

In giving a better understanding to what we mean by dynamic, fig. 2.7 illustrates the route execution of a three drone in DVRP. Before the drones leave the depot at time $t = 0$, an initial route plans to visit the currently known requests. While the drones execute their routes, three new requests (P12, P13 and P14) appear at $t = x$ and the initial route is adjusted to fulfill them. Finally, at time $t = T$ the all routes are executed, and the initial route plans are changed due to the emerge of the new requests. This example expose how dynamic routing fundamentally adjusts routes in an ongoing fashion, which requires real-time communication between drones and the ground communication center.

In 2015, Schopka and Kopfer [109] implements the adaptive large neighborhood search (ALNS) in generating cost-efficient transportation plans for the reverse open vehicle routing

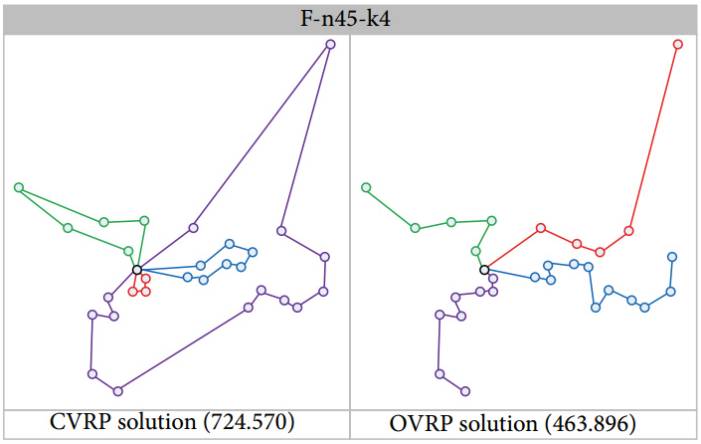

FIGURE 2.8 – The difference of best known solutions between classic VRP and OVRP. Source : [92]

problem (ROVRP). To us, ROVRP is another approach for DVRP, since it is used in updating vehicle return-trip to the depot. For that reason, we use this approach in our thesis.

2.5.2 Reverse Open Vehicle Routing Problem

Reverse open vehicle routing problem (ROVRP) is an opposite version of the Open VRP (OVRP), which has been presented before by Sariklis and Powell [108] in their paper on distribution management problems. The special characteristic which distinguishes open-VRP (OVRP) from the classic-VRP (CVRP) is that the vehicles are not demanded to return to the depot. In different case, the vehicles could return to the depot by tracking the same tour in reverse order [92]. In classic VRP each route is TSP which requires a Hamiltonian cycle, but in OVRP each route is a Hamiltonian path. The classic VRP and OVRP are both NP-Hard problem. In fig. 2.8, the best known solution for the OVRP is very different from the classic VRP.

The OVRP has been discussed in several studies, for example in a train plan model for British Rail freight services through the Channel Tunnel [46], a case study for the school bus routing problem in Hong Kong [77], the distribution of fresh meat in Greece [115], the distribution of a daily newspaper in the USA [104] [24], a lubricant distribution problem in Greece [100], and a mines material transport vehicle routing optimization in China [125]. Whereas, according to our knowledge, the ROVRP has not been much discussed. We found a work from Schopka and Kopfer [109] who modified the Adaptive Large Neighborhood Search (ALNS) for generating ROVRP in updating vehicle return-trip to the depot. Also, Park and Kim [90] and Li and Fu [77] who reviewed and made a case study of the School Bus Routing Problem.

We apply ROVRP to define a set of to-depot returning vehicle routes from any position

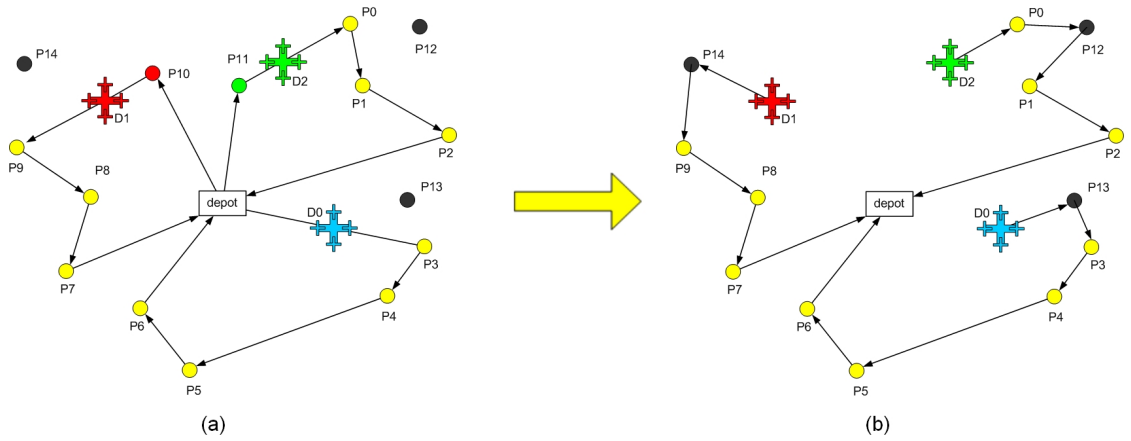

FIGURE 2.9 – The application of ROVRP. (a) At time $t = x$ when three new requests (P12, P13 and P14) appear during mission execution. (b) The result of ROVRP in adjusting the routes.

but depot, in constructing a new paths due to a dynamic scenario in a fleet of drones as shown in fig. 2.7. At the time $t = x$, when the drones execute their routes, three new requests (P12, P13 and P14) appear. At the time when a new request appears, the ROVRP is applied to adjust the initial routes in real-time so that we get the result as shown in fig. 2.9. This method is divided into 2 phases i.e. clustering and routing.

2.5.2.1 Partitioning and Clustering

In partitioning and clustering, we simply determine it by seeking the nearest neighbor of each new points emerge. For example each new points emerge in fig. 2.9, the nearest neighbor for the points P12, P13 and P14 are points P0, P3 and P9 respectively. So, we insert the points P12, P13 and P14 to the cluster set of points {P11, P0, P1, P2}, {P3, P4, P5, P6} and {P10, P9, P8, P7} respectively. But, since the points P10 and P11 is already visited (marked by the same color with the drones who visited them) by one of the drones who is carrying out a mission, those points are not inserted to the new cluster, thus the new cluster to be taken into account are {P12, P0, P1, P2}, {P13, P3, P4, P5, P6} and {P14, P9, P8, P7}.

2.5.2.2 Routing

The algorithm we implement in this step is the saving algorithm (SA). This steps is divided into 2 parts : creating saving list and constructing routes.

The saving list is only made for each cluster of points. It means, if $S_{i,j}$ is the saving value between point-i and point-j, we should ensure that point-i and point-j are in the same cluster.

The savings value between point i and j is calculated as :

$$S_{i,j} = c_{i,P} + c_{P,j} - c_{i,j}$$

where $c_{i,P}$ is the traveling distance between the actual position of the drone (P) and point-i, $c_{P,j}$ is the traveling distance between the actual position of the drone (P) and point-j, and $c_{i,j}$ is the traveling distance between two points i and j. After calculation, all savings values are collected in the savings list, and the values in the savings list are sorted in decreasing order.

The route constructing procedure starts from the top of the savings list (the biggest value of $S_{i,j}$). Both point i and j will be merged into the same route if the total flight time does not exceed the drone's maximum flight duration. Since ROVRP is a to-depot returning vehicle route, the route will be defined as an ordered sequence of points which starts at the actual position of the drone (P), visits all its points once and only once, and terminates at the depot (0). The cost of a route is defined as the sum of the traveling costs through its cluster members. In this phase we aim to find the least cost route in each cluster.

2.6 Positioning

In this section, we will position our contributions with respect to the described state-of-the-art. Classical heuristic algorithms and metaheuristic algorithms are widely implemented to solve various combinatorial optimization problem in decades, especially the vehicle routing problem (VRP). The existence of those algorithms are very important since the exact algorithms can only solve the small scale opotimization problem. Throughout the chapter, we reviewed several algortihm to solve VRP. In recent decades, hybridizing two or more algorithms become a widely used methods since none of the heuristics works very well on accuracy and flexibility.

We choose to hybridize genetic algorithm (GA) and saving algorithm (SA) in solving VRP for the following reasons :

— GA was among the most widely used metaheuristic for solving VRP. For more than two decades, in average, thousands of papers are issued per year providing GA for solving VRP.

— GA still opens opportunities for improving the algorithm by considering several basic items such as : (1) A good genetic representation of a solution in a form of a chromosome, (2) An initial population constructor, (3) An evaluation function to determine the fitness value for each solution, (4) Genetic operators, simulating reproduction and mutation, (5) Values for parameters ; population size, probability of using operators, etc. That's why, hybridize GA with another algorithm is one of solutions that could be considered.

— SA is the best heuristic in speed and simplicity. SA is easy to implement but is suitable mainly to the small scale optimization. It does not work well when the scale of problem increases. This opens an opportunity to hybridize it with an algorithm that can play the improvement role, since we know that SA is a constructive heuristic. In order to boost their performance, most constructive heuristic are followed by an improvement methods.

— Hybridizing GA and SA is perfect for each other. GA can play a role as improvement phase heuristic for SA, since SA is a construction heuristic. SA could be the perfect initial population constructor for GA. Since one of the basic items that need to be carefully considered for the algorithm to work as effective as possible is : a good initial population constructor.

Reverse Open VRP (ROVRP) is the opposite version of open VRP. ROVRP occurs in dynamic VRP (DVRP), when return trips to a central depot have to be generated, due to a planning updates caused by the occurrence of a dynamic scenario. We apply ROVRP to define a set of non-depot returning vehicle routes, in constructing a new paths due to a dynamic scenario in a fleet of drones. In recent decade, DVRP is discussed by number of papers issued in journals and books, but research attention on the ROVRP is still limited. We perform the ROVRP in two phases : clustering and routing. We simply implement nearest neighbor for clustering, and SA for routing, for the sake of speed and simplicity, so that no much time wasted to calculate the routing updates.

2.7 Conclusion

In this chapter, we have reviewed several algorithms for solving the combinatorial optimization problem, in particular vehicle routing problem (VRP). Since the exact algorithms can only solve the small scale VRP, the heuristic and metaheuristic methods are widely applied in a lot of studies. But, condidering that none of the classical heuristics works very well on accuracy and flexibility, hybridizing two or more heuristic and metaheuristic algorithms become a widely used methods in decades. In this thesis, we hybridize Genetic Algorithm (GA) and Saving Algorithm (SA) for solving the VRP. GA is one of the most used algorithm in solving VRP. To improve the performance of the GA, we consider to find a good initial population constructor. For that reason, we choosed SA, a heuristic with the best score in speed and simplicity. It is a perfect method to play the role as initial population constructor. Hybridizing GA and SA is perfect for each other. GA can play a role as improvement phase heuristic for SA. Also, SA could be the perfect initial population constructor for GA.

To tackle the dynamic challenge in VRP, we applied the opposite version of Open VRP (OVRP) named reverse open vehicle routing problem (ROVRP). Research attention on the ROVRP is limited but we believe this approach is suitable to answer this problem. We applied

ROVRP with tho phases : clustering and routing. In partitioning and clustering, we simply determine it by seeking the nearest neighbor of each new points emerge. Then we insert each new points emerge to its neighbors cluster. For routing phase, we implement the SA, since SA is very fast and we need to make a good decision as fast as possible.

Chapter 3

Approximation Algorithm for 3-Dimensional Vehicle Routing Problem for a Fleet of Drones

3.1 Introduction

Nowadays, drone plays a big role in civilian purposes, and it will be getting bigger and more important in the future. Because of its flexibility and versatility, the application of drone is more extensive, covering surveillance, intelligent logistics, search and rescue, scientific studies, etc.

Recently, new trends are moving towards to manage a fleet of drones which collaborate in order to achieve the given mission. This issue opens many research opportunities, and our project is made to answer the challenge, to develop a platform for managing a fleet of drones. As being researched in decades, Vehicle Routing Problem (VRP) is a perfect study to answer this challenge, in order to find the best route for each drone, with several constraints to be considered.

In decades, many researchers had extensively studied this field of study as surveyed in [71] [73]. Dantzig and Ramser [31] were the first to address VRP in the optimization literature by the appearance of their paper in the journal Management Science, concerning the routing of a fleet of gasoline delivery truck between a bulk terminal and a number of service station supplied by the terminal. The problem formulated in [31] was named 'truck dispatching problem', but several literatures on the VRP cited their paper as the first example of this problem.

According to [71], there are many methods to solve VRP, and could be categorized into two groups i.e. : exact and approximation methods. But, since VRP is classified as an NP-hard optimization problem, and the exact methods do not agree with the optimal solution for VRP with more than 50 clients [56], we decided to choose an approximation method for this project.

In chapter 2, we have already reviewed several approximation methods that widely used in solving VRP. Among several algorithms in approximation methods, we prefer to apply Genetic Algorithm (GA) in this project. It is an approximation method which designed by an inspiration to the evolutionary ideas of genetic and natural selection [55]. This algorithm relies on bio-evolution procedures such as crossover, mutation and selection, to find the better generation than the previous one.

Arostegui et al [3] and Youssef et al [126] made comparison between GA and another metaheuristic approaches, and the results show that the main advantage of GA in comparison to other metaheuristic algorithms is the performance and final result on time constraints and limited computer power, while still resulting in competitive solutions. Although some other metaheuristics are able to find better solutions than GA, GA can generally find adequate solutions in a shorter time frame [3] [126]. This is also the main reason that GA are still used in solving the routing, locating and other NP hard problems.

In the field of VRP, there are six famous metaheuristics that have been applied extensively to solving the problem [15] : Simulated Annealing (SiA), Deterministic Annealing (DA), Tabu

Search (TS), Ant Systems (AS), Neural Networks (NN) and Genetic Algorithms (GA). The algorithms SiA, DA and TS move from one solution to another one in the neighbourhood until a stopping criterion is satisfied. The fourth method, AS, is a constructive mechanism creating several solutions in each iteration based on information from previous generations. NN is a learning method, where a set of weights is gradually adjusted until a satisfactory solution is reached. Finally, GA maintain a population of good solutions that are recombined to produce new solutions.

But all GA must have the following basic items that need to be carefully considered for the algorithm to work as effective as possible [83] :

1. A good genetic representation of a solution in a form of a chromosome,
2. An initial population constructor,
3. An evaluation function to determine the fitness value for each solution,
4. Genetic operators, simulating reproduction and mutation,
5. Values for parameters ; population size, probability of using operators, etc.

Points (1), (2) and (4) lead to an idea of hybridize GA, in order to boost the performance. Hybridize GA with another algorithm is one of solutions that could be considered.

GA is widely used on several combinatorial optimization problem [99] [111]. Reference [111] used GA to find the optimal flyable path for the UAV in a 3D environment. The main obstacle of their problem is the complex environment and coverage zones of radars, since the UAV intended to travel all control points in an optimal way while avoiding radar. [111] solved the Travelling Salesman Problem (TSP) for the purpose of reaching an optimal path. As the implementation environment, [111] preferred MATLAB to show the solution visually. Reference [99] used GA to define a shortest path for a quadrotor-type UAV to travel through target point without hitting an obstacle. Obstacle avoidance is the major factor they put in the problem.

3.2 Problem Formulation

In this section, we will discuss the problem formulation of this work. VRP usually defined as a complete undirected network $G = (N, E)$ with a set of points $N = \{0, 1, ..., n, n+1\}$, where the starting point, known as depot, denoted as point 0 and point $N + 1$, and a set of edge $E = \{(i, j)|i, j \in N\}$. Here, we also have a set of drones $V = \{1, 2, .., v\}$ which will be tasked to visit points.

The biggest concern in our VRP is determining the best route with the minimum total distance as the best solution. Using GA, we consistently evolve the solution by choosing the better chromosomes, which representing the route, in every iteration. The parameter that we use to define the better chromosomes is a function called fitness function. Designing

fitness function is an important aspect, since it is the one-and-only factor to decide which chromosomes will be chosen as the best solution. In our work, the fitness function F can be formulated as shown in equation (3.1) :

$$F = min \sum_{v=1}^{V} \sum_{i=0}^{N} \sum_{j=1}^{N+1} T_{ij} X_{ij}^{v} \tag{3.1}$$

Where T_{ij} is the distance covered, associated with set of edge E, when a drone travels from point-i to point-j. T_{ij} is a matrix of a complete graph. Then, X_{ij}^{v} is a binary variable which defined for each arc (i, j), such that equal to 1 if and only if a drone-v visit point-i and then travels directly to point-j, and equal to 0 if otherwise.

In constructing the routes, there are some conditions to be considered as below :

— Each vehicle visits a point once, and each point is visited once and only by a single vehicle.

$$\sum_{i=0;i\neq j}^{N} X_{i,j} = 1, \quad \forall j \in \{1, .., N+1\} \tag{3.2}$$

$$\sum_{j=1;j\neq i}^{N+1} X_{i,j} = 1, \quad \forall i \in \{0, .., N\} \tag{3.3}$$

— Each vehicle leaves from point where it arrived in.

$$\sum_{i=0;i\neq g}^{N} X_{i,g} - \sum_{j=1;j\neq g}^{N+1} X_{g,j} = 0, \quad \forall g \in \{1, .., N\}, \quad j \neq i \tag{3.4}$$

— Each vehicle should make a complete tour, and the number of routes (R) which leave and come to the starting point are the same.

$$R = \sum_{j=1}^{N} X_{0j} \tag{3.5}$$

$$R = \sum_{i=1}^{N} X_{ij}, \quad \forall j \in \{N+1, .., N+V\} \tag{3.6}$$

And also, the value of R should not exceed the value of V, where V is the number of drones prepared.

$$R \leq V \tag{3.7}$$

— If t_v is the total flight time of route which traveled by drone v which visit points $(i0, i1, .., il)$, where given by :

$$t_v = \sum_{j=0}^{l_v-1} T_{i_j i_{j+1}} \tag{3.8}$$

and l_v is the length of route which traveled by drone-v, the value of t_v should not exceed the value of Q_v, where Q_v is the maximum time duration can be performed by drone-v.

$$t_v \leq Q_v \tag{3.9}$$

— The length of the created chromosomes (λ) is the sum of the number of points and the number of drones.

$$\lambda = N + V \tag{3.10}$$

3.3 Implementing Genetic Algorithms

In this section, we explain the details of implementing Genetic Algorithm (GA) in our work. According to [1], the workflow of GA is shown in Algorithm 1 :

Algorithm 1 The workflow of GA

```
1: Produce an initial population of chromosomes
2: Evaluate the fitness of all chromosomes
3: Store the chromosome with the best value as the BestChromosome
4: while termination condition not met do
5:    Select fitter chromosomes for reproduction and produce new chromosomes by crossover
      and mutation procedures
6:    Evaluate the fitness of all new chromosomes
7:    if one of new chromosomes has a better fitness then
8:       Store the chromosome as the BestChromosome
9:    end if
10: end while
```

We firstly generate several number of populations, by a random manner, which will become our initial generation or first known solution. Then we evaluate that first population by applying the fittness function to find the best solution to be involved in the next procedures. Next, we will run several procedures such as chromosomes-selection, crossover and mutation, as the evolutionary procedure for generating a new generation, to find the best fitness value as the best solution in answering the optimization problem that we are facing.

Fig. 3.1 shows the brief diagram of the GA which we proposed as the method we used in this work.

