# AN ABSTRACT OF THE DISSERTATION OF

Kyoung Cheol Kim for the degree of Doctor of Philosophy in Industrial Engineering presented on May 13, 2011.
Title: Vehicle Routing and Scheduling with Delivery and Installation

Abstract approved: _____

Sundar Atre

For manufacturing and service industries to stay competitive in this rapidly changing, globalized world, one of the most important operations, and one that many businesses struggle to plan and manage efficiently, is material handling. There has been a large amount of research on various vehicle routing problems (VRPs) in recent decades, but relatively less work on certain unique types of VRPs that characterize some specific industries.

This thesis identifies an interesting and new type of VRP with unique characteristics in order to serve the needs and goals of the electronics industry. The problem deals with two types of customers; some require only delivery, while the others require both delivery and installation. There are two different types of vehicles in this problem: delivery vehicles and installation vehicles. The delivery vehicle and (if needed) the installation vehicle are allowed to visit each customer only once. There is an additional constraint to provide the guaranteed service quality, which is measured by the amount of time between the delivery and the installation. A customer must be visited by an installation vehicle within the predetermined maximum allowable time

after the visit by a delivery vehicle. Therefore, it is required that both types of vehicles be synchronized for the customers requiring both delivery and installation. The problem under consideration is more complicated than other, traditional VRPs that have been studied widely in the past, since in this case two different types of vehicles must be synchronized.

A mixed-integer nonlinear programming (MINP) model for this problem is formulated. A hierarchical approach using a genetic algorithm (GA) is proposed to solve the problem effectively. Various examples are tested to show the effectiveness of the proposed hierarchical approach. To demonstrate the robustness of the proposed approach, partial factorial experimental design varying the parameters of characteristics in the problem is performed using the Taguchi method. In the hierarchical approach, an algorithmic limitation is conjectured in which the attempt to find the best solution tends to dwell on the local optimal solution, searching only part of the entire solution space. In order to tackle the limitation of the hierarchical approach and search the entire solution space effectively and efficiently for a global optimal solution, an endosymbiotic evolutionary algorithm (EEA), which concurrently considers subproblems having cooperative interactions, is considered. Various test examples are solved using the EEA, and this method's efficiency and effectiveness are shown by comparing the computational results with the ones from the hierarchical approach. A set of problems is solved by the MINP model, the hierarchical approach using a genetic algorithm, and the EEA, and solutions from the three approaches are compared.

# Vehicle Routing and Scheduling with Delivery and Installation

By

Kyoung Cheol Kim

A DISSERTATION

Submitted to

Oregon State University

In partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented May 13, 2011
Commencement June 2011

Doctor of Philosophy dissertation of Kyoung Cheol Kim presented on May 13, 2011

APPROVED:

---

Major Professor, representing Industrial Engineering

---

Head of the School of Mechanical, Industrial, & Manufacturing Engineering

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

---

Kyoung Cheol Kim, Author

# ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Dr. Sundar Atre, for his excellent guidance and concern, and for providing me with an excellent atmosphere for the completion of my research at Oregon State University. I am also deeply thankful to my former advisor, Dr. Shiwoo Lee, whose continuous encouragement, from the preliminary stage to the conclusion of the project, enabled me to develop an understanding of the subject.

My gratitude extends to my committee members, Dr. Kenneth H. Funk and Dr. Thinh Nguyen for their constant encouragement and useful feedback, which have been of the greatest help at all times. Additionally, I am thankful to Dr. J. Antonio Torres for serving as the Graduate Council Representative (GCR) for a while. I also thank Dr. Lech Muszynski for being my GCR for the final defense.

I thank my close friends Sejoon Park, Hyunpae Lim, Juthamas Choomlucksana, Behrouz Behmardi, and Weerakit Denkaew for their constant support and encouragement.

Finally, I would like to thank my parents, Kiseong and Soonja Kim, for their help and support throughout my life. Special gratitude is due also to my brothers, Kyounghoon and KyoungJoon Kim, and their families for their loving support. I would like to extend my appreciation to Peter and June Nam for their encouragement. I would also like to thank my parents in law, Jongkyu Kim and Junghee Ha. They were always supporting me and encouraging me with their best wishes. I would like to dedicate this thesis to my wife, Haengim, and my daughter, Michelle (Jeongsun), for their love, patience, and understanding as they allowed me to spend most of my time on this thesis.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

Table Page

LIST OF APPENDIX FIGURES

Table Page

LIST OF APPENDIX TABLES

Table Page

LIST OF APPENDIX TABLES (Continued)

Table Page

# 1. INTRODUCTION

In order to survive in this competitive business environment, a company must have an appropriate way to handle materials cost-effectively. Especially in the manufacturing industry, handling methods for raw materials and work-in-process are as important as the one for final products. For material handling activities to satisfy various demands effectively, vehicle routing and scheduling has been studied and implemented extensively. In this thesis, a vehicle routing problem (VRP) found in the electronics industry, which has unique characteristics of material handling, is considered. The problem under consideration and the objectives of the thesis are described in the following sections.