3.3.1 Pre-crossover Procedure

Genetic Algorithm is always started by generating first population. It is known as the initial solution for the problem, before the better generation is obtained by each iteration in searching the best generation. As common in GA, we generate the first population randomly, as described in Listing 3.1

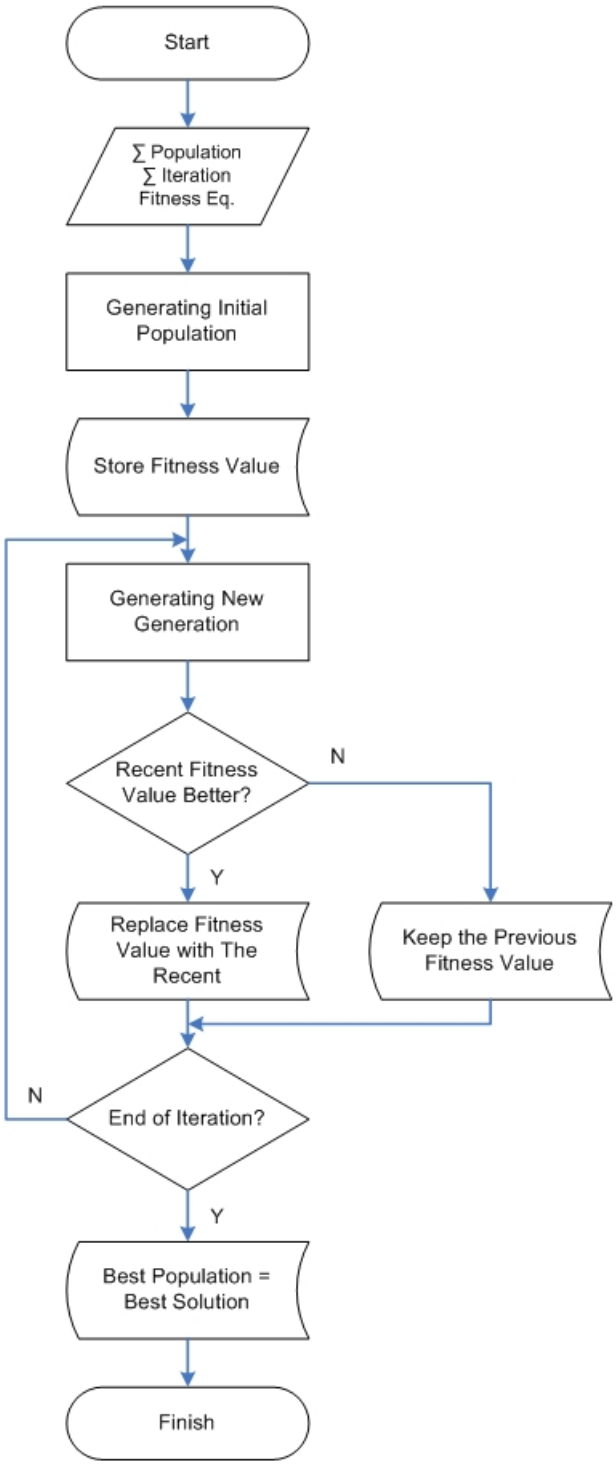

FIGURE 3.1 – Diagram of Genetic Algorithm

```
1 to initial-population
2    create-turtles number-of-population [
3    set string n-values (number-of-points * 20) [random (number-of-points + number
      -of-drones)]
4    set string remove-duplicates string
5    set string remove 0 string
6    set string lput (number-of-points + number-of-drones) string
7  ]
8 end
```

Listing 3.1 – Netlogo pseudo code for generating initial population

The solution will be an array of a chain of number from 1 to $\lambda - 1$ that arranged randomly as shown in Listing 3.1 line-3, and λ is placed in the last chain of number as shown in Listing 3.1 line-6. λ is length of this chain of numbers. In the terminology of GA, this chain of numbers is denoted as chromosomes, and the size of the array is denoted as the population number. The length of the chromosomes is the sum of the number of points (N) and the number of drones (V). For example, if we have 13 number of points and 2 number of drones, so the chromosomes will have 15 chains of number, where ended with 15. The number from $N + 1$ to λ is denoted to the ending point for each drone. For example the chromosomes in population 1 as shown in table 3.1, number 14 and 15 is the ending point for the two drones, so that the paths for each drone are $(1, 2, 9, 11, 10, 12, 4)$ and $(3, 5, 8, 7, 6, 13)$ consecutively.

population 1	1	2	9	11	10	12	4	14	3	5	8	7	6	13	15
population 2	3	2	9	13	11	4	14	5	1	12	6	10	8	7	15
population 3	6	2	11	10	12	4	14	1	3	5	9	7	8	13	15
population 4	1	5	2	10	12	7	4	14	3	8	6	9	11	13	15
population 5	9	11	10	13	3	4	14	1	2	5	8	7	6	12	15
population 6	12	4	1	2	9	11	10	14	13	3	5	8	7	6	15

TABLE 3.1 – The 15-cells of Chromosomes from 6 populations. The yellow highlighted numbers are the ending points (depots) for each drone

After creating the initial population, we evaluate the fitness value of it, and store it as the temporary best solution, as described in Listing 3.2. In line-3, we describe the best solution as the winner, that is the turtle that has the least fitness number. Later, every time we create new generation in each iteration, the fitness value will be used to rate the betterness of the new generation. If the fitness value of the new generation is better than the previous one, we will update the fitness value, and the latest will be denoted as the new temporary best solution, until the iteration is finished.

The creation of the new generation consists of two phases i.e. crossover and mutation. Those phases will be explained in the next subsection. Before we create the new generation, we have to choose two parents from the previous population. To choose two chromosomes from the population that we have, we use the tournament selection procedure.

```
to calculate-fitness
  set fitness fitness-value
  set winner min-one-of turtles [fitness]
end
```

Listing 3.2 – Netlogo pseudo code for finding a chromosome with the best fitness value

```
to choose-parents
  ask turtles [
  let parent1-p min-one-of (n-of tournament-size old-generation) [fitness]
  let parent2-p min-one-of (n-of tournament-size old-generation) [fitness]]
end
```

Listing 3.3 – Netlogo pseudo code for choosing two parents for the new generation

Tournament selection is a method that takes a subset from the solution set randomly then selects the best value among them. This is a part to select 2 chromosomes among population to be parents in the next evolution process, i.e. crossover and mutation. In this part, we choose randomly $tournament - size$-number of chromosomes from solution set which we denote as old generation, as described in Listing 3.3. Then, we choose two select chromosomes to be processed as the parents of the new generation.

3.3.2 Crossover and Mutation

After choosing two chromosomes by tournament selection shown in Listing 3.3, we combine them to produce new generation with a procedures called crossover. It is the important phase in GA, since it can affect the performance of the algorithm. Umbarkar and Sheth [123] stated that the selection of crossover operator has more impact on the performance of GA. In the other words, selecting appropriate crossover technique is important to obtain a better result.

Reference [123] reviews several crossover operators proposed and experimented by various researchers. One of them which called edge recombination crossover (ERX) is a technique that we implement in this work. This technique is also used by [59] in solving TSP. But since the VRP is more complicated than TSP, we made several modifications for this work.

The first step in our crossover procedure is to create an array of datas called an edgemap. Algorithm 2 shows the formula to produce an edgemap from the chosen parents. First, we put two chosen chromosomes as the parents as shown in Table 3.2. Then, we make an edge-map for each parent and extend the table by adding the edges that are incident to each gene/city. The first row from each parent is set as the sequence for the next table we made, as shown in Table 3.4. In the next step, we unite the 2nd and 3rd row of each parent as shown in Table 3.5.

After the final edge-map has already composed, the crossover procedure to generate a child could be started. Algorithm 3 shows the steps of crossover procedure in searching for a

Algorithm 2 Creating EdgeMap

Require: $parent_1$, $parent_2$
1: **for** $i \leftarrow 1$ to 2 **do**
2: $matrix_i[row0] \leftarrow parent_i$
3: $matrix_i[row1] \leftarrow RightRotate(parent_i)$
4: $matrix_i[row2] \leftarrow LeftRotate(parent_i)$
5: $SortColumn(matrix_i)$
6: $Remove(matrix_i[row0])$
7: $i \leftarrow i + 1$
8: **end for**
9: $EdgeMap[row0] \leftarrow matrix_1[row0]$
10: $EdgeMap[row1] \leftarrow matrix_1[row1]$
11: $EdgeMap[row2] \leftarrow matrix_2[row0]$
12: $EdgeMap[row3] \leftarrow matrix_2[row1]$
13: **return** EdgeMap

Algorithm 3 Crossover Procedure

Require: $EdgeMap$
Ensure: $i = 0$
1: $y \leftarrow rand(n)$, $\forall n \in \{0..(\lambda - 1)\}$
2: $x \leftarrow y$
3: $Child[column_i] \leftarrow x$
4: Remove all x in $EdgeMap$
5: $i \leftarrow i + 1$
6: $x \leftarrow$ top data in y^{th} column of $EdgeMap$
7: $Child[column_i] \leftarrow x$
8: Remove all x in $EdgeMap$
9: $i \leftarrow i + 1$
10: **while** $i \leq (\lambda - 1)$ **do**
11: $y \leftarrow$ top data in x^{th} column of $EdgeMap$
12: $x \leftarrow y$
13: $Child[column_i] \leftarrow x$
14: Remove all x in $EdgeMap$
15: $i \leftarrow i + 1$
16: **end while**
17: **return** Child

Chromosomes of Selected Parents															
Parent 1	15	1	2	9	11	10	12	4	14	3	5	8	7	6	13
Parent 2	5	14	1	11	9	7	10	8	3	13	12	6	15	4	2

TABLE 3.2 – Chromosomes of Selected Parents

Edgemap of Each Chromosome															
Parent1	15	1	2	9	11	10	12	4	14	3	5	8	7	6	13
	13	15	1	2	9	11	10	12	4	14	3	5	8	7	6
	1	2	9	11	10	12	4	14	3	5	8	7	6	13	15
Parent2	5	14	1	11	9	7	10	8	3	13	12	6	15	4	2
	2	5	14	1	11	9	7	10	8	3	13	12	6	15	4
	14	1	11	9	7	10	8	3	13	12	6	15	4	2	5

TABLE 3.3 – Edgemap of Each Chromosome

Sorted Edgemap															
Parent1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	15	1	14	12	3	7	8	5	2	11	9	10	6	4	13
	2	9	5	14	8	13	6	7	11	12	10	4	15	3	1
Parent2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	14	4	8	15	2	12	9	10	11	7	1	13	3	5	6
	11	5	13	2	14	15	10	3	7	8	9	6	12	1	4

TABLE 3.4 – Sequenced Egde-Map for each Parent

Sequence														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Chromosomes														
15	1	14	12	3	7	8	5	2	11	9	10	6	4	13
2	9	5	14	8	13	6	7	11	12	10	4	15	3	1
14	4	8	15	2	12	9	10	11	7	1	13	3	5	6
11	5	13	2	14	15	10	3	7	8	9	6	12	1	4

TABLE 3.5 – Final Egde-Map

better solution to the problem, called a child. First, we produce a random number, and we define it as the first point in route. Next, the random number is used to define the next tour, by use it to find the x^{th} column where the first data on it will be taken as the next point.

For example, we got 1 as the random number. So, we put 1 as the first point, and remove 1 in the whole table. Then, we go to 1^{st} column and find $(15, 2, 14, 11)$ in it. We choose 15 since it is the first data on it, and again, we put 15 as the next point and remove 15 in the whole table. Then, we go to 15^{th} column and find $(13, 6, 4)$ in it, and we take 13 as the next point, as shown in Table 3.7. After that, we do it continuously until all the route is completed, as shown in the Table 3.8.

Sequence														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Chromosomes														
		14	12	3	7	8	5	2	11	9	10	6	4	13
2	9	5	14	8	13	6	7	11	12	10	4		3	
14	4	8		2	12	9	10	11	7		13	3	5	6
11	5	13	2	14		10	3	7	8	9	6	12		4
The Child														
1	15													

TABLE 3.6 – Generating a Child

Sequence														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Chromosomes														
		14	12	3	7	8	5	2	11	9	10	6	4	
2	9	5	14	8		6	7	11	12	10	4		3	
14	4	8		2	12	9	10	11	7			3	5	6
11	5		2	14		10	3	7	8	9	6	12		4
The Child														
1	15	13												

TABLE 3.7 – Generating a Child for the Next Gene

The Child														
1	15	13	6	7	8	5	3	14	4	12	10	11	9	2

TABLE 3.8 – The Completed Tour

Initial Result from Crossover Procedure														
1	15	13	6	7	8	5	3	14	4	12	10	11	9	2
Mutated Result														
1	15	13	12	7	8	5	3	14	4	6	10	11	9	2

TABLE 3.9 – Mutation Table

After the crossover procedure is completed, we jump into the mutation procedure, where the algorithm is shown in Algorithm 4. Mutation procedure is able to emerge a new chromosome that is not come from the crossover procedure. In VRP, mutation will result a swap or a sequence change from the solution. The simple explanation on mutation procedure is shown in Table 3.9.

3.4 Graphical User Interface Setup

In order to simulate and investigate the proposed methods, a Graphical User Interface (GUI) is developed. We have searched a framework that can be programmed to model our

Algorithm 4 Mutation Procedure

Require: $Child$

1: $y \leftarrow rand(n), \forall n \in \{0..100\}$
2: **if** $y \leq MutationRate$ **then**
3: $\quad i \leftarrow rand(n), \forall n \in \{0..(\lambda - 1)\}$
4: $\quad j \leftarrow rand(n), \forall n \in \{0..(\lambda - 1)\}$
5: $\quad a \leftarrow Child[Column_i]$
6: $\quad b \leftarrow Child[Column_j]$
7: $\quad Child[Column_i] \leftarrow b$
8: $\quad Child[Column_j] \leftarrow a$
9: **end if**
10: **return** Child

system, and also simulate and investigate events in a 3-Dimensional environment, where user can give instructions at many agents to operate independently and concurrently. There are some available framework to develop this tool, and Netlogo is considered as the chosen framework.

NetLogo is the next generation of the series of multi-agent modeling languages [118]. It was designed by Uri Wilensky in 1999, and since then, it has been developed continuously in Northwestern's Center for Connected Learning and Computer-Based Modelling (CCL). NetLogo enables users to open simulations and "play" with them, exploring their behavior under various conditions. NetLogo is also an authoring environment that is simple enough to enable students and researchers to create their own models, even if they are not professional programmers. That makes Netlogo becomes suitable in education and research, and it is also used by many researchers worldwide. Further information about netlogo is put in the annex of this thesis.

With Netlogo, we can create an interface window as the control panel of the algorithm, which consist of interfaces to control the parameters of experimental and simulation, buttons to control the process, and also monitors to see the solution visually, as shown in Fig. 3.2.

Fig. 3.2 shows the interface window that we made for our work. In the right side, there are two plots to see how the evolutionary procedure works and affects the process in finding the best solution for our problem. In the left-top side, as shown in Fig. 3.3, are the panels to control the parameters in the experiment and simulation. The parameters used in the experiment and their values are shown in Table 3.10.

In addition to it, as shown in Fig. 3.4, in the left-bottom side, there are also several button to control the process of the experiment and simulation.

Between the parameter control and process control, there are some monitors to see the solution visually, as shown in Fig. 3.5.

To run the process, first we click "Load Nodes" to put the set of points in the 3D environment. Then, Netlogo will show the position of points in the 3D environment as shown

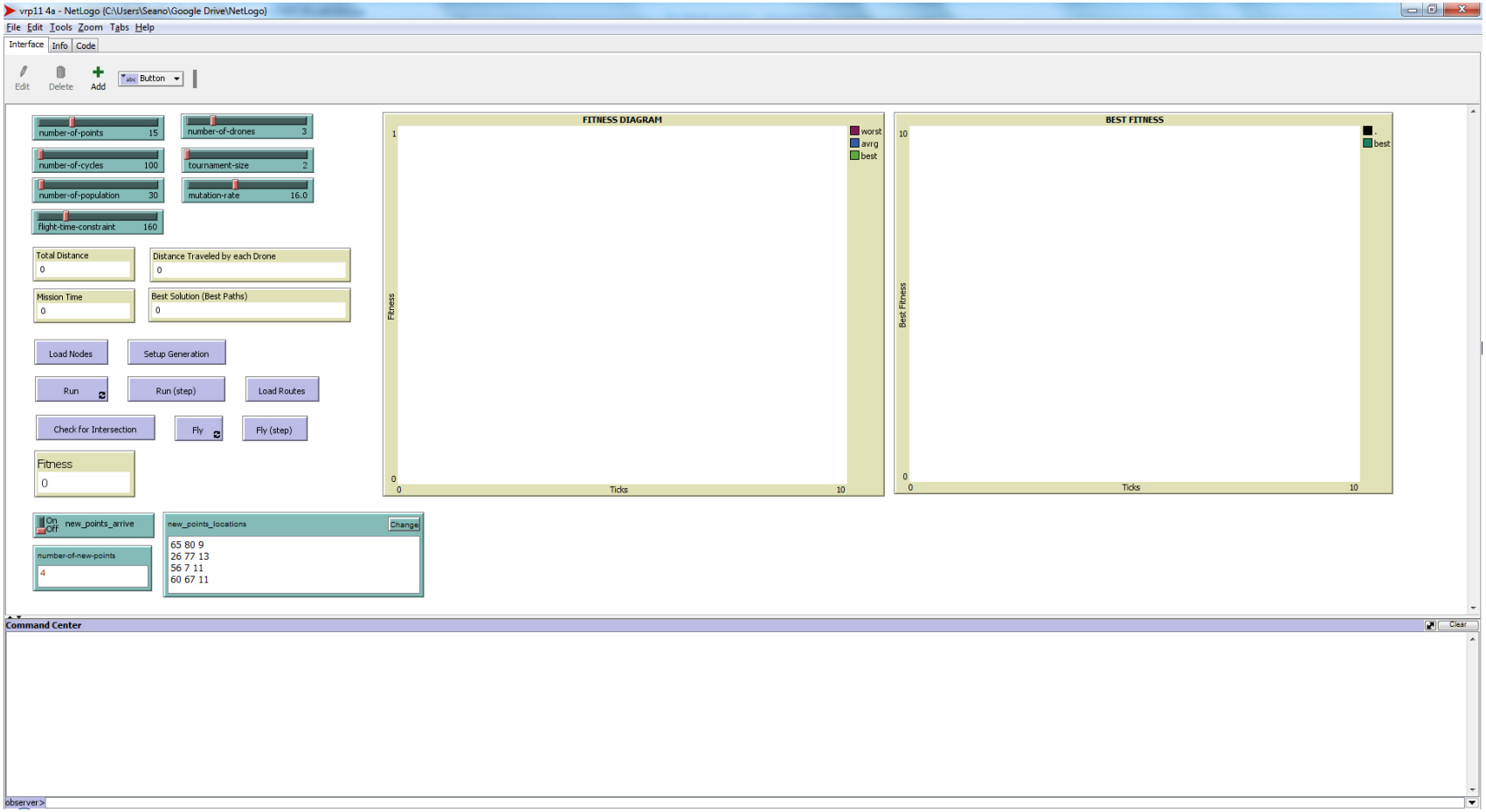

FIGURE 3.2 – The interface window we design

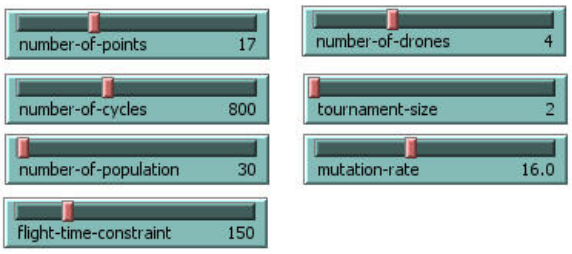

FIGURE 3.3 – The Parameter Control Panels.

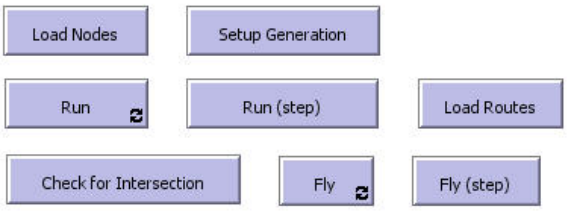

FIGURE 3.4 – The Process Control Buttons.

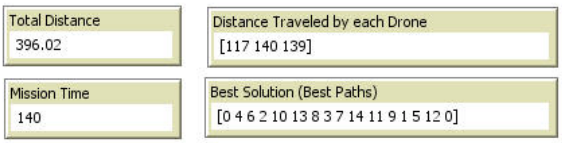

FIGURE 3.5 – The Monitoring Windows.