## 1.1 Problem description

The electronics industry has recently experienced rapidly emerging changes in its post-sales service, i.e., delivery and installation. In the past, local stores were individually responsible for these services. However, nowadays electronics manufacturers are increasingly required to directly deliver products to their customers and to provide on-site installation. Electronics sales via e-commerce, large discount stores, general merchandise stores, and department stores are increasing very rapidly while sales via existing local stores are decreasing. Moreover, electronics manufacturers are putting intensive efforts into increasing sales through professional electronics franchises like Staples (US), OfficeMax (US), Hi-Mart (Korea), and other such stores, which do not provide delivery and installation themselves. These trends tend to push greater responsibilities for delivery and installation onto electronics manufacturers, and the number of direct deliveries from electronics manufacturers to customers continues to increase at an explosive pace.

Another unique characteristic of the electronics industry has to do with the installation service itself. Some products, such as air conditioners, have long required professional installation services. Many other new products similarly require both delivery and professional installation; these products include wall-mounted televisions, home theaters, washers and dryers, refrigerators with a water purifier, special cook-tops, numerical control machines, and computer servers.

Another organizational need makes the task of planning vehicle routing and scheduling more complicated. The expense to maintain a nationwide distribution and service network is too high to make it practical and economical. Therefore, manufacturers adopt the practice of outsourcing the delivery to third parties while maintaining their own service teams or commissioning authorized service providers for the installation.

The VRP under consideration assumes that there exist two types of demands in a complex electronics market: one requires the delivery only, and the other requires both delivery and installation. To satisfy both types of demands, a single distribution center separately operates two different types of vehicles (delivery and installation vehicles). It is assumed that delivery vehicles have a limited loading capacity to carry the products and that installation vehicles do not. Both types of vehicles start from a single depot at the beginning and return to the depot within a specified time. Delivery demands of all customers are known in advance. Based on the delivery demands, a set of customers is assigned to a delivery vehicle. The sum of customers' demands assigned to a single delivery vehicle cannot exceed the loading capacity of the delivery vehicle. The delivery vehicle and (if needed) the installation vehicle are permitted to visit each customer only once. In addition, there is a constraint to satisfy the expected service quality, which is measured as the amount of time between the delivery and the installation. It is necessary for the installation vehicle to visit a customer within the

predetermined maximum allowable time after the delivery vehicle's visit to that customer. Therefore, the synchronization of both types of vehicles needs to be carefully planned to guarantee the promised service quality. Figure 1.1 shows a typical example of the VRP under consideration and its potential solution.

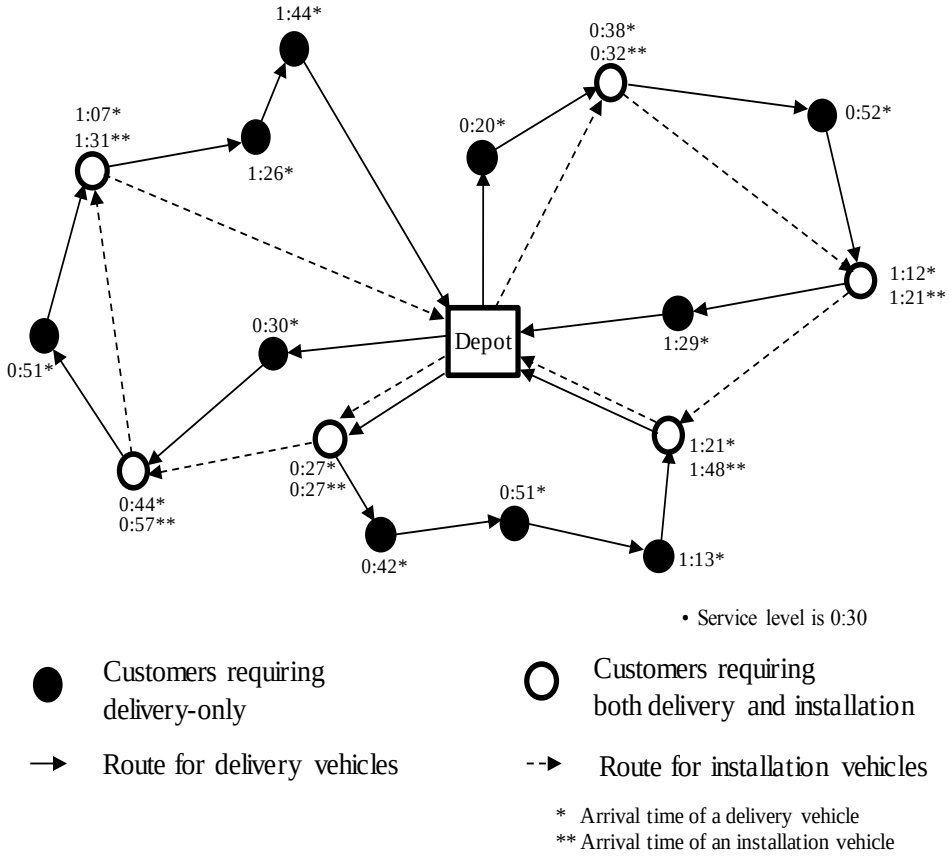



Figure 1.1 An example of vehicle routing problems under consideration.