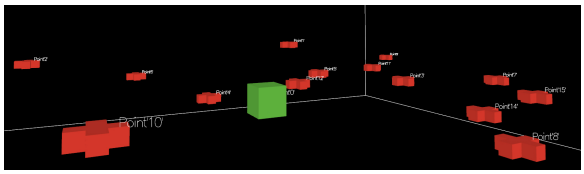

FIGURE 3.6 – 3D View of Loading Points.

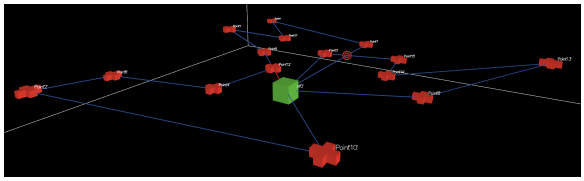

FIGURE 3.7 – 3D View of Path Planning.

Parameter	Value
Number of points	1 - 100
Number of drones	1 - 10
Number of cycles	10 - 2000
Tournament size	2 - 20
Number of population	0 - 1000
Mutation rate	0 - 40 (in %)
Flight time constraint	50 - 500

TABLE 3.10 – The Parameters used in the Interface Window

Title	Function
Load Nodes	to place the set of points in the 3D environment
Setup Generation	generate the initial population
Run	start the iteration
Load Routes	alocate the task to each drone
Check for Intersection	check whether an or several intersection exist
Fly	simulate the flight

TABLE 3.11 – The Control Panel List

in Fig. 3.6. The number of nodes is defined by "number-of-points" panel in the control panel.

Then, as noted in the previous section, the next step is to generate the initial population (solution) in the genetic algorithm by clicking "Setup Generation".

After the initial generation is already setup, it is time to run the algorithm and start the iteration in searching the best solution by clicking the "Run" button. Once the process is completed, the path will appear in the 3D environment as shown in Fig. 3.7. After the best path is already defined, the next step is to allocate the task by pressing "Load Routes". The last button "Fly" is to run the flight simulation of the drones.

3.5 Simulation Results

For finding the best tuning parameter for the algorithm, comparing the combination of parameter is needed. Table 3.12 is made base on our experimental using Netlogo. The first column is the tournament size, and the second is the number of iterations. Column 3 gives us the fitness value of the solution, column 4 gives the best solution, and the last column is the relative error to the best solution known.

Table 3.12 shows that the value of tournament size does not give a big impact in finding the optimal solution. Even though, we can see that, generally, the bigger tournament size, could give the better fitness value.

Because of the little impact of the value of tournament size, in the table 3.13, we will not

Tournament Size	Number of Iterations	Fitness Value	Best Solution[1]	Relative Errors
2	200	164.54	140.27	17.30%
	400	163.94	140.27	16.87%
	600	160.61	140.27	14.5%
	800	154.96	140.27	10.47%
	1000	155.37	140.27	10.76%
	1200	141.25	140.27	0.69%
	1600	140.27	140.27	0%
	2000	140.27	140.27	0%
6	200	167.58	140.27	19.47%
	400	165.69	140.27	18.12%
	600	161.89	140.27	15.41%
	800	160.72	140.27	14.58%
	1000	155.22	140.27	10.66%
	1200	150.79	140.27	7.5%
	1600	140.27	140.27	0%
	2000	140.27	140.27	0%

[1]Best solution is got from exact method calculation

TABLE 3.12 – Computational Result with Population Size = 100, and Mutation Rate = 16%.

Mutation Rate	Number of Iterations	Fitness Value	Best Solution[1]	Relative Errors
16%	200	160.58	140.27	14.48%
	400	158.55	140.27	13.03%
	600	157.32	140.27	12.15%
	800	148.95	140.27	6.19%
	1000	140.27	140.27	0%
	1200	140.27	140.27	0%
30%	200	157.65	140.27	12.39%
	400	155.63	140.27	10.95%
	600	151.35	140.27	7.9%
	800	145.17	140.27	3.49%
	1000	140.27	140.27	0%
	1200	140.27	140.27	0%

[1]Best solution is got from exact method calculation

TABLE 3.13 – Computational Result with Population Size = 250, and Tournament Size = 2.

compare the tournament size, but mutation rate. The first column is the mutation rate, and the second is the number of iterations. Column 3 gives us the fitness value of the solution, column 4 gives the best solution, and the last column is the relative error.

From the Table 3.12 and Table 3.13, we can see that if population size and mutation rate

increase, the solution are getting better, the relative errors are decreasing, and the number of iterations needed is also decreasing.

3.6 Hybridizing Saving and Genetic Algorithm

The main reason of hybridizing two or more algorithms in solving the combinatorial optimization problem is to improving the result and the performance of the algorithm. We need to consider two criteria in determining which algorithm we use to solve the optimization problem : speed and accuracy[28]. Beside those main criteria, there are also another additional attributes that are also essential : simplicity and flexibility [28]. But, the problem we frequently meet is not easy to get those criteria in a heuristic we apply. If it has speed, sometimes it isn't accurate. Otherwise, if it has accuracy, it isn't fast.

None of the classical heuristics fares very well on accuracy and flexibility. In this category, saving algorithm (SA) has at least the distinct advantage of being very quick and simple to implement. SA is one of the best known and remains widely used in practice to this day. It is a heuristic with the best score in speed and simplicity as well [28]. SA contains no parameters and easy to code, that is why it has a good score in simplicity. But, as stated by [78], saving algorithm does not work well when the scale of costumer increases. So, we need to apply another algorithm to improve and boost the weak attributes of SA. This is the main reason of hybridizing.

SA is one of the best and most famous in constructive heuristic [27] [28]. In order to boost their performance, most constructive procedures are followed by an improvement phase [27]. Compared with classical heuristics, metaheuristics perform much more thorough search of the solution space, allowing inferior and sometimes infeasible moves, as well as recombinations of solutions to create new ones. This area of research has experienced a formidable growth over the decades and has produced some highly effective and flexible VRP heuristics [28]. A metaheuristic that fit and compatible to play the role as an improvement heuristics methods is Genetic Algorithm (GA) [15].

In decades, Hybrid genetic algorithms have received significant interest. A genetic algorithm is able to incorporate other techniques within its framework to produce a hybrid that reaps the best from the combination [40]. Not only in VRP application, but also in distribution [50] [69], industrial application [61], economic and banking [86], etc.

In [50], a hybrid GA is used for a multi-time period production/distribution planning. It deals with a production/distribution problem to determine an efficient integration of production, distribution and inventory system so that products are produced and distributed at the right quantities, to the right customers, and at the right time, in order to minimize system wide costs while satisfying all demand required. In [69], a hybrid GA is applied for the design of water distribution networks. It uses heuristic-based approach to provide a good

initial population for genetic algorithm runs.

In industrial applications, paper [61] presents a hybrid GA to optimize the sequence of component placements on a PCB and the arrangement of component types to feeder simultaneously for a pick-and-place machine with multiple stationary feeders, a fixed board table and a movable placement head. In economic and banking field, [86] uses hybrid GA and support vector machines for bankruptcy prediction. GA is used to optimize both a feature subset and parameters of the support vector machine (SVM) for a bankruptcy prediction, an important and widely studied topic since it can have significant impact on bank lending decisions and profitability.

In the field of Operation Reasearch (OR), especially in solving VRP, we found a lot of papers that study and discuss the combination algorithm using GA as one of the combined algorithm, and mostly combined with Tabu Search (TS). Few of them are [25] [12] [21] [22] [57] [64] [60] [80] [82] and [125]. To our knowledge, [54] is the first to hybridize GA with TS for optimization purpose, while [25] is the first to hybridize GA with TS for solving VRP. [25] focuses on the study of a hybrid of two search heuristics, TS and GA on Vehicle Routing Problem with Time-Windows (VRPTW). Papers [21] and [12] implement hybrid GA to handle VRPTW. In [22], GA is hybridized with a decomposition technique. GA is developed to solve the clustering problem, while the decomposition technique is formulated to solve the set of traveling salesman problems for each cluster. Reference [57] applies combination between GA and TS in solving a practical OVRP, to optimize the mines material transport vehicle routing in a Chinese Coal Mine. Ismail and Irhamah [64] propose a hybrid GA-TS in considering a version of the stochastic vehicle routing problem where customer demands are random variables with known probability distribution. Paper [80] uses some heuristic in addition during crossover or mutation for tuning the system to obtain better result. Miao et al [82] combines GA and TS in addressing a Three-Dimensional Loading Capacitated Vehicle Routing Problem (3L-CVRP) which combines a three-dimensional loading problem and vehicle routing problem in distribution logistics. GA is developed for vehicle routing and TS for three-dimensional loading, while these two algorithms are integrated for the combinatorial problem. Paper [125] deals with hybrid GA–TS algorithm for open vehicle routing optimization of coal mines material in Zhengzhou, China.

The previous section shows us that applying GA in our project gave us a satisfactory result. But later, we found that if the number of points is hugely augmented, the number of iterations to get a satisfactory result would be increasing extremely.

In this thesis, we combine GA with saving algorithm (SA) in solving VRP for a fleet of drones. Clarke and Wright's [26] SA is developed to solve combinatorial optimization problems since it can be very difficult to be solved by an exact method. This method is started by calculates saving amount for every pair of points, as shown in the line 5 to 8 in Listing 3.4. Those saving amounts are then ranked from the biggest to the smallest, as shown in Listing

```
to constructing-saving-table
    while [a < number-of-points] [
      set x_a item a p-x  set y_a item a p-y  set z_a item a p-z
      set x_b item b p-x  set y_b item b p-y  set z_b item b p-z
      ask patch x_a y_a z_a [set da  distancexyz-nowrap item 0 p-x item 0 p-y
    item 0 p-z]
      ask patch x_b y_b z_b [set db  distancexyz-nowrap item 0 p-x item 0 p-y
    item 0 p-z
                              set dab distancexyz-nowrap x_a y_a z_a]
      set sab (da + db - dab)
      set data_saving lput a data_saving
      set data_saving lput b data_saving
      set data_saving lput sab data_saving
      set saving_table lput data_saving saving_table
      set data_saving []
      set b b + 1
      if (b > number-of-points) [set a a + 1  set b a + 1]
    ]
end
```

Listing 3.4 – Netlogo pseudo code for constructing saving table

3.5. Starting from the biggest amount, the paired points are linked and added to the tour, and the other points are placed on routes into which they can be linked (otherwise a new route is started) until a constraint is reached, as shown in Algorithm 5.

We found few papers that combine GA with SA in solving VRP, for example [60] and [89]. In [60], GA is hybridized with three heuristics. [60] A hybrid genetic algorithm for the multi-depot vehicle routing problem. The three heuristics hybridized in the algorithm are SA, the nearest neighbor heuristic, and the iterated swap procedure. Paper [89] also combined GA and SA in their work, but in different way with ours. They implemented the SA in order to generate new generation, but we implement SA in order to generate initial population as shown in the diagram in fig. 3.8.

GA can replace the weaknesses of SA, vice versa. Hybridizing GA and SA is perfect for each other. GA can play a role as improvement phase heuristic for SA, since SA is a construction heuristic as shown in fig. 3.9 and table 3.14. Otherwise, SA could be the perfect initial population constructor for GA. Since one of the basic items that need to be carefully considered for the algorithm to work as effective as possible is : a good initial population constructor.

	Speed	Simplicity	Improvement	Accuracy
Genetic Algorithm (GA)	Slow	Average	Yes	Low
Saving Algorithm (SA)	Fastest	High	No	High
Hybrid GA+SA	Fast	High	Yes	Very High

TABLE 3.14 – Hybrid vs Non-Hybrid

```
to sorting-saving-table
    set saving_table_temp saving_table
    set a 0  set b 0
    let biggest_temp 0  let biggest_item# 0
    let saving_sorted []
    while [length saving_table_temp > 0] [
      set a 0  set b 0
      set biggest_temp 0
      set biggest_item# 0
      while [a < length saving_table_temp] [
        set b item 2 item a saving_table_temp
        if (b > biggest_temp) [set biggest_temp b  set biggest_item# a]
        set a a + 1
      ]
      set saving_sorted lput item biggest_item# saving_table_temp saving_sorted
      set saving_table_temp remove-item biggest_item# saving_table_temp
    ]
end
```

Listing 3.5 – Netlogo pseudo code for sorting saving table

3.7 Simulation Results for The Hybrid of Saving and Genetic Algorithm

The previous section shows us that applying GA in our project gave us a satisfactory result. But later, we found that if the number of points is hugely augmented, the number of iterations to get a satisfactory result would be increasing extremely.

Therefore, we applied Clarke and Wright's Saving Algorithm (SA) to generate the initial population in GA. So, it is no longer generated randomly, as shown in fig. **??**.

We made a comparison test between a pure GA and a GA combined with SA. We increase the number of the points, and compare the result. Table 3.15, shows us the comparison using 16 points, with number of population = 100. As shown by fig. 3.10, we can see that combining GA and SA will give a better performance in searching the optimal solution of the problem. By using only 200 iterations with this combination, the result still better than using 1600 iterations (8 times more) with the pure GA.

Table 3.16 use bigger number of points than table 3.15, and the comparison result is more encouraging. As shown by fig. 3.11, the combination still give better performance, and more significant.

Few number of points is added in table 3.17, and the comparison is getting more obvious as shown by fig. 3.12. With a little number of iterations, which means also a little number of time, the combination of GA and SA could give a better result, as indicated with the small relative error.

From the data we get in this section, we found that the result of hybridizing GA and SA is encouraging and also satisfying. The comparison result also shown that this hybrid give

Algorithm 5 Constructing Path

Require: *SortedSavingTable*
Ensure: $i = 0$
while $i < length(SortedSavingTable)$ **do**
  $pair(i, j) \leftarrow SortedSavingTable[i]$
  if neither i nor j have already been assigned to a route **then**
    include i and j in new route
  else if one of the two points (i or j) has already been included in an existing route and that point is not interior to that route **then**
    if adding $pair(i, j)$ to the route will not exceed the constraint **then**
      $pair(i, j)$ is added to that route
    end if
  else if both i and j have already been included in two different existing routes AND neither point is interior to its route **then**
    if merge the two routes will not exceed the constraint **then**
      the two routes are merged
    end if
  end if
  $i \leftarrow i + 1$
end while
return *Route*

Number of Iterations	Fitness Average of GA	Relative Error	Fitness Average of GASA	Relative Error
200	471.59	20.68%	398.97	2.10%
400	451.65	15.58%	398.55	1.99%
600	417.85	6.93%	401.20	2.67%
800	416.62	6.62%	398.27	1.92%
1000	417.25	6.78%	394.44	0.94%
1200	418.97	7.22%	400.78	2.56%
1600	432.01	10.56%	392.41	0.42%

TABLE 3.15 – GA vs GASA Experiment Table with Number of Points = 16 and Number of Population = 100.

more significant result when the number of points is increasing.

3.8 Conclusions

In this chapter a preview and idea about hybridizing genetic algorithm (GA) and saving algorithm (SA) for 3-dimensional vehicle routing problem in a fleet of drones were discussed. In general, we are confident that SA could be the best and fastest GA's initial population constructor, since SA is a heuristic with the best score in speed and simplicity. Hybridizing GA and SA is perfect for each other. GA can play a role as improvement phase heuristic for

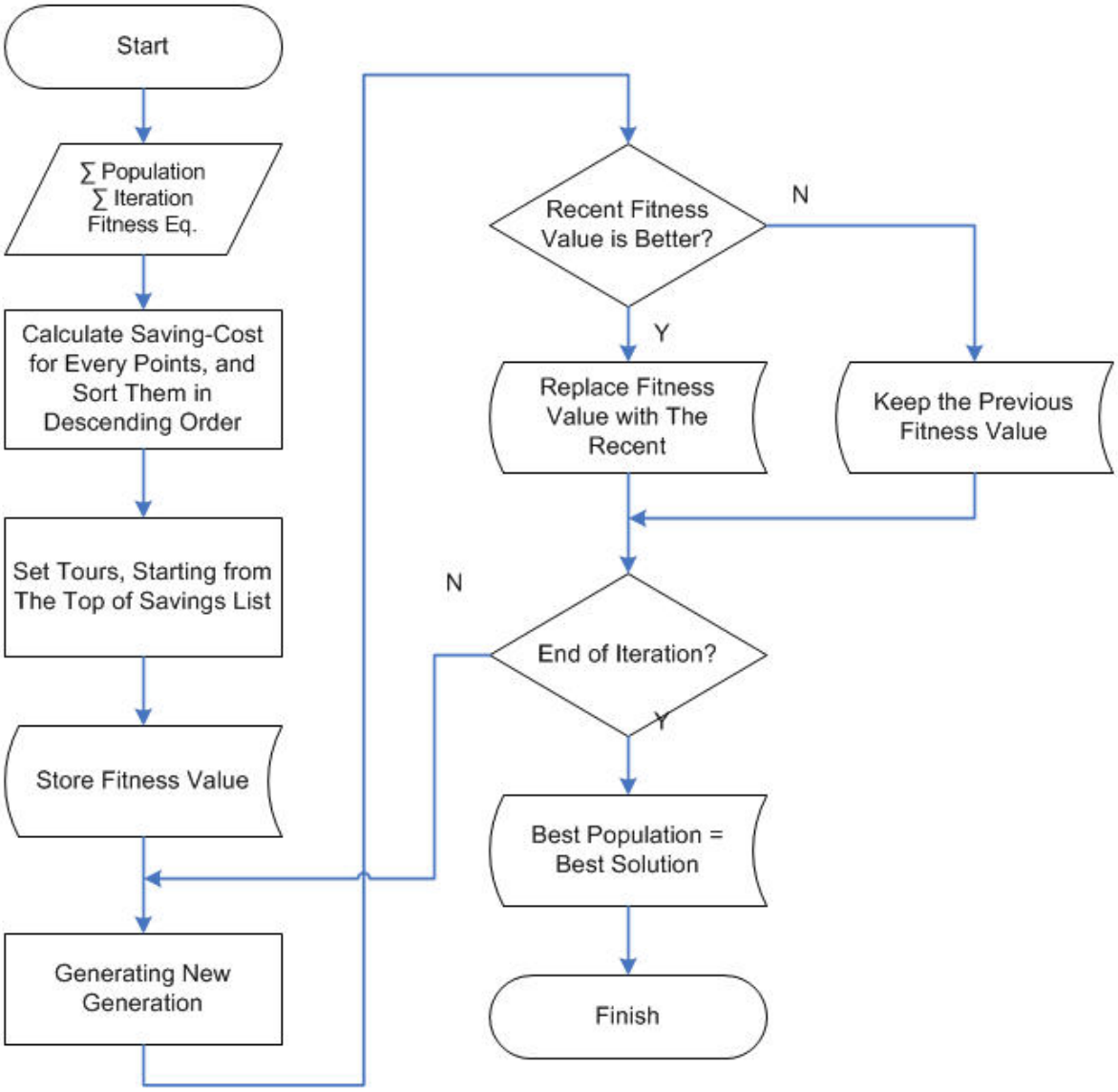

FIGURE 3.8 – Diagram of Combining Genetic Algorithm and Saving Algorithm.

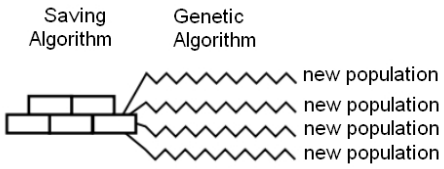

FIGURE 3.9 – Construction and Improvement Heuristic. SA as the Construction Heuristic, and GA as the Improvement Heuristic.

Number of Iterations	Fitness Average of GA	Relative Error	Fitness Average of GASA	Relative Error
200	648.30	43.99%	460.03	2.17%
400	603.8	34.11%	460.03	2.17%
600	585.13	29.96%	460.03	2.17%
800	573.32	27.34%	460.03	2.17%
1000	559.15	24.19%	460.03	2.17%
1200	511.72	13.65%	459.09	1.97%
1600	496.84	10.35%	450.24	0.00%

TABLE 3.16 – GA vs GASA Experiment Table with Number of Points = 22 and Number of Population = 100.

Number of Iterations	Fitness Average of GA	Relative Error	Fitness Average of GASA	Relative Error
200	674.85	43.85%	475.40	1.33%
400	630.34	34.36%	473.70	0.97%
600	595.81	27%	475.97	1.45%
800	602.77	28.48%	475.12	1.27%
1000	572.42	22.01%	475.12	1.27%
1200	588.14	25.36%	475.12	1.27%
1600	574.67	22.49%	473.30	0.97%

TABLE 3.17 – GA vs GASA Experiment Table with Number of Points = 25 and Number of Population = 100.

SA. In the other hand, SA could be the perfect initial population constructor for GA.

The first experimental result in this chapter shows satisfactory results in applying Genetic Algorithm (GA) for this work. But since we need to hugely increase the number of iteration when several points is added, we hybridize GA with Saving Algorithm (SA). SA in inserted in GA as the constructor of the initial population. And we found that the result of this hybridization of GA and SA is encouraging and also satisfying. The comparison results also show that this combination gives more significant result when the number of points is increasing.

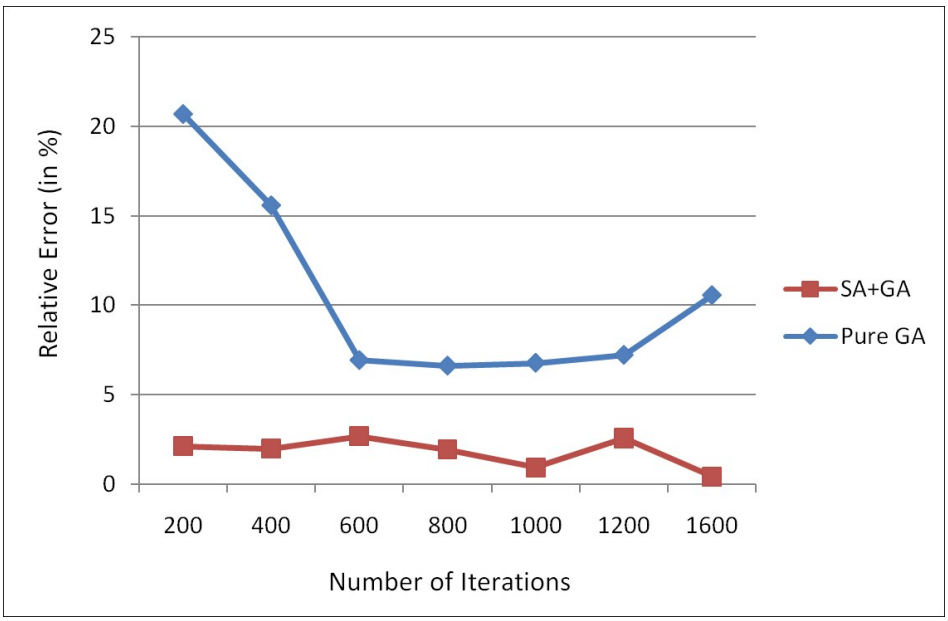

FIGURE 3.10 – "GA" vs "SA+GA" with Number of Points = 16 and Number of Population = 100.

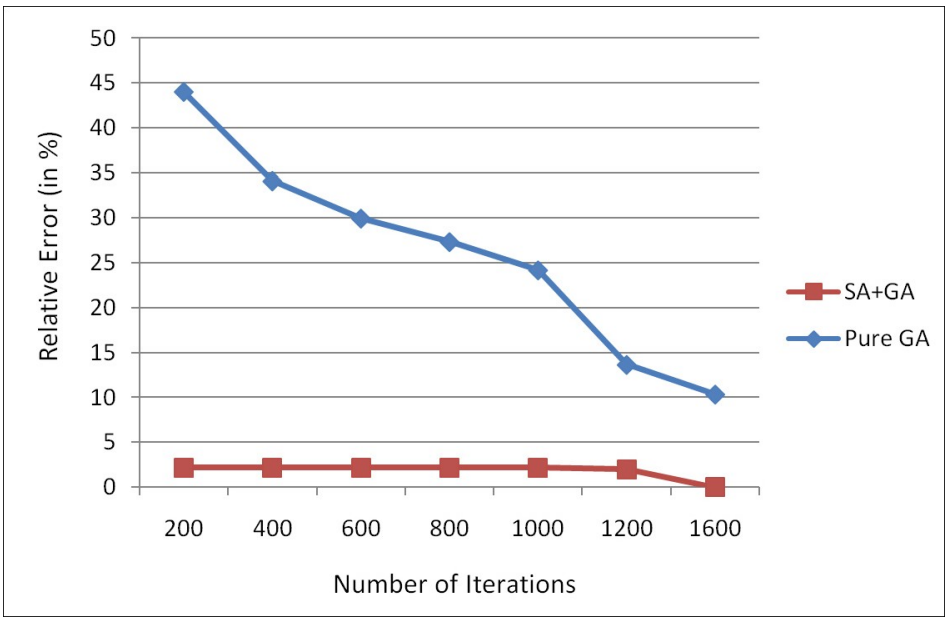

FIGURE 3.11 – "GA" vs "SA+GA" with Number of Points = 22 and Number of Population = 100.

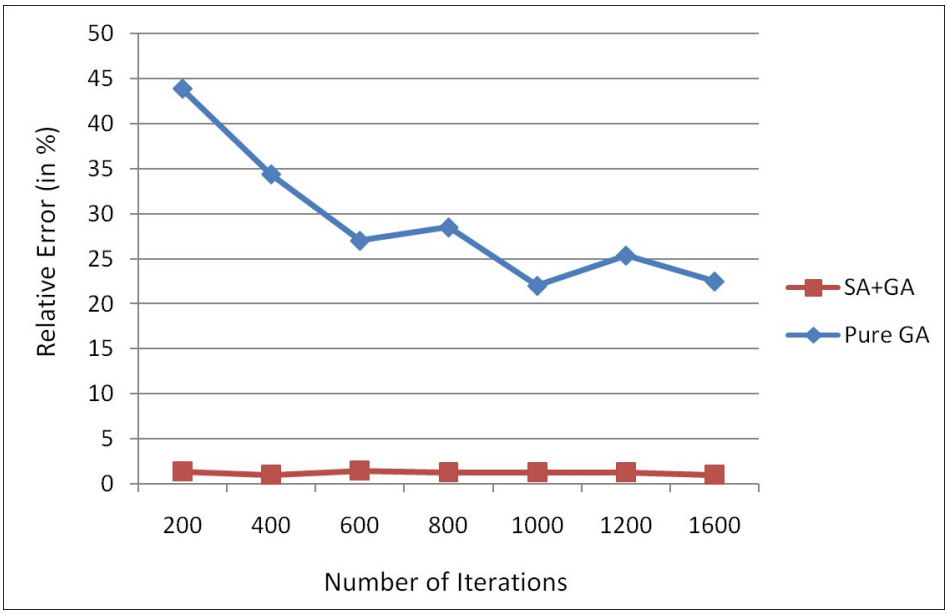

FIGURE 3.12 – "GA" vs "SA+GA" with Number of Points = 25 and Number of Population = 100.

Chapter 4

Reserve Open Vehicle Routing Problem for Dynamic Scenarios in a Fleet of Drones

4.1 Dynamic Vehicle Routing Problem (DVRP)

Nowadays, drone plays a big role in civilian purposes, and it will getting bigger and more important in the future. Because of it's flexibility and versatility, the application of drone is more extensive, covering surveillance, intelligent logistics, search and rescue, scientific studies, etc.

New trends are moving towards to manage a fleet of drones in order to achieve the given mission. This issue opens many research opportunities, and our project is made to answer the challenge, to develop a platform for managing a fleet of drones. As being researched in decades, Vehicle Routing Problem (VRP) is a perfect study to answer this challenge, in order to allocate the tasks and find the best path for each drone, with several constraints to be considered.

Lately, in the real-world, the mission may be affected by uncertainty and may be not always completely available because of unexpected or uncertain events, such as ongoing customer requests, actual amount of demand, failure of one or more drones, etc. To deal with this "dynamic" challenge, the study of the dynamic version of VRP has been done. During the decades, number of papers have issued in many journals and books discussing DVRP. Psaraftis [95] [96] [97] was among the first to study dynamic versions of the VRP. In [96], he outlines the status and prospects for future research within DVRP. Pillac et al [93] provides a review in this subject with a broad introduction to it. The most common source of dynamism in vehicle routing is the online arrival of customer requests during the operation. More specifically, requests can be a demand for goods [63], services [116] and travel time [5].

Several strategy have been applied by researchers in order to solving the DVRP. Pavone [91] uses queueing strategy in modelling the dynamic routing problem for a robotic system. Ichoua et al. [63] and Attanasio et al. [5] tackled the dynamics problem with tabu search (TS). Pureza and Laporte [98] introduces the request buffering that consists in delaying the assignment of some requests to vehicles in a priority buffer, so that more urgent requests can be handled first in the emerged dynamic scenario. Branke et al [18] and Thomas [116] implement the waiting strategy in DVRP. The waiting strategy consists in deciding whether a vehicle should wait after servicing a request, before heading toward the next customer; or planning a waiting period on a strategic location. This strategy is particularly important in problems with time windows, where time lags appear between requests. Aside from the waiting after or before servicing a customer, a vehicle can be relocated to a strategic position, where new requests are likely to appear. Larsen [75] applies the relocation strategy to a DVRP. Sarasola et al.[107] applies variable neighborhood search (VNS) for the stochastic and dynamic VRP. Okulevicz et al. [88] use a two-phase particle swarm optimization (PSO) in solving the DVRP. In each time slice, a PSO assigns costumers to vehicles, then different PSOs (hybridized with 2-Opt algorithm) solve a TSP instance for each single vehicle. Schyns [110] applies ant colony system (ACS) in the DVRP, and proposes an adaptation of ACS in a

context where the objective is the responsiveness, i.e. servicing a customer as soon as possible in its time window.

DVRP force making decisions in an online manner, which often compromises reactiveness with decision quality. In other words, the time invested searching for better decisions, comes at the price of a lower reactiveness to input changes [93]. This aspect requires a good decision, at the time when new requests appear, to be made as fast as possible. For that reason, we implement the reverse open vehicle routing problem (ROVRP) to deal with the DVRP. To us, ROVRP is another approach for DVRP, since it is used in updating vehicle return-trip to the depot. That is why, we use this approach in our thesis as well. We believe that ROVRP is fast enough to carry out this challenges.

4.2 Reverse Open Vehicle Routing Problem (ROVRP)

In this thesis, we consider a special variant of VRP called Reverse Open VRP (ROVRP) to deal with the dynamic scenario in the mission of a fleet of drones. It is a reverse version of the Open VRP (OVRP), which has been presented before by Sariklis and Powell [108] in their paper on distribution management problems. The special characteristic which distinguishes OVRP from the classic VRP is that the vehicles are not demanded to return to the depot. In different cases, the vehicles could return to the depot by tracking the same tour in reverse order [92].

The OVRP has been discussed in several studies, for example in a train plan model for British Rail freight services through the Channel Tunnel [46], a case study for the school bus routing problem in Hong Kong [77], the distribution of fresh meat in Greece [115], the distribution of a daily newspaper in the USA [104] [24], a lubricant distribution problem in Greece [100], and a mines material transport vehicle routing optimization in China [125]. Whereas, according to our knowledge, the ROVRP has not been much discussed. We found a work from Schopka and Kopfer [109] who modified the Adaptive Large Neighborhood Search (ALNS) for generating ROVRP in updating vehicle return-trip to the depot. Also, Park and Kim [90] and Li and Fu [77] who reviewed and made a case study of the School Bus Routing Problem.

We apply ROVRP to define a set of to-depot returning vehicle routes, in constructing a new paths due to a dynamic scenario in a fleet of drones. We consider a dynamic scenario as shown in Fig. 4.1 and as described as follows. For example, we have 3 drones and 12 targets to be visited, and the drones are assigned to visit the target points as shown in Fig. 4.1(a). In this case, drone $D0$, $D1$ and $D2$ are assigned to visit set of points $\{P3 - P4 - P5 - P6\}$, $\{P10 - P9 - P8 - P7\}$ and $\{P11 - P0 - P1 - P2\}$ consecutively. After we assign the points visiting task to each drone, then the mission is launched as shown in Fig. 4.1(b). Afterward, few or several new targets appear at a random location and at a random time, like points $P12$,

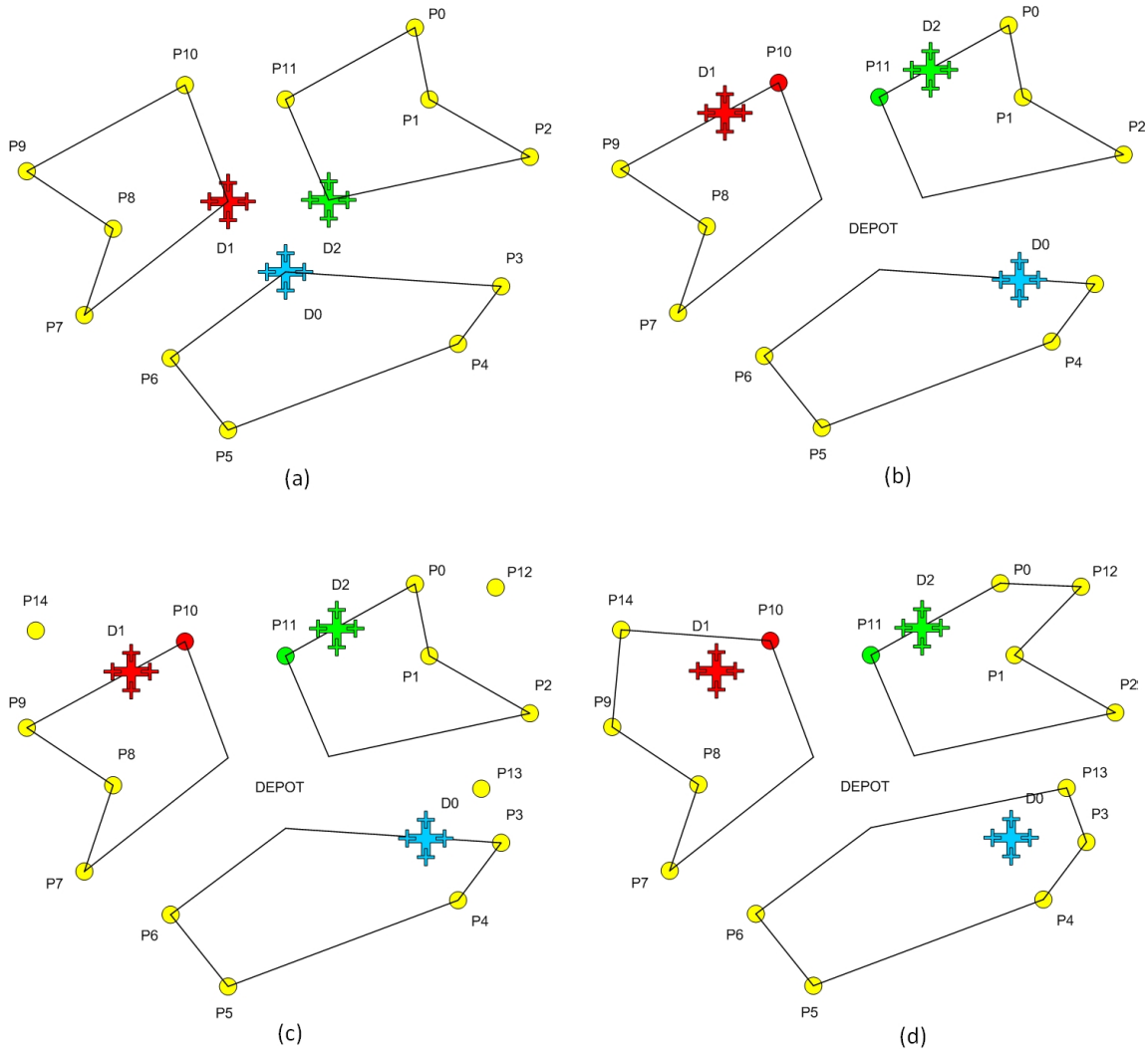

FIGURE 4.1 – The Proposed Dynamic Scenario

$P13$ and $P14$ in Fig. 4.1(c). As consequence, we need to construct new paths for the drones to return to the depot. As shown in Fig. 4.1(d), the remaining route for drone $D2$ is changed from set of points $\{P0 - P1 - P2 - depot\}$ to set of points $\{P0 - P12 - P1 - P2 - depot\}$.

Since the ROVRP is classified as an NP-hard optimization problem, we decide to choose a heuristic method for this project, and we prefer to apply Clarke and Wright's [26] Saving Algorithm (SA) in this project. SA is developed to solve combinatorial optimization problems since it can be very difficult to be solved by an exact method. This method is started by define the *saving* value among every pair of points in the system. Those *saving* amounts are then ranked from the biggest to the smallest. Starting from the biggest amount, the paired points are linked and added to the tour, and the other points are placed on routes into which

they can be linked (otherwise a new route is started) until a constraint is reached. Pichpibul and Kawtummachai [92] proposed this heuristic method to solve the OVRP.

4.3 Problem Formulation

We define ROVRP as a complete undirected network $G = (N, E)$ with a set of edge $E = \{(i, j)|i, j \in N\}$. We have set of drones $V = \{1, 2, .., v\}$ who will be tasked to visit points. We have also a set of points $N = \{0, 1, ..., n, n + 1, n + 2, ..., n + v\}$, where the ending point, known as depot, denoted as point 0. The set of points $\{n + 1, n + 2, ..., n + v\}$ are the initial position (P) of each drones, i.e. the last position of each drone at the time when the dynamic scenario is activated.

The main objective of this project is to minimise the total traveling and vehicle operating cost. It can be formulated as follow :

$$F = min \sum_{v=1}^{V} \sum_{i=0}^{N} \sum_{j=1}^{N+1} c_{i,j} X_{i,j}^{v} \tag{4.1}$$

where $c_{i,j}$, associated with set of edge E, is the traveling cost and also the cost of implementation that a drone travel from point-i to point-j. $c_{i,j}$ is a matrix of a complete graph. Then, $X_{i,j}^{v}$ is a binary variable which defined each arc (i, j), such that equal to 1 if and only if a drone-v visit point-i and then travels directly to point-j, and equal to 0 if otherwise.

Then the problem is designed to find the set of routes in such a way that satisfy these 3 criterias :

1 Each route depart at initial position and must terminates at the depot.

$$R = \sum_{i=N+1}^{N+V} X_{i,j}, \quad \forall j \in \{0, .., N\} \tag{4.2}$$

$$R = \sum_{i=1}^{N+V} X_{i,j}, \quad j = 0 \tag{4.3}$$

where R is the set of r routes which must finish at depot. The number of routes (R) should not exceed the number of drones (V).

$$R \leq V \tag{4.4}$$

2 Each point is visited once and only by exactly one drone.

$$\sum_{v=1}^{V} \sum_{j=0}^{N} X_{i,j}^{v} = 1, \quad \forall i \in \{1, .., N + V\} \tag{4.5}$$

$$\sum_{v=1}^{V}\sum_{i=1}^{N+V} X_{i,j}^{v} = 1, \quad \forall j \in \{0,..,N\} \tag{4.6}$$

And each vehicle leaves from point where it arrived in.

$$\sum_{i=1;i\neq g}^{N+V} X_{i,g} - \sum_{j=0;j\neq g}^{N} X_{g,j} = 0, \quad \forall g \in \{1,..,N\}, \quad j \neq i \tag{4.7}$$

3 The total time spent to visited all customers in each route less than or equal to the maximum flight time of the vehicle assigned to serve the route.

$$t_v \leq Q_v - t \tag{4.8}$$

Q_v is the maximum time duration can be performed by drone-v.
t is the total time that has been spent until the dynamic scenario is occurs.
t_v is the total flight time of route which traveled by drone v which visit points $\{i0, i1, ..., il\}$. The value of t_v is given by

$$t_v = \sum_{j=0}^{l_v-1} T_{i_j,i_{j+1}}, \quad \forall j \in \{1,..,l_v\} \tag{4.9}$$

where l_v is the length of route which traveled by drone-v

In order to forming the vehicle route, we need to make a list of saving amount for every pair of points, called the *saving* value, as the core in Saving Algorithm. The amount is calculated as follow :

$$S_{i,j} = c_{i,0} + c_{0,j} - c_{i,j} \tag{4.10}$$

where $S_{i,j}$ is called saving value between two points i and j.