The example consists of 16 customers, 10 requiring the delivery only (which are represented by the filled circles) and 6 requiring both delivery and installation (which are represented by the open circles). The solid and dotted lines are the routes of the delivery and installation vehicles, respectively. There are three delivery and two installation vehicles in Figure 1.1. The arrival times of delivery and installation vehicles are shown next to the customers. Installation

vehicles can visit customers earlier than delivery vehicles, causing waiting times for installation vehicles at the corresponding customer locations. If installation vehicles visit customers later than delivery vehicles, the guaranteed maximum time lapse between delivery and installation must be satisfied.

Traditional VRPs are defined as combinatorial optimization problems, for which it is difficult or impossible to obtain optimal solutions through general optimization methods owing to their high computational complexity. The VRP under consideration in this thesis is by far more complicated than other, traditional VRPs studied widely in the past, since it deals with more decision variables and constraints, and the nonlinearity from synchronization requirement adds another level of complexity.

In order to solve the VRP under consideration, three different approaches are introduced in this thesis. First, a mathematical model for the VRP is formulated and used to solve test problems, using commercially available optimization software. Second, a hierarchical approach using a genetic algorithm is proposed to obtain good solutions efficiently in a reasonable amount of time. The hierarchical approach divides the VRP of interest into two subproblems: the VRP for delivery vehicles in the first stage and the VRP for installation vehicles in the second stage. Vehicle routes and schedules for delivery vehicles are determined in the first stage; then, based on the results of the first stage, vehicle routes and schedules for installation vehicles are determined in the second stage. Various test problems are solved to demonstrate the effectiveness of the hierarchical approach. The performance of the hierarchical approach is subject to problem characteristics, such as the number of installation customers, predetermined service quality, installation time, and fixed cost per vehicle. The implications of these characteristics for the problem are studied through the Taguchi method, known as one of the more robust design tools. The purpose of the Taguchi method is to show

the robustness of the algorithm for the problem with various conditions of the characteristics, instead of identifying the optimized process parameters of the proposed algorithm.

The hierarchical approach has the natural limitation that the solutions in the second stage rely on the quality of the solution obtained in the first stage. Since the hierarchical approach may not be able to search the solution space of the problem thoroughly, the solutions generated by this approach can be local optima. It is thus necessary to develop a method to search the entire solution space effectively to find global optima for the corresponding problem. Instead of solving the problems hierarchically, methods considering two subproblems at the same time have been studied to get over the limitation of the hierarchical approach. Therefore, finally, an endosymbiotic evolutionary algorithm (EEA), which concurrently searches partial solution spaces for subproblems of the original complex problem, is also proposed to tackle this limitation. Various test problems are solved using the proposed EEA to demonstrate its effectiveness and efficiency. The computational solutions generated by the aforementioned approaches are given to compare the performance of each approach.

## 1.2 Objectives of the thesis

The first objective of this thesis is to define a unique and interesting vehicle routing problem, found recently in the electronics industry, which requires synchronization of two types of vehicles for delivery and installation. The problem under consideration can be divided into two subproblems, and each subproblem can be modeled as an existing VRP. Distinct characteristics for each subproblem are observed and defined, and the characteristics bridging the two subproblems are also defined.

The second objective is to develop and validate a mathematical model to solve the VRP under consideration with optimality. A mathematical model is formulated as a mixed-integer nonlinear programming (MINP) model. The mathematical model can be used to find optimal solutions using commercially available software, but it has been found extremely hard to determine optimal solutions in a reasonable amount of time. The third objective is to develop a hierarchical approach using a genetic algorithm (GA) to effectively solve VRPs of various sizes. GAs for the delivery and installation, respectively, are developed to solve the subproblems in a reasonable amount of time. The fourth objective is to develop a symbiotic evolutionary algorithm to further improve the performance of the hierarchical approach in terms of speed or solution quality and to compare its performance with the mathematical model and the hierarchical approach. An endosymbiotic evolutionary algorithm (EEA) is developed to achieve this objective.

## 1.3 Organization of the thesis

This thesis consists of eight chapters. Chapter 1 introduces the problem of interest. Chapter 2 reviews the literature on existing VRPs, their solution methods (especially GAs), the Taguchi method, and the EEA. Since the genetic algorithm is used in the proposed hierarchical approach, the procedures and process parameters of GAs