4.4 Implementation

We implement ROVRP to define a set of to-depot returning vehicle routes from any position but depot, in constructing a new paths due to a dynamic scenario in a fleet of drones as shown in fig. 2.7. At the time $t = x$, when the drones execute their routes, three new requests (P12, P13 and P14) appear. At the time a new request appears, the ROVRP is applied to adjust the initial routes in real-time so that we get the result as shown in fig. 4.2.

To determine the route adjustment as a consequence of the appearance of a dynamic scenario, we implement the ROVRP using an approach which alike to the sweep algorithm : partition first, route later. The steps is shown in fig 4.3.

This method is divided into 2 phases i.e. clustering and routing.

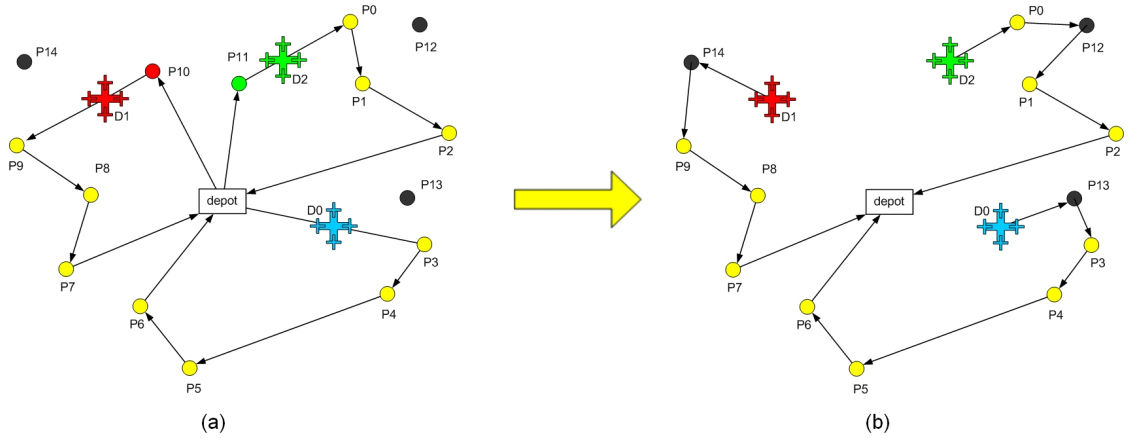

FIGURE 4.2 – The application of ROVRP. (a) At time $t = x$ when three new requests (P12, P13 and P14) appear during mission execution. (b) The result of ROVRP in adjusting the routes.

4.4.1 Clustering

In this phase, we form clusters of to-be-visited points. Each cluster will be served by a single drone. So, the number of clusters is equal to the number of routes, and should not exceed the number of drones. The member of a cluster is the member of established route. For example in Fig. 4.2a, drone $D0$ is assigned to visit set of points $\{P3 - P4 - P5 - P6\}$, so those points mentioned are the member of Cluster 0, which is served by drone $D0$.

When the dynamic scenario is occured, we merge the new points into the closest existing cluster. For example, as shown in Fig. 4.2a, set of points $\{P12, P13, P14\}$ emerge when the mission has been already launched. So, we define onto which cluster is those new points will be merged. Since $P12$, $P13$ and $P14$ are close to the cluster 2, 0 and 1 consecutively, we merge those new points into the clusters mentioned after, as shown in Fig. 4.2b. Listing 4.1 shows the pseudo code for clustering phase.

4.4.2 Routing

We divide this phase into 2 steps i.e. creating saving list and constructing routes.

4.4.2.1 Creating Saving List

The saving list is only made for each cluster of points. It means, if $S_{i,j}$ is the saving value between point-i and point-j, we should ensure that point-i and point-j are in the same cluster.

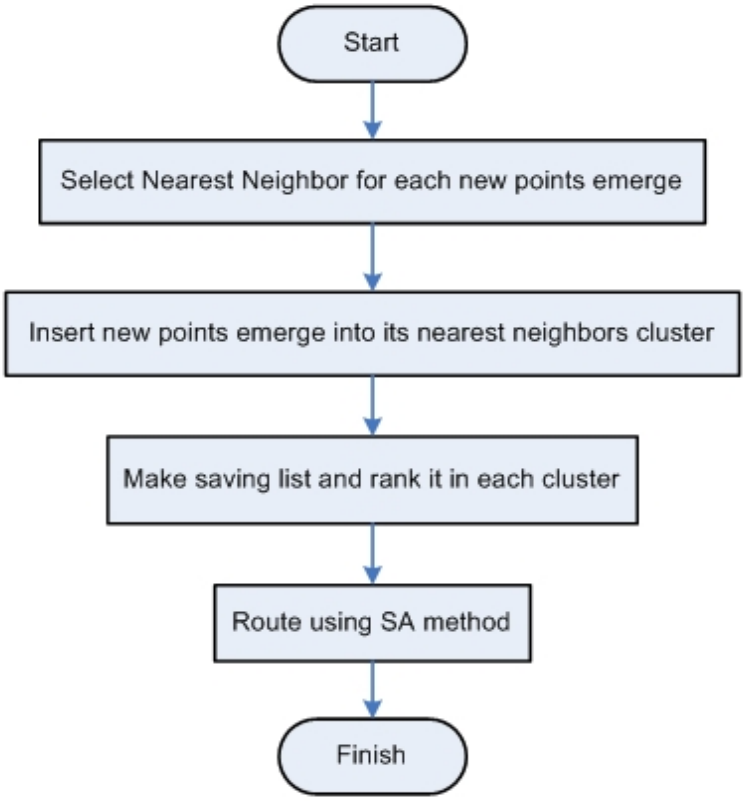

FIGURE 4.3 – The steps of ROVRP implementation.

The savings value between point i and j is calculated as :

$$S_{i,j} = c_{i,P} + c_{P,j} - c_{i,j}$$

where $c_{i,P}$ is the traveling cost between the initial position of the drone (P) and point-i, $c_{P,j}$ is the traveling cost between the initial position of the drone (P) and point-j, and $c_{i,j}$ is the traveling cost between two points i and j. Listing 4.2 shows this calculation in the line 14 to 16. After calculation, all savings values are collected in the savings list, and the values in the savings list are sorted in decreasing order, as shown in Listing 4.3

The saving list creation is described by Fig. 4.4 which shows a cluster of points that consist of 5 points, i.e. point P as the initial position of the drone which appointed as the starting point of the route, set of points $\{1, 2, 3\}$, and point 0 or denoted as *depot* or the end-point. Fig. 4.4a shows that the cluster is a complete undirected network. The table of $c_{i,j}$ for every points in Fig. 4.4 is shown in Table 4.1.

From the information provided in Table 4.1, we can calculate the saving value among

```
to clustering
  set cluster remain
  set point# number-of-points + 1
  while [point# < number-of-points + number-of-new-points + 1][
    set row# 0
    while [row# < route#] [
      set column# 0
      while [column# < length item row# cluster - 1][
        set old-point item column# item row# cluster
        ask patch item point# p-x item point# p-y item point# p-z [
          set point-distance distancexyz-nowrap item old-point p-x item old-point
     p-y item old-point p-z
        ]
        if (point-distance < nearest) [
          set nearest point-distance
          set nearest-point old-point
          set nearest-cluster row#
        ]
        set column# column# + 1
      ]
      set row# row# + 1
    ]
    set x item nearest-cluster cluster
    set x insert-item 0 x point#
    set cluster replace-item nearest-cluster cluster x
    set nearest 1000
    set point# point# + 1
  ]
end
```

Listing 4.1 – Netlogo pseudo code for clustering new points that emerge

$c_{i,j}$	P	1	2	3	0
P	-	4	5	3	9
1	4	-	3	5	10
2	5	3	-	4	7.2
3	3	5	4	-	6
0	9	10	7.2	6	-

TABLE 4.1 – Table of $c_{i,j}$ for every points in Fig. 4.4

rank	$S_{i,j}$	*saving* value	rank	$S_{i,j}$	*saving* value
1	$S_{2,0}$	6.8	4	$S_{2,3}$	4
2	$S_{3,0}$	6	5	$S_{1,0}$	3
3	$S_{1,2}$	6	6	$S_{1,3}$	2

TABLE 4.2 – The saving value list for every points in Fig. 4.4

every pairs in the cluster, and rank them as shown in Table 4.2. Then, we will construct the route based on that table.

```
to constructing-saving-table
  set row# 0
  set column# 0
  while [row# < cluster#][
    set clusterx item row# cluster
    set column# 0
    while [column# < (length clusterx) - 1] [
      set a item column# clusterx
      set b item (column# + 1 + c) clusterx
      set x_a item a p-x  set y_a item a p-y  set z_a item a p-z
      set x_b item b p-x  set y_b item b p-y  set z_b item b p-z
      ask patch x_a y_a z_a [set dab distancexyz-nowrap x_b y_b z_b]
      ask turtle (number-of-population + inters# + collin# + row# + 1) [
                set pb  distancexyz-nowrap x_b y_b z_b
                set pa  distancexyz-nowrap x_a y_a z_a]
      set sab (pa + pb - dab)
      set data_saving lput a data_saving
      set data_saving lput b data_saving
      set data_saving lput sab data_saving
      set saving_table lput data_saving saving_table
      set data_saving []
      set c c + 1
      if (c > (length clusterx) - 2 - column#) [set column# column# + 1 set c 0]
    ]
    set row# row# + 1
    set saving_table_full lput saving_table saving_table_full
    set saving_table []
  ]
end
```

Listing 4.2 – Netlogo pseudo code for constructing saving table

4.4.2.2 Constructing Routes

The route constructing procedure starts from the top of the savings list (the biggest value of $S_{i,j}$). Both point i and j will be merged into the same route if the total flight time does not exceed the drone's maximum flight duration. Algorithm 6 shows the constructing steps in each cluster. Since ROVRP is a to-depot returning vehicle route, the route will be defined as an ordered sequence of points which starts at the initial position of the drone (P), visits all its points once and only once, and terminates at the depot (0). The cost of a route is defined as the sum of the traveling costs through its cluster members. In this phase we aim to find the least cost route in each cluster.

In Fig. 4.4a, P has already been assigned as the starting point, and 0 as the end point. Then, from the top to bottom of the list in the Table 4.2, we merge the pairs into the route. Table 4.2 shows that $S_{2,0}$ has the biggest value in the list, so we merge the edge $(2, 0)$ into the route, as shown in Fig. 4.4b.

According to Table 4.2, the next saving value in the rank is $S_{3,0}$. But, since the point 0 is the end point in the route, it has only one edge adjacent to it. So, we will not merge the edge $(3, 0)$ into the route.

```
to sorting-saving-table
  let biggest_temp 0  let biggest_item# 0
  let saving_sorted []
  set row# 0
  set saving_table_temp saving_table_full
  while [row# < cluster#] [
    set clusterx item row# saving_table_temp
    set a 0  set b 0
    while [length clusterx > 0] [
      set a 0  set b 0
      set biggest_temp 0
      set biggest_item# 0
      while [a < length clusterx] [
        set b item 2 item a clusterx
        if (b > biggest_temp) [set biggest_temp b  set biggest_item# a]
        set a a + 1
      ]
      set saving_sorted lput item biggest_item# clusterx saving_sorted
      set clusterx remove-item biggest_item# clusterx
    ]
    set saving_table_temp replace-item row# saving_table_temp saving_sorted
    set saving_sorted []
    set row# row# + 1
  ]
end
```

Listing 4.3 – Netlogo pseudo code for sorting saving table

In Fig. 4.4c, we merge the edge $(1, 2)$ to the route, as it's saving value is in the next list. The edge $(2, 3)$ could not be merged into the route, because its already 2 edges adjacent to point 2. The edge $(1, 0)$ also can't be merged into the route, because point 0 has already an edge adjacent to it. Then, the last pair in the list, the edge $(1, 3)$, we can merge it into the route, as shown in Fig. 4.4d. Since the point 3 is the last point to be merged in the cluster, the route is already complete.

Furthermore, in case of non-merged points caused by some constraint, each point will be serviced by a drone that starts at the depot, visits the non-merged points, and returns to the depot.

4.5 Experimental Setup and Results

As implemented in chapter 3, in this chapter we also use Netlogo as the tool for programming and simulating. In this chapter, we add 3 new input panel in the interface window that we made for our previous chapter as shown in Fig. 4.5. The whole new interface panel window will be seen as shown in Fig. 4.6.

The toggle switch named *new_points_arrive* in the additional input panels is to indicate the emerge of new point(s) in the mission. It should be manually turned on and off to indicate the emerge of new point(s). The $number - of - new - points$ panel is to inform how many

Algorithm 6 Constructing Path

Require: $SortedSavingTable$
Ensure: $i = 0$
while $i < length(SortedSavingTable)$ **do**
  $pair(i, j) \leftarrow SortedSavingTable[i]$
  if neither i nor j have already been assigned to the route **then**
    if one of the two points (i or j) is zero **then**
      define this pair as the *end_point*
    else
      include i and j in the route
    end if
  else if one of the two points (i or j) has already been included in the route **then**
    if adding $pair(i, j)$ to the route will not exceed the constraint **then**
      if one of the two points (i or j) is zero AND zero has not already been assigned to the route **then**
        define this pair as the *end_point*
      end if
      $pair(i, j)$ is added to that route
    end if
  end if
  $i \leftarrow i + 1$
end while
return *Route*

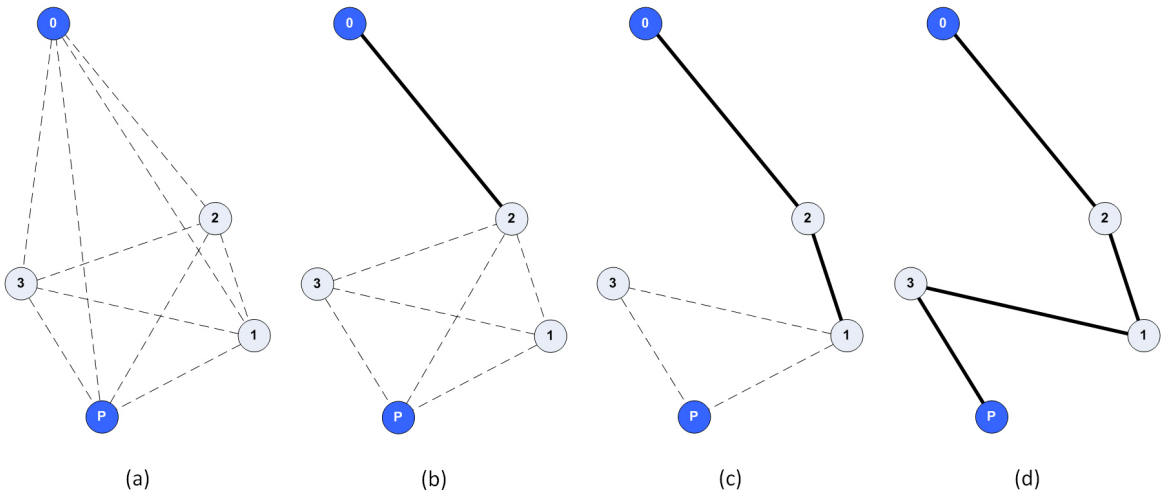

FIGURE 4.4 – The Constructing Route Procedure

points that emerge at a certain time, and the *new_points_locations* panel is to inform the location of the new point(s) in 3-dimensional XYZ coordinates. The amounts in the $number - of - new - points$ and the *new_points_locations* panels are filled manually before we turn the *new_points_arrive* switch on.

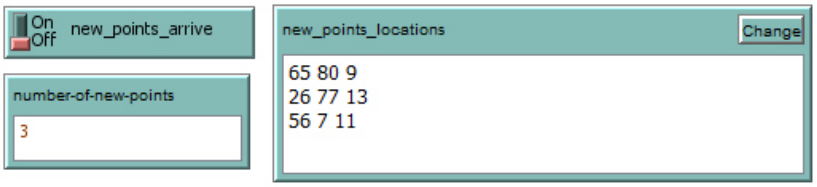

FIGURE 4.5 – The additional input panels

The problem sizes that we use in this section is a small-scale problem. We use 15 points in initial state, as shown in Fig. 4.7, but it is available if we want to enlarge the size to a bigger scale. The problem include a not to be exceeded flight-time-constraint but we do not apply the capacity constraints. The 3-dimensional environment as shown in Fig. 4.7 shows that the problem is started with task allocation and vehicle routing problem that has already done in Chapter 3.

Then, when the mission is already launched, one or more new point(s) emerge in the environment, at a random time, and request a visit. It requires a route recontruction as shown in Fig. 4.8. The blue line in Fig. 4.8 indicates the initial route of the drones, and the yellow line indicates the path that already trailed by drones. The yellow line shows that the route is already reconstructed due to a request by the new points.

Base on our experimental using Netlogo, the Saving Algorithm that we implement is suitable to cover this problem. Table 4.3 and table 4.4 show the result in implementing it. The first column indicates the time when new points are appear (t). At $t = 80$, the mission assigned to drone $D1$ (t_1) is already completed the initial mission, as shown in the 11^{th} row of both tables. The second and third column show the flight time needed for each drone to complete the reconstructed routes. We have 3 clusters since we have 3 drones in the initial state. Column 4 provides us the Best Known Solution (BKS) for each problem and each cluster formed. Then, the last column shows the percentage performance between our proposed implementation and the best known solution, which is calculated as follows :

$$Performance = \left(\frac{t_v + BKS}{BKS}\right) * 100$$

As we can see, most of the result in the performance column is 100%, which means this proposed method is feasible to cover this Reverse Open Vehicle Routing Problem.

4.6 Conclusions

In this chapter we discuss the dynamic scenario that could occur in a mission of a fleet of drones. Literatures denote it as dynamic vehicle routing problem (DVRP). To solve DVRP we use ROVRP approach because it can solve this dynamic problem fast, since we need a

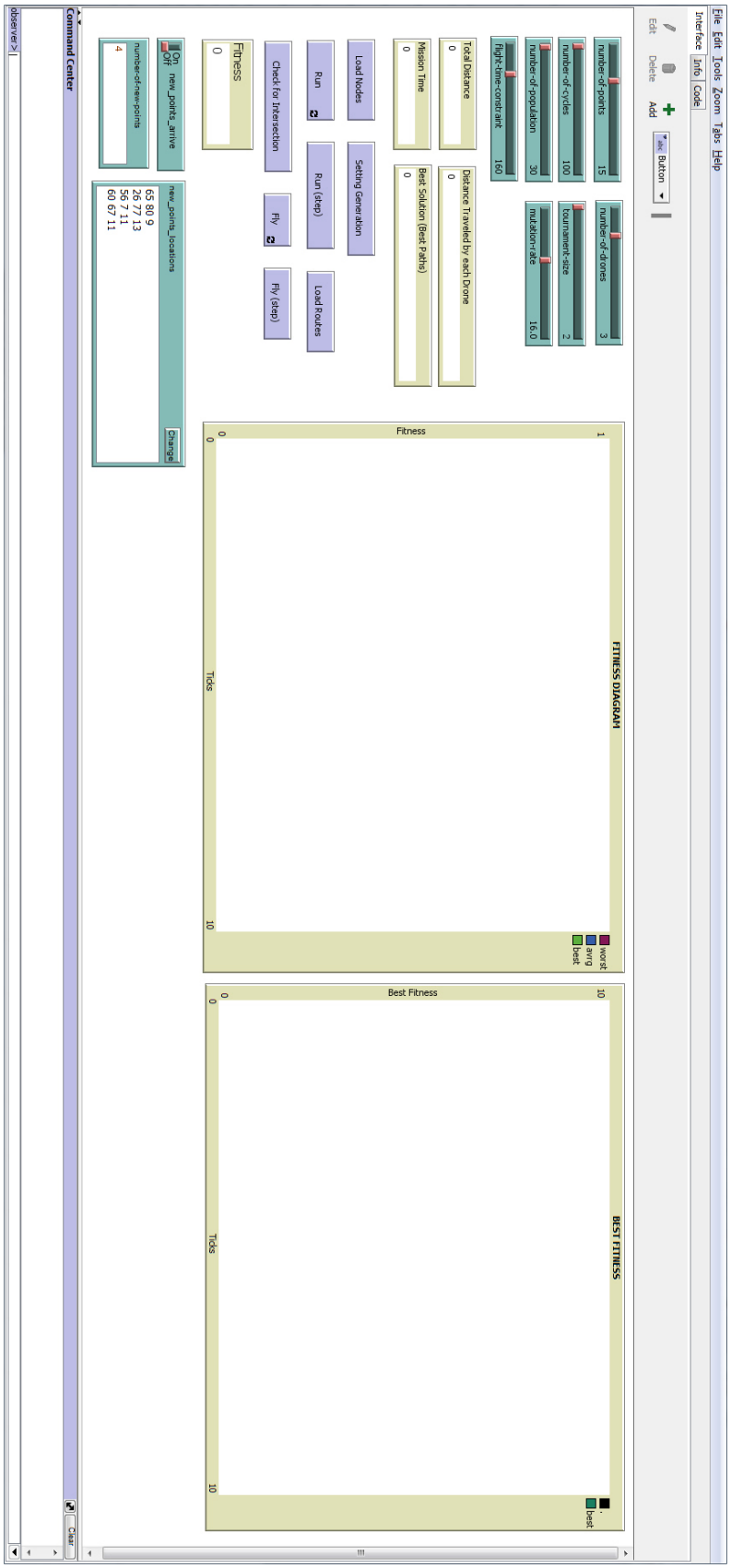

FIGURE 4.6 – The whole interface panels window

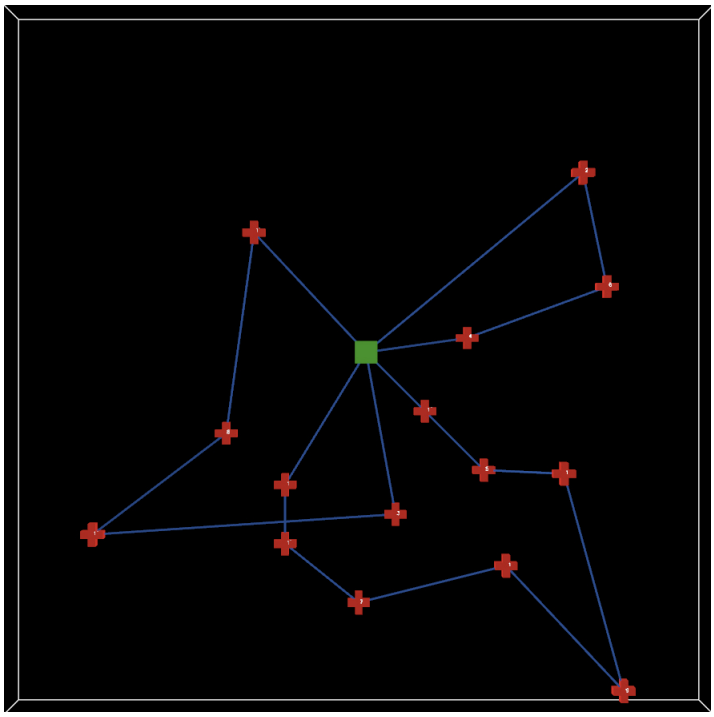

FIGURE 4.7 – 3 Routes has been assigned, with 3 drones and 15 points.

t		t_v	BKS	Perf. (in %)
20	t_0	133,0424	133,0424	100
	t_1	83,2870	83,2870	100
	t_2	138,0829	137,0269	99,23
30	t_0	125,5930	125,5930	100
	t_1	80,5816	80,5816	100
	t_2	142,5546	142,5546	100
40	t_0	118,4111	118,4111	100
	t_1	58,8001	58,8001	100
	t_2	117,6774	117,6774	100
80	t_0	104,1932	104,1932	100
	t_1	mission is already finish		
	t_2	116,6962	116,6962	100

TABLE 4.3 – Performance of the proposed algorithm with 3 new points that emerge.

fast method to solve DVRP. ROVRP is divided into 2 phase : clustering and rotuing. For the clustering method, we simply search the nearest neighbor to insert the new points into it. Finally, for the routing, we implement saving algorithm.

We also have proposed a heuristic approach to solve the Reverse Open Vehicle Routing Problem (ROVRP). We divided our proposed method into 2 phases i.e. clustering and routing.

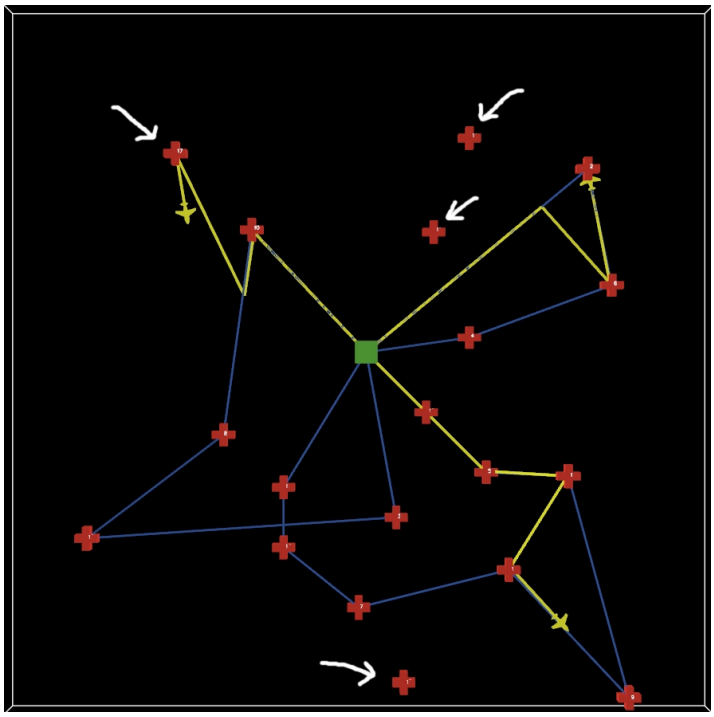

FIGURE 4.8 – The appearance of 4 points which request a visit. The new points are appointed with white arrows.

t		t_v	BKS	Perf. (in %)
20	t_0	133,0424	133,0424	100
	t_1	87,9993	87,9993	100
	t_2	138,0829	137,0269	99,23
30	t_0	123,6548	123,6548	100
	t_1	89,8677	89,8677	100
	t_2	143,2264	143,2264	100
40	t_0	118,4111	118,4111	100
	t_1	83,7682	83,7682	100
	t_2	151,3183	151,3183	100
80	t_0	81,4686	81,4686	100
	t_1	mission is already finish		
	t_2	157,7975	157,7975	100

TABLE 4.4 – Performance of the proposed algorithm with 4 new points that emerge.

We also have done experiments using 15 points and 3 drones in initial state, and also 3 and 4 new points that emerge in dynamic scenario. We use Netlogo in this experiments. The results show that our approach is feasible to cover this problem.

Chapter 5

Conclusion and Perspectives

5.1 Conclusion

In this chapter, we summarize the main contributions achieved in this work, and we put our work in perspective with suggestions for future works. The growth and advancements in the studies and researches on managing a fleet of drones has lead the emerged of this thesis.

In this thesis work, we apply vehicle routing problem (VRP) as a platform for manage a fleet of drones in accomplishing a given mission. So many researchs in finding algorithm to solve VRP, and all of them could be divided into two categories : exact methods and approximation methods. Since VRP is classified as an NP-hard problem, applying approximation methods is more feasible to find solutions, especially for a large-scale problem with huge number of costumers. The approximation methods can be classified into two classifications : classic heuristic and metaheuristic.

In Chapter 3, we presented a hybrid saving algorithm (SA) and genetic algorithm (GA) to solve the VRP for a fleet of drones. GA is a widely used metaheuristic, especially in solving VRP and traveling salesman problem (TSP). Eventhough GA is feasible to find solutions in this combinatorial optimization problem, we were thinking that hybridized it could improve the performance of this algorithm. For that reason, we choosed SA to hybridize with.

As stated in chapter 2, heuristic methods is divided into two types : construction heuristic and improvement heuristic. The use of both types is preferable in order to find a better result. We chose saving algorithm because it is the methods which easy to implemented. But, as a constructive heuristic, saving algorithm usually does not working well when the scale of the VRP is large and increasing. That is why SA needs an improvement heuristic to improve the result. One of heuristic that could play this role is GA.

Later, in Chapter 3, we could see that this combination between SA and GA could improve the performance of the algorithm very satisfactorily. When GA is overwhelmed by the increase of the costumer, the hybrid GA-SA could handle it.

In Chapter 4, we added a dynamic scenario into the problem. We used the reverse open VRP as the approach in solving this dynamic VRP. Also in chapter 4, we could see that ROVRP could be used in tacklig dynamic VRP.

5.2 Perspectives

Managing the Multi Fleet of Drones. Our thesis was taking into account a single fleet of drones, which only have a single depot for the drones. As the future perspective, we consider to manage multi fleet of drones in a huge terrain with number of depots. Each depot has number of drones which can be sent to complete a certain mission.

Better Crossover Operators. Eventhough it is already been discussed for decades, Genetic Algorithm still opens opportunities for performance improvement and exploration. One of the most important item that need to be considered for the algorithm to work as

effective as possible is the crossover operator for reproduction. This operator is the backbone of the Genetic Algorithm. We believe, by finding the best crossover operator, the performance of Genetic Algorithm will be boosted, especially its speed.

The Failure of Drones : Another Dynamic Scenario. In this thesis, as the dynamic scenario, we considered the case where a single or multi requests appear, and asking for a visit during the mission execution by a fleet of drones. For the future works, we consider a dynamic scenario where at a random time, one drone is lost or fall or experience a failure so that it could not be able to continue the mission. So, we need to construct a new paths for the drones to return to the depot. With this scenario, clustering process like we did in chapter 4 is no longer feasible.

As shown in Fig. 5.1(a), a new path is already been constructed. But suddenly, at a random time, one drone is lost or fall or experience a failure so that it could not be able to continue the mission, like drone $D2$ in Fig. 5.1(b). We turn the color of drone $D2$ into grey to show that the drone is fail to complete the mission. Again, we need to construct a new paths for the drones to return to the depot. As shown in Fig. 5.1(c), the remaining routes for both drone $D1$ and $D0$ are changed. $D1$'s remaining route is changed from set of points $\{P9 - P8 - P7 - depot\}$ to set of points $\{P9 - P8 - P7 - P6 - P5 - depot\}$. Also, $D0$'s remaining route is changed from set of points $\{P13 - P3 - P4 - P5 - P6 - depot\}$ to set of points $\{P13 - P4 - P3 - P2 - P12 - P1 - depot\}$.

Managing the Fleet of Heterogeneous Drones. Our thesis was taking into account a fleet of drones that are homogeneous, which every drones have the same characteristic and specification, and also carry the same payloads and devices. In our thesis, the task also homogeneous, so that every nodes or points are eligible to be visited by any drones, since they are homogeneous. For the future works, we consider to take into account the fleet of drones that are heterogeneous, with heterogeneous tasks. Every drones will not carry the same payloads and devices, and only certain drones are eligible to visit certain nodes or points, depend on the tasks in certain points and also the payloads and devices carried by certain drones. In this case, some nodes or points probably visited by more than just a single drone, in case of the task inside it could not be solved and accomplished by a single drone.

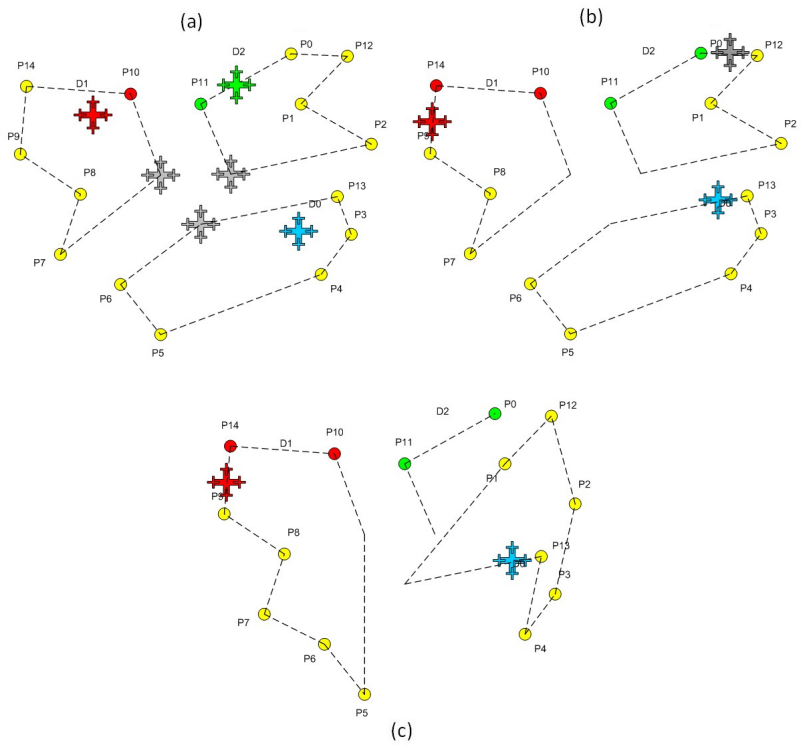

FIGURE 5.1 – The Dynamic Scenario as Perspective

Appendices

Annex A

Constructive Heuristic

Here, we list constructive heuristics that are widely used in slving vehicle routing problem (VRP),

A.0.1 Nearest Neighbor Method

Rosenkrantz et al [103] proposed the nearest neighbor method in 1977. This method start from the depot and search for the nearest unvisited costumer as the next costumer. Repeat this procedure in the condition of not exceeding the capacity until all costumer are visited. the advatages of this algorithm is that the computation speed is fast and we can easily get initial solution. The disadvantage is that it is easy to fall into local optimum.

A.0.2 Sweep Algorithm

The main idea of this algoritmh is : "partition first, route later". [106] presents the sweep algorithm and its application to VRP. The sweep algorithm applies to planar instance of the VRP. The depot is joined with an arbitrarily chosen point. All other nodes are joined to the depot and then aligned by in creasing the angles which are formed by the segment that connects the nodes to the depot. It consists of two parts : - Split : Feasible clusters are initiated formed rotating a ray centered at the depot based on their capacity ; - TSP : A vehicle routing is then obtained for each cluster by solving a TSP.

A.0.3 2-Phase Algorithm

The problem is decomposed into its two natural components : (1) clustering of vertices into feasible routes then use k-opt to optimize the routes respectively and (2) reduce the total travel cost by swapping between routes and optimize the routes with k-opt. The other 2-phase algorithm is proposed by Fisher and Jaikumar (1981). The main idea is to solve a Generalized Assignment Problem (GAP) to decide the feasible cluster and solve the TSP in each route.

A.0.4 Insertion Heuristics

Insertion heuristics are popular methods for solving a variety of vehicle routing and scheduling problems. The main principle of insertion heuristics is to start from a single node that is usually called a seed node and that forms the initial route from the depot. Other nodes are inserted one by one evaluating certain functions to select a node and the place in the route for insertion.

Annex B

Metaheuristics for Solving Vehicle Routing Problem

According to [51], there are a wide variety of metaheuristics. Some properties are use to classify them, such as local search or global search, single-solution or population-based, hybridization or memetic algorithms, parallel metaheuristcs and mature-inspired metaheuristcs. The most common and studied metaheuristics include Ant Colony Optimization, Simulated Annealing, Tabu Search and Evolutionary Algorithm (AE). Genetic algorithm, evolutionary programming and memetic algorithm belong to EA.

B.0.1 Ant Colony Optimization

This metaheuristic is inspired from a natural metaphor, namely the communication and cooperation mechanisms among real ants that allow them to find short paths from their nest to food sources. The communication medium is a chemical compound, known as pheromone, which is laid down on the ground. While an isolated ant would more or less wander randomly, an ant detecting a pheromone path will follow it, with some probability, and will strengthen it with its own pheromone. Thus, the probability that other ants will follow a given path in the future increases with the number of ants that previously followed it. This leads to the emergence of shortest paths, since pheromone tends to accumulate faster on those paths. In the artificial metaphor known as Ant Colony Optimization (ACO) [35], a number of artificial ants construct solutions in a randomized and greedy way at each cycle. Each ant chooses the next element to be incorporated into its current partial solution based on some heuristic evaluation of that element and the amount of pheromone, represented by a weight, associated with it. The pheromone represents the memory of the system and is related to the presence of that element in good solutions previously constructed by the ants. ACO has quite naturally been applied to the Traveling Salesman Problem (TSP), where a shortest Hamiltonian cycle must be found over a complete graph. However, the ACO metaheuristic has also been adapted

to the VRP and some of its extensions.

B.0.2 Greedy Randomized Adaptive Search Procedure

The basic idea of the Greedy Randomized Adaptive Search Procedure (GRASP) [101] is to use a randomized greedy construction heuristic within a multistart procedure to generate a variety of solutions. At each step of the greedy construction heuristic, the elements not yet incorporated into the current partial solution are evaluated with a heuristic function, and the best elements are kept in a restricted candidate list. One element is then randomly chosen from that list and incorporated into the partial solution. When the construction process is completed, the solution is further improved with a local search. The best solution obtained after a certain number of restarts is then returned at the end.

B.0.3 Simulated Annealing

This metaheuristic is a randomized local search method, where a modification to the current solution that leads to an increase in solution cost can be accepted with some probability. This mechanism allows the method to escape from bad local optima. Simulated Annealing (SA) [72] comes from an analogy with the physical annealing process aimed at generating solids with low-energy states. In condensed matter physics, annealing is a process in which a solid is first melted by increasing its temperature. This is followed by a gradual temperature reduction to recover a solid state of low energy. A careful annealing through a series of temperature levels, where the temperature is held long enough at each level to allow the system to reach equilibrium, leads to the more regular structures associated with solids with low-energy states. In a vehicle routing context, a solution or set of routes corresponds to a state and the solution cost to its energy. At each iteration, the current solution is modified by randomly selecting a modification based on a particular class of modifications that defines the neighborhood structure. If the new solution is better than the current one, it becomes the current solution. Otherwise, the new solution is accepted according to a probabilistic criterion, where a modification is more likely to be accepted if a parameter called the temperature (by analogy with the physical process) is high and the cost increase is low. During the procedure, the temperature parameter is progressively lowered according to some predefined cooling schedule, and a number of iterations is performed at each temperature level. When the temperature is sufficiently low, only improving modifications can be accepted and the method stops in a local optimum. As opposed to most metaheuristics, it has been shown that SA asymptotically converges to a global optimum. The success of SA has sparked the development of deterministic analogs whose performance has been quite similar to that of SA : Threshold Accepting [39], Record-to-record Travel [38], and the Great Deluge Algorithm [38]. In these methods, as in SA, the acceptance of deteriorating solutions becomes progressively less frequent as the algorithm unfolds.

B.0.4 Tabu Search

Like Simulates Annealing, Tabu Search (TS) [53] is a local search-based metaheuristic where, at each iteration, the best solution in the neighborhood of the current solution is selected as the new current solution, even if it leads to an increase in solution cost. Through this mechanism, the method can thus escape from bad local optima. A short-term memory, known as the tabu list, stores recently visited solutions (or attributes of recently visited solutions) to avoid short-term cycling. The search stops after a fixed number of iterations or after a number of consecutive iterations have been performed without any improvement to the best known solution.

B.0.5 Variable Neighborhood Search

Variable Neighborhood Search (VNS) [58] is another local search-based metaheuristic which exploits many different transformation classes, or neighborhoods, to escape from bad local optima. When a local optimum is reached with regard to a given neighborhood, another neighborhood is selected and used in the following iterations. More precisely, given a set of (often nested) neighborhoods, a solution is randomly generated in the first neighborhood of the current solution, from which a local descent is performed (possibly based on a completely different neighborhood structure). If the local optimum obtained is not better than the current solution, then the procedure is repeated with the next neighborhood in the nested structure. The search restarts from the first neighborhood when either a solution which is better than the current solution is found or all neighborhoods have been tried. A well-known variant is the Variable Neighborhood Descent (VND) where the best neighbor of the current solution is considered instead of a random one. Also, no local descent is performed on this neighbor. Rather, the latter becomes the new current solution if it provides an improvement. The search is then restarted from the first neighborhood. Otherwise, the next neighborhood is considered.

B.0.6 Genetic Algorithm

Evolutionary algorithms are a large class of metaheuristics, also inspired from a natural metaphor, with Genetic Algorithms (GAs) [55] being one of the best known. Basically, they imitate the technique species evolve and adapt to their environment, according to the Darwinian principle of natural selection. Under this paradigm, a population of solutions (often encoded as bit or integer strings, known as chromosomes) evolves from one generation to the next through the application of operators that are similar to those found in nature, like selection of the fittest, genetic crossover and mutation. Through the selection process, only the best solutions are allowed to become parents and to generate offspring. The mating process, known as crossover, then takes two selected parent solutions and combines their most desirable features to create one or two offspring solutions. This is repeated until a new population of

offspring is obtained. Finally each offspring is randomly perturbed by a mutation operator. Starting from a randomly or heuristically generated initial population, this cycle is repeated for a number of generations, and the best solution found is returned at the end. When applied to vehicle routing problems, the classical GA solution scheme is often modified. In particular, the encoding of solutions into chromosomes is either completely ignored (by applying the various operators directly on the solutions) or designed in a very particular way to take advantage of specialized crossover and mutation operators.

Annex C

Real Life Applications of Dynamic Vehicle Routing Problems (DVRP)

According to [75], we list a number of real-life applications of dynamic vehicle routing problems (DVRP)

C.0.1 The Traveling Repairman

Consider the situation that arises when for instance a bank teller machine breakes down and must be repaired by a service technician. The route to be followed by the technician may be determined using a distance based objective or it might take the urgency of the call (is the teller-machine located in a high intensity area or is it located in a remote area ?) into consideration. This problem is often referred to as the Dynamic Traveling Repairman Problem (DTRP) and is one of the most well-studied dynamic vehicle routing problems. A similar example is the repairman from the electric power company traveling from house to house to repair sudden break-downs in the electric power supply.

C.0.2 Courier Mail Services

Courier mail service companies throughout the world offer to pick-up mail and/or packages at one location and deliver the goods safely at another location within a certain time limit. Often, the mail/packages to be deliveredare not local, but shipped from other cities or countries. Hence, the deliveries are shipped to a hub and then distributed from this hub to the delivery trucks. The deliveries form a static routing problem, because all recipients are known by the driver (and the dispatcher) before the vehicle leaves the depot. However, the pick-ups to be handled during the deliveries has the effect that the problem become dynamic in the sense that the driver and the dispatcher do not have all information on when and where the pick-ups are going to take place.

C.0.3 Distribution of Heating Oil

The distribution of heating oil to private households is often based on the so-called degree-days which is a simple measure of the accumulated outdoor temperature. The oil companies use the degree-day measure to keep track of how much oil the customers have used for heating their houses. Whenever the database tells the routing system that a customer is running low on heating oil, the customer is included in the pool of customers waiting to be served. Eventhough the degree-day customers could be thought of as static customers, the routing problem often becomes dynamic due to the fact that a subset of the customers might run out of oil before the degree-day database includes the customers in the replenishing list. Reasons for this situation could for instance be that a sudden change in the weather causes a high use of oil just before replenishment was to take place or a general higher use of oil due to changes in the behavior of the customer (for example when the house owner invites people to stay and therefore has to heat rooms which are normally left unheated). Experiences show that approximately 20% of the customers visited by the oil company in a day are dynamic customers calling in during the day requesting immediate service. Another element which makes this problem different from - and much more difficult to solve than - the conventional static and deterministic routing problems is the fact that the degree-day measure is not a precise measure for the actual use of heating oil, but merely an estimate. This implies that the problem becomes stochastic in relation to the demand.

C.0.4 Dynamic Dial-A-Ride Systems

Dial-a-ride transportation systems are one application of the general pickup and delivery vehicle routing problem, in which one or more types of commodities must be picked-up at one location and brought to another location where the goods are delivered. One example of a dynamic diala-ride system is the transportation of elderly and handicapped people. In Copenhagen, Denmark, the local urban bus companies also provide a service for elderly and handicapped people. At present time customers are supposed to call in for service one day before the requested trip is going to take place. This policy of course makes the system static, but in the future the bus company might offer the service as an online service for instance via a world wide web based booking system.

C.0.5 Taxi Cab Services

Managing taxi cabs is yet another example of a real-life dynamic routing problem. In most taxi cab systems the percentage of dynamic customers is very high, i.e., only very few customers are known to the planner before the taxi cab leaves the taxi central at the beginning of its duty. A special attribute of the taxi cab routing and dispatching problem is that the state of the taxi can either be for hire or it can be engaged by one or more passengers. When

the taxi is free for hire, the driver often repositions the vehicle to a centrally located taxi rank where the probability of being hailed is higher than it would have been if the driver had chosen to wait at the destination of the last customer. The location of the taxi ranks could either be based on extensive empirical data of calls or it could simply be based upon the intuition and experience of the driver. The policies for assigning customers to taxis differ from country to country and from company to company. The larger taxi cab companies in Denmark are owned by relatively few contractors, each of whom might have between one and 100 taxis. The contractors share a central call and dispatch center. The customers are then assigned to the taxi according to the number of taxi cabs owned by each contractor. This implies that the taxi cab routing and dispatching system will have to use a load balancing strategy when assigning the customers to taxis. This policy means that the contractors could be sure that they will get the best service from the company.

C.0.6 Emergency Services

The dispatching of emergency services (police, fire and ambulance services) resembles the dynamic vehicle routing problem through the fact that requests for service arrive in real-time and that the system resembles a geographical based queueing system. In most situations though, routes are not formed, because the requests are usually served before a new request appears. The problem then is to assign the best vehicle (for instance the nearest) to the new request. Methods for designing emergency service dispatch are therefore often based on location analysis for deciding where to locate the vehicles and crews. This area has been studied mostly from queueing oriented approaches.

Annex D

Static versus Dynamic Vehicle Routing

Psaraftis [95] lists the points that differ the classic VRP and the dynamic version of VRP. Before we proceed with methodological issues, it is important to be explicit with what we mean by the word "dynamic". Although this may seem obvious or times between nodes are known and deterministic, and the salesman spends a known service time at each node. What is the routing policy that minimizes the average, over all demands, expected time until service of the demand is completed ? Alternatively, what is the routing policy that maximizes the average expected number of demands serviced per unit time ?

This problem is dynamic because part of the input required to solve it (that is, which nodes actually request service) is revealed to the dispatcher concurrently with the determination of the route. Given this, it is impossible for an optimal route to be produced in advance. At best, what can be produced is a policy, specifying what action should be taken as a function of the state of the system. More about this problem and its variants later The above examples show that the way information about a particular routing problem evolves through time and is received by the decision maker is critical for the characterization of the problem as static or dynamic. More important, this plays an important role in determining which methodologies can be used.

The following taxonomy can be useful in characterizing attributes of information that forms the input of a certain vehicle routing problem :

— evolution of information : (static/dynamic).

— quality of information : (known-deterministic/forecast/probabilistic/unknown).

— availability of information : (local/global).

— processing of information : (centralized/decentralized).

D.0.1 Evolution of information

The differentiation between static and dynamic inputs is relevant here. The former are known for the entire duration of the routing process and are not updated (although they may very well be functions of time, as in the time-dependent TSP). The latter are not known for the entire duration of the routing process (which may be open ended anyway), and will generally be revealed or updated as time goes on.

D.0.2 Quality of information

Some (or all) inputs to the problem can be known with certainty (deterministic), throughout the duration of the routing process. Examples : Number of nodes, number of vehicles, vehicle capacities, internode distances, etc. Some other inputs may not be known with certainty, but only as forecasts. These inputs are subject to revision as the routing process evolves. Examples : Demand at a certain node, travel times between certain node pairs, etc. Yet other inputs may be probabilistic, that is, follow prescribed probability distributions or evolve according to known stochastic processes. Examples : Demand location on a Euclidean instance is uniformly distributed on the unit square, travel time between nodes follows a prescribed distribution, arc costs follow a Markov process, quantity demanded at a node follows a given distribution, etc. Finally, there may be certain inputs on which no information is available at the time of decision. Examples : The time at which the next demand for service is received, the location of that demand, etc. It is important to realize that the attributes of information quality for a certain input variable are defined for the specific point in time at which the decision has to be made, and may change when time moves along (this is true for dynamic inputs). For instance, a certain variable may be probabilistic now, but becomes deterministic when the realization of its value is revealed. The same is true for forecast and unknown input variables. In addition, the values of probabilities of certain other variables may very well change, as a result of observations during the course of the routing process. The quality of information in dynamic vehicle routing is usually good for near-term events, and becomes poorer for more distant events.

D.0.3 Availability of information

There may be problems in which information is available only on a local basis. For instance, the travel time between two nodes may be random, and its actual value may be revealed (or forecast) only when the vehicle arrives at the starting node. The same may be true when the driver of a tank truck learns of the amount of oil needed to replenish a certain customer's inventory only on location. On the other hand, some inputs may be available globally. For instance, estimated travel times may be received by radio (or by some other device) even for remote parts of the network. Or, customer inventories may be automatically monitored by a

device, and this information may be transmitted to the dispatcher on a global and continual basis. Technologies will generally increase the global availability of information, although the issue of who receives such information and when he receives it is a crucial parameter of system architecture and design. For instance, the central vehicle dispatcher may have at his fingertips all information on a global basis, although he will probably choose to reveal to the driver of a particular vehicle only information that is needed by that vehicle and driver.

D.0.4 Processing of information

The two main schemes that exist regarding what is done with the information that is available are the centralized and decentralized ones. According to the former, all information is collected and processed by a central unit (be that a human scheduler, a man-machine computer-assisted dispatching system, or a fully automated system). On the other hand, it may make sense for some of the information to be processed separately. For instance, if the driver of the truck is given latitude to decide his own routing for a certain set of demand points (either by himself, or by an on-board computer), we have a (partly) decentralized system. This attribute is very important with respect to the methodology that is used to solve the routing problem. Some of the partitioning or decomposition schemes that have been used for many vehicle routing problems in the past may lend themselves to processing decentralization. It is our opinion that such schemes are more important for dynamic routing problems, given the faster running time requirement of the dynamic scenario vis- t-vis the static one.

D.0.5 12 Issues that Differ DVRP

Psaraftis [95] also lists 12 issues on which the dynamic vehicle routing problem differs from the conventional static routing problem. Below, is a brief summary of these issues as they are indeed very central to the discussion of static versus dynamic routing.

1. Time dimension is essential. In a static routing problem the time dimension may or may not be important. In the dynamic counterpart time is always essential. The dispatcher must as a minimum know the position of all vehicles at any given point in time and particularly when the request for service or other information is received by the dispatcher.1
2. The problem may be open-ended. The process is often temporally bounded in a static problem. The routes start and end at the depot. In a dynamic setting the process may very well be unbounded. Instead of routes one considers paths for the vehicles to follow.
3. Future information may be imprecise or unknown. In a static problem all information is assumed to be known and of the same quality. In a real-life dynamic routing problem the future is almost never known with certainty. At best probabilistic information about the future may be known.

4. Near-term events are more important. Due to the uniformity of the information quality and lack of input updates all events carry the same weight in a static routing problem. Whereas in a dynamic setting it would be unwise immediately to commit vehicle resources to long-term requirements. The focus of the dispatcher should therefore be on near-term events when dealing with a dynamic routing problem.
5. Information update mechanisms are essential. Almost all inputs to a dynamic routing problem are subject to changes during the day of operation. It is therefore essential that information update mechanisms are integrated into the solution method. Naturally, information update mechanisms are not relevant within a static context.
6. Re-sequencing and reassigning decisions may be warranted. In dynamic routing new input may imply that decisions taken by the dispatcher become suboptimal. This forces the dispatcher to reroute or even reassign vehicles in order to respond to the new situation.
7. Faster computation times are necessary. In static settings the dispatcher may afford the luxury of waiting for a few hours in order to get a high quality solution, in some cases even an optimal one. In dynamic settings this is not possible, because the dispatcher wishes to know the solution to the current problem as soon as possible (preferably within minutes or seconds). The running-time constraint implies that rerouting and reassignments are often done by using local improvement heuristics like insertion and k-interchange.
8. Indefinite deferment mechanisms are essential. Indefinite deferment means the eventuality that the service of a particular demand be postponed indefinitely because of that demands unfavorable geographical characteristics relative to the other demands. This problem could for instance be alleviated by using time window constraints or by using a nonlinear objective function penalizing excessive wait.
9. Objective function may be different. Traditional static objectives such as minimization of the total distance traveled or the overall duration of the schedule might be meaningless in a dynamic setting because the process may be open-ended. If no information about the future inputs is available, it might be reasonable to optimize only over known inputs. Some systems also use nonlinear objective functions in order to avoid undesirable phenomena such as the above mentioned indefinite deferment.
10. Time constraints may be different. Time constraints such as latest pickup times tend to be softer in a dynamic routing problem than in a static one. This is due to the fact that denying service to an immediate demand, if the time constraint is not met, is usually less attractive than violating the time constraint.
11. Flexibility to vary vehicle fleet size is lower. In static settings the time gap between the execution of the algorithm and the execution of the routes usually allows adjustments

of the vehicle fleet. However, within a dynamic setting the dispatcher may not have instant access to backup vehicles. Implications of this may mean that some customers receive lower quality of service.

12. Queueing considerations may become important. If the rate of customer demand exceeds a certain threshold, the system will become congested and the algorithms are bound to produce meaningless results. Although vehicle routing and queueing theory are two very well-studied disciplines, the effort to combine these has been scant.

Annex E

Netlogo

NetLogo is a programmable modeling environment for simulating natural and social phenomena. It was authored by Uri Wilensky in 1999 and has been in continuous development ever since at the Center for Connected Learning and Computer-Based Modeling.

NetLogo is particularly well suited for modeling complex systems developing over time. Modelers can give instructions to hundreds or thousands of "agents" all operating independently. This makes it possible to explore the connection between the micro-level behavior of individuals and the macro-level patterns that emerge from their interaction.

NetLogo lets students open simulations and "play" with them, exploring their behavior under various conditions. It is also an authoring environment which enables students, teachers and curriculum developers to create their own models. NetLogo is simple enough for students and teachers, yet advanced enough to serve as a powerful tool for researchers in many fields.

NetLogo has extensive documentation and tutorials. It also comes with the Models Library, a large collection of pre-written simulations that can be used and modified. These simulations address content areas in the natural and social sciences including biology and medicine, physics and chemistry, mathematics and computer science, and economics and social psychology. Several model-based inquiry curricula using NetLogo are available and more are under development.

NetLogo is the next generation of the series of multi-agent modeling languages including StarLogo and StarLogoT. NetLogo runs on the Java Virtual Machine, so it works on all major platforms (Mac, Windows, Linux, et al). It is run as a desktop application. Command line operation is also supported.

The information about Netlogo can be reached online in : https ://ccl.northwestern.edu/netlogo/docs/

References

[1] M. Affenzeller, S. Wagner, S. Winkler, A. Beham, *Genetic Algorithms and Genetic Programming : Modern Concepts and Practical Applications*. CRC Press, 2009.

[2] M. Alighanbari, L. F. Bertuccelli, J. P. How, *A Robust Approach to the UAV Task Assignment Problem*. 45th IEEE Conference on Decision and Control, San Diego, CA, USA, December 13 - 15, 2006.

[3] M. A. Arostegui, S. N. Kadipasaoglu and B. M. Khumawala, *An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems*. Int. J. Prod. Econ.,103, 742–754, 2006.

[4] A. Aslidis, *Minimization of overstowage in containership operations*. Operational Research '90, vol. 18, pp. 457-471, 1991.

[5] A. Attanasio, J. Bregman, G. Ghiani and E. Manni, *Real-time fleet management at Ecourier Ltd*. In : V. Zeimpekis, C.D. Tarantilis, G.M. Giaglis and I. Minis (Eds.), Dynamic Fleet Management, Operations Research/Computer Science Interfaces, vol. 38, Springer, US, pp. 219–238 (chapter 10), 2007.

[6] H. Awad, R. Elshaer, A. Abdelmoez and G. Nawara, *An Effective Genetic Algorithm for Capacitated Vehicle Routing Problem*. International Conference on Industrial Engineering and Operations Management, Bandung, Indonesia, March 6-8, 2018.

[7] A. Bachrach, A. de Winter, R. He, G. Hemann, S. Prentice, N. Roy, *RANGE - robust autonomous navigation in GPS-denied environments*. 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, May 3 - 7, 2010.

[8] B. M. Baker and M. A. Ayechew, *A genetic algorithm for the vehicle routing problem*. Computers & Operations Research, Vol. 30, Issue 5, Pages 787-800, April 2003.

[9] R. Baldacci, N. Christofides and A. Mingozzi, *An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts*. Mathematical Programming, 115, 351–385, 2008.

[10] J. E. Beasley, *Route-first cluster-second methods for vehicle routing*. Omega, 11 :403–408, 1983.

[11] G. Berbegliaa, J. F. Cordeaub, G. Laporte, *Dynamic pickup and delivery problems.* European Journal of Operational Research, Vol. 202, Issue 1, 1 April 2010, Pages 8-15.

[12] J. Berger and M. Barkaoui, *A new hybrid genetic algorithm for the capacitated vehicle routing problem.* Journal of the Operational Research Society, 54 :1254-1262, 2004.

[13] L. F. Bertuccelli, M. Alighanbari, J. P. How, *Robust Planning For Coupled Cooperative UAV Missions.* IEEE Conference on Decision and Control 3 :2917 - 2922, Vol.3, January 2005.

[14] L. Bianchi, M. Dorigo, L. M. Gambardella and W. J. Gutjahr, *A survey on metaheuristics for stochastic combinatorial optimization.* Natural Computing, 8, 239–287. 2009.

[15] A. S. Bjarnadóttir, *Solving the Vehicle Routing Problem with Genetic Algorithms.* PhD Thesis. Informatics and Mathematical Modelling (IMM), Technical University of Denmark (DTU), 2004.

[16] C. Blum and A. Roli, *Metaheuristics in combinatorial optimization : overview and conceptual comparison.* ACM Computing Surveys, 35, 189–213. 2003.

[17] K. Braekers, A. Caris and G. K. Janssens, *Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots.* Transportation Research Part B : Methodological, 67, 166–186, 2014.

[18] J. Branke, M. Middendorf, G. Noeth and M. Dessouky, *Waiting strategies for dynamic vehicle routing.* Transportation Science 39 (3), 298–312, 2005.

[19] G.G. Brown, C.J. Ellis, G.W. Graves and D. Ronen, *Real-time, wide area dispatching of mobil tank trucks.* Interfaces 17, p 107-120, 1987

[20] A. Bry, A. Bachrach, N. Roy,*State Estimation for Aggressive Flight in GPS-Denied Environments Using Onboard Sensing.* 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, May 14 - 18, 2012.

[21] Y. Chang and L. Chen, *Solve the vehicle routing problem with time windows via a genetic algorithm.* Discrete and Continuous Dynamical Systems. 2007.

[22] C. B. Cheng, and K. P. Wang, *Solving a vehicle routing problem with time windows by a decomposition technique and a genetic algorithm.* Expert Systems with Applications, 36(4), 7758–7763. 2009.

[23] Z. Cheng, Y. Sun and Y. Liu, *Path planning based on immune genetic algorithm for UAV.* International Conference on Electric Information and Control Engineering. 2011.

[24] W. C. Chiang, R. Russell, X. Xu, and D. Zepeda (2009), *A simulation/metaheuristic approach to newspaper production and distribution supply chain problems.* International Journal of Production Economics, vol. 121, no. 2, pp. 752-767, 2009.

[25] A. Chin, H. Kit and A. Lim, *A new GA approach for the vehicle routing problem.* 11th International Conference on Tools with Artificial Intelligence, Chicago, IL, USA, 1999.

[26] G. Clarke and J. W. Wright, *Scheduling of Vehicles from a Central Depot to a Number of Delivery Points*, Operations Research, Vol. 12, 1964, pp. 568-581

[27] J. F. Cordeau, M. Gendreau, A. Hertz, G. Laporte and J. S. Sormany JS, *New Heuristics for the Vehicle Routing Problem.* In : Langevin A., Riopel D. (eds) Logistics Systems : Design and Optimization. Springer, Boston, MA. 2005.

[28] J. F. Cordeau, M. Gendreau, G. Laporte, J. Y. Potvin and F. Semet, *A guide to vehicle routing heuristics.* Journal of the Operational Research Society, 53(5), 512–522. 2002.

[29] T. G. Crainic, P. Dejax and M. Gendreau, *Modeling the container fleet management problem using a stochastic programming approach.* Operational Research '90, vol 18, pp. 473-486, 1991.

[30] G. D. Cubber, H. Balta, D. Doroftei, Y. Baudoin, *UAS deployment and data processing during the Balkans flooding.* IEEE International Symposium on Safety, Security, and Rescue Robotics, Hokkaido, Japan, October 27 - 30, 2014.

[31] G. B. Dantzig and J. H. Ramser, *The Truck Dispatching Problem*, Management Science, Vol. 6, No. 1, pp. 80-91, INFORMS, 1959.

[32] A. I. De Castro, F. M. Jiménez-Brenes, J. Torres-Sánchez, J. M. Peña, I. Borra-Serrano, F. López-Granados, *3-D Characterization of Vineyards Using a Novel UAV Imagery-Based OBIA Procedure for Precision Viticulture Applications.* Remote Sens. 2018, 10(4), 584 ; https ://doi.org/10.3390/rs10040584.

[33] M. Desrochers and G. Laporte, *Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints.* Operations Research Letters, 10, 27–36, 1991.

[34] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis, *Time Constrained Routing and Scheduling.* Handbooks in Operations Research and Management Science, vol 8, pages 35-140, 1995.

[35] M. Dorigo and T. Stützle, *Ant Colony Optimization.* The MIT Press, Cambridge, MA, 2004.

[36] J. Dréo, *Classification of metaheuristics.* [Online]. Available at : http ://metah.nojhan.net/post/2007/10/12/Classification-of-metaheuristics. 2007.

[37] M. Dror, M. Ball and B. Golden, *A computational comparison of algorithms for the inventory routing problem.* Annals of Operations Research 4, p 3-23, 1985.

[38] G. Dueck, *New optimization heuristics : the great deluge algorithm and the record-to-record travel.* Journal of Computational Physics, 104 :86–92, 1993.

[39] G. Dueck and T. Scheuer, *Threshold Accepting : a general purpose optimization algorithm.* Journal of Computational Physics, 90 :161–175, 1990

[40] T. A. El-mihoub, A. A. Hopgood, L. Nolle and A. Battersby, *Hybrid Genetic Algorithms : A Review*. Engineering Letters. 3(2). 2006.

[41] N. A. El Sherbeny, *Vehicle routing with time windows : An overview of exact, heuristic and metaheuristic methods*, Journal of King Saud University (Science), 22, 123–131, 2010.

[42] M. L. Fisher, *Network Routing*, Handbooks in Operations Research and Management Science, vol 8, pages 1-33, 1995.

[43] M. L. Fisher, J. Huang and B. X. Tang, *Scheduling bulk pickup-delivery vehicles in Shanghai*. Interfaces 16, p 18-23, 1986.

[44] M. L. Fisher and R. Jaikumar, *A generalized assignment heuristic for the vehicle routing problem*. Networks, 11 :109–124, 1981.

[45] M. Flood, *The Traveling Salesman Problem*. Operations Research, 4(1) :61–75. 1956.

[46] Z. Fu and M. Wright, *Train plan model for British rail freight services through the channel tunnel*. The Journal of the Operational Research Society, vol. 45, no. 4, pp. 384-391, 1994.

[47] R. Fukasawa, H. Longo, J. Lysgaard, M. P. de Aragão, M. Reis, E. Uchoa and R. F. Werneck, *Robust branch-and-cut-and-price for the capacitated vehicle routing problem*. Mathematical programming, 106, 491–511, 2006.

[48] M. Garmilla, *Computer-aided routing of combination carriers*. MSc Thesis, Department of Ocean Engineering, MIT, 1988.

[49] S. Geetha and N. Vijayalakshmi, *A Survey on Genetic Algorithm for Vehicle Routing Problem*. International Journal of Advanced Research in Computer and Communication Engineering. Vol. 5, Issue 2, February 2016.

[50] M. Gen and A. Syarif, *Hybrid genetic algorithm for multi-time period production/distribution planning*. Computers and Industrial Engineering 48, 799–809. 2005.

[51] M. Gendreau, J. Y. Potvin, O. Bräysy, G. Hasle, A. Lokketangen, *Metaheuristics for the Vehicle Routing Problem and Its Extensions : A Categorized Bibliography*, In : Golden B., Raghavan S., Wasil E. (eds) The Vehicle Routing Problem : Latest Advances and New Challenges. Operations Research/Computer Science Interfaces, vol 43. Springer, Boston, MA, 2008.

[52] B. E. Gillett and L. R. Miller, *A heuristic algorithm for the vehicle dispatch problem*. Operations Research, 22 :340–349, 1974.

[53] F. Glover and M. Laguna, *Tabu Search*. Kluwer, Boston, MA, 1997

[54] F. Glover, P. Kelly and M. Laguna, *Genetic algorithms and Tabu search : hybrids for optimizations*. Computer Operation Research 22 (1), 111–134. 1995.

[55] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York, 1989.

[56] B. L. Golden, S. Raghavan and E. A. Wasil, *The vehicle routing problem : latest advances and new challenges*. Springer Science & Business Media, vol. 43. , 2008.

[57] J. L. Guo and S. W. Yu, *A Hybrid Algorithm for Open Vehicle Routing Optimization of Coal Mines Material*. Applied Mechanics and Materials, Vol. 197, pp. 455-461, 2012.

[58] P. Hansen and N. Mladenović, *Variable Neighborhood Search*. Chapter 6 in Handbook of Metaheuristics, F. Glover and G.A. Kochenberger, eds., Kluwer, 145–184, 2003

[59] W. Hileman, *Netlogo User Community Models, Classic Travelling Salesman*. [Online]. Available : http ://ccl.northwestern.edu/netlogo/models/community. 2011

[60] W. Ho, G. T. S. Ho, P. Ji and H. C. W. Lau, *A hybrid genetic algorithm for the multi-depot vehicle routing problem*. Engineering Applications of Artificial Intelligence, 21, 548–557. 2008.

[61] W. Ho and P. Ji, *A hybrid genetic algorithm for component sequencing and feeder arrangement*. Journal of Intelligent Manufacturing 15, 307–315. 2004.

[62] H. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI. 1975.

[63] S. Ichoua, M. Gendreau and J.-Y. Potvin, *Exploiting knowledge about future demands for real-time vehicle dispatching*. Transportation Science 40 (2), 211–225, 2006

[64] Z. Ismail and I. Irhamah, *Solving the Vehicle Routing Problem with Stochastic Demands via Hybrid Genetic Algorithm-Tabu Search*. Journal of Mathematics and Statistics, 4(3), 2008.

[65] R. R. Iwańkowicz and Z. Sekulski, *Solving The Problem Of Vehicle Routing By Evolutionary Algorithm*. Advances in Science and Technology Research Journal, Volume 10, No. 29, pages 97–108, March 2016.

[66] E. S. Jones, S. Soatto, *Visual-Inertial Navigation, Mapping and Localization : A Scalable Real-Time Causal Approach*. The International Journal of Robotics Research, IJRR, Vol 30, Issue 4, 2011.

[67] C. Kamboj, M. Singh and D. Rana, *Hybridization of Heuristic Genetic Algorithm : A Survey*. International Research Publication House, Volume 7, Number 1, pp. 43-5, 2014.

[68] K. Karakatic, and V. Podgorelec, *A survey of genetic algorithms for solving multi depot vehicle routing problem*. Applied Soft Computing, 27, 519–532. 2015.

[69] E. Keedwell, S. T. Khu, *A hybrid genetic algorithm for the design of water distribution networks*. Engineering Applications of Artificial Intelligence 18, 461–472. 2005.

[70] J. Kelly, G. S. Sukhatme, *Visual-Inertial Simultaneous Localization, Mapping and Sensor-to-Sensor Self-Calibration*. 2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation - (CIRA), Daejeon, South Korea, December 15 - 18, 2009.

[71] S. N. Kumar and R. Panneerselvam, *A Survey on the Vehicle Routing Problem and Its Variants*, Intelligent Information Management, Vol. 4, pp. 66-74, 2012.

[72] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing : Theory and Applications*. Springer, Dordrecht, 1987.

[73] G. Laporte, *The Vehicle Routing Problem : An overview of exact and approximate algorithms*, European Journal of Operational Research, Vol. 59, pp. 345-358, 1992.

[74] G. Laporte and F. Semet, Classical heuristics for the capacitated VRP. In : P. Toth and D. Vigo (eds), The Vehicle Routing Problem, pages 109–128. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia. 2002.

[75] A. Larsen, O. B. G. Madsen, M. M. Solomon, *Classification of Dynamic Vehicle Routing Systems*. Dynamic Fleet Management, 19–40, 2007.

[76] A. Larsen, *The Dynamic Vehicle Routing Problem*. Ph.D Thesis, IMM Dept. University od Denmark (DTU), 2000.

[77] L. Y. O. Li and Z. Fu, *The school bus routing problem : a case study*. Journal of the Operational Research Society, 53(5), 552-558, 2002.

[78] Y. Liu, *Optimization of Vehicle Routing Problem for Field Service*. Ph.D Thesis, Ecole Centrale de Lille, 2017.

[79] V. A. Mabert, A. V. Hill and D. W. Montgomery, *A decision support system for the courier vehicle scheduling problem*. Omega 16, p 333-345, 1988.

[80] A. K. M. Masum, M. Shahjalal, M. F. Faruque and M. I. H. Sarker, *Solving the Vehicle Routing Problem using Genetic Algorithm*. International Journal of Advanced Computer Science and Applications 2(7), 2011.

[81] D. Mellinger and V. Kumar, *Minimum Snap Trajectory Generation and Control for Quadrotors*. 2011 IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 9 - 13, 2011.

[82] L. Miao, Q. Ruan, K. Woghiren and Q. Ruo, *A hybrid genetic algorithm for the vehicle routing problem with three-dimensional loading constraints*. RAIRO - Operations Research - Recherche Opérationnelle, Volume 46, no. 1, p. 63-82, 2012.

[83] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs*. 3rd, revised and extended edition. Springer-Verlag, 1996.

[84] H. Min, *The multiple vehicle routing problem with simultaneous delivery and pickup points*. Transportation Research 23A, p 377-386, 1989.

[85] H. Min, *International intermodal choices via chance-constrained programming*. Transportation Research 25A, p 351-362, 1991.

[86] S. H. Min, J. Lee, and I. Han, *Hybrid genetic algorithms and support vector machines for bankruptcy prediction*. Expert Systems with Applications 31, 652–660. 2006.

[87] Netlogo Home Page. [Online]. Available at : https ://ccl.northwestern.edu/netlogo/references.shtml.

[88] M. Okulewicz and J. Mandziuk, *Application of Particle Swarm Optimization Algorithm to Dynamic Vehicle Routing Problem.* 12th International Conference of Artificial Intelligence and Soft Computing, ICAISC, Zakopane, Poland, June 9-13, 2013.

[89] H. Pan and W. Li, *Hybrid Genetic-Saving Algorithm and Its Application in Vehicle Routing Problem*, International Conference on Management and Service Science, Wuhan, China, Sept 20 - 22, 2009.

[90] J. Park and B.-I. Kim, *The school bus routing : A review.* European Journal of Operational Research, 202(2), 311-319, 2010.

[91] M. Pavone, *Dynamic Vehicle Routing for Robotic Networks.* Ph.D Thesis, Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2010.

[92] T. Pichpibul and R. Kawtummachai, *A Heuristic Approach Based on Clarke-Wright Algorithm for Open Vehicle Routing Problem.* The Scientifc World Journal, Vol. 2013, Article ID 874349, 2013.

[93] V. Pillac, M. Gendreau, C. Guéret, A. L. Medaglia, *A Review of Dynamic Vehicle Routing Problems.* European Journal of Operational Research. Vol. 225, Issue 1, p 1-11, 2013.

[94] V. Pillac, C. Guéret and A. Medaglia. *Dynamic Vehicle Routing Problems : State of the art and Prospects.* Technical Report 10/4/AUTO. 2011

[95] H. N. Psaraftis, *Dynamic vehicle routing : Status and prospects.* Annals of Operations Research, Vol. 61, Issue 1, pp 143–164, 1995.

[96] H. Psaraftis, *A dynamic-programming solution to the single vehicle many-tomany immediate request dial-a-ride problem.* Transportation Science 14, 2, 130–154, 1980.

[97] H. N. Psaraftis, *Vehicle Routing : Methods and Studies, chapter Dynamic Vehicle Routing Problems.* pages 223-248. Elsevier Science Publishers B.V. (North Holland), 1988.

[98] V. Pureza and G. Laporte, *Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows.* INFOR 46 (3), 165–175, 2008.

[99] L. G. Reagan, P. D. Elmer, A. A. Bandala, *Path Planning for Quadrotor UAV using Genetic Algorithm*, 7th IEEE International Conference Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM) 2014, Philippines, 2014.

[100] P. P. Repoussis, D. C. Paraskevopoulos, G. Zobolas, C. D. Tarantilis, and G. Ioannou, *A web-based decision support system for waste lube oils collection and recycling.* European Journal of Operational Research, vol. 195, no. 3, pp. 676-700, 2009.

[101] M. G. C. Resende and C. C. Ribeiro, *Greedy Randomized Adaptive Search Procedures.* Chapter 8 in Handbook of Metaheuristics, F. Glover and G.A. Kochenberger, eds., Kluwer, 219–249, 2003.

[102] D. Ronen, *Short term scheduling of vessels for shipping bulk or semi-bulk commodities originating in a single area.* Operations Research 34, p 164-173, 1986.

[103] D. J. Rosenkrantz, R. E. Stearns and P. M. Lewis, *An analysis of several heuristics for the traveling salesman problem.* SIAM journal on computing, 6, 563-581, 1977.

[104] R. Russell, W. C. Chiang, and D. Zepeda, *Integrating multiproduct production and distribution in newspaper logistics.* Computers & Operations Research, vol. 35, no. 5, pp. 1576-1588, 2008.

[105] I. Sabuncuoglu and D. Hommertzheim, *Dynamic dispatching algorithm for scheduling machines and sutomated guided vehicles in a flexible manufacturing system.* International Journal of Production Research 30, p 1059-1079, 1992.

[106] M. N. M. SAP, *Sweep algorithm in vehicle routing problem for public transport.* Jurnal Antarabangsa (Teknologi Maklumat), 2, 51–64, 2002.

[107] B. Sarasola, K. F. Doerner, V. Schmid, and E. Alba, *Variable neighborhood search for the stochastic and dynamic vehicle routing problem.* Annals of Operations Research, no. 236, pp. 425 – 461, 2016.

[108] D. Sariklis and S. Powell, *A heuristic method for the open vehicle routing problem.* The Journal of the Operational Research Society, vol. 51, no. 5, pp. 564-573, 2000.

[109] K. Schopka and H. Kopfer, *An Adaptive Large Neighborhood Search for the Reverse Open Vehicle Routing Problem with Time Windows.* Logistics Management : Contributions of the Section Logistics of the German Academic Association for Business Research, Braunschweig, Germany (pp.243-257), 2015.

[110] M. Schyns, *An ant colony system for responsive dynamic vehicle routing.* European Journal of Operational Research, vol. 245, no. 3, pp. 704 – 718, 2015.

[111] A. Sonmez, E. Kocyigit and E. Kugu, *Optimal Path Planning for UAVs Using Genetic Algorithm*, International Conference on Unmanned Aircraft System (ICUAS), Denver, Colorado, USA, June 9 - 12, 2015.

[112] V. Speidel, *EDP-assisted fleet scheduling in tramp and coastal shipping.* In : Ship Operation Automation, eds. Pitkin, Roche and Williams (North-Holland), 1976.

[113] SwRI Home Page. [online]. Available at : https ://www.swri.org/press-release/swri-led-team-develop-drones-use-fukushima-daiichi-nuclear-power-plant.

[114] S. Tang and V. Kumar, *A Complete Algorithm for Generating Safe Trajectories for Multi-Robot Teams.* Robotics Research, pp 599-616, 2018.

[115] C. D. Tarantilis and C. T. Kiranoudis, *Distribution of fresh meat.* Journal of Food Engineering, vol. 51, no. 1, pp. 85-91, 2002.

[116] B.W. Thomas, *Waiting strategies for anticipating service requests from known customer locations.* Transportation Science 41 (3), 319–331, 2007.

[117] P. M. Thompson and H. N. Psaraftis, *Cyclic transfer algorithms for multivehicle routing and scheduling problems*, Operations Research 41, p 935-946, 1993.

[118] S. Tisue and U. Wilensky, *Netlogo : A Simple Environment for Modelling Complexity*, presented at the International Conference on Complex Systems (ICCS), Boston, May 16 - 21, 2004.

[119] P. Toth and D. Vigo, *Vehicle routing : problems, methods, and applications.* SIAM, 2014.

[120] P. Trudeau and M. Dror, *Stochastic inventory routing : Route design with stockouts and route failures.* Transportation Science 26, p 171-184, 1992.

[121] M. Turpin, N. Michael, V. Kumar, *Trajectory Planning and Assignment in Multirobot Systems.* Algorithmic Foundations of Robotics, X, pp 175 - 190. 2013.

[122] M. Turpin, N. Michael, V. Kumar, *CAPT : Concurrent assignment and planning of trajectories for multiple robots.* The International Journal of Robotics Research, IJRR, Vol 33(1), 98 – 112, 2014.

[123] A. J. Umbarkar and P. D. Sheth, *Crossover Operators in Genetic Algorithms : A Review*, ICTACT Journal on Soft Computing, Vol. 06, Issue : 01, October, 2015.

[124] N. Wilson and N. Colvin, *Computer control of the Rochester dial-a-ride system.* Technical Report Report R77-31, Dept. of Civil Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts. 1977.

[125] S. Yu, C. Ding, and K. Zhu, *A hybrid GA-TS algorithm for open vehicle routing optimization of coal mines material.* Expert Systems with Applications, vol. 38, no. 8, pp. 10568-10573, 2011.

[126] H. Youssef, S. M. Sait and H. Adiche, *Evolutionary algorithms, simulated annealing and tabu search : a comparative study.* Eng. Appl. Artif. Intell., 14, 167–181, 2001.

[127] Y. Zhang, J. Liu, F. Duan and J. Ren, *Genetic Algorithm in Vehicle Routing Problem.* Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2007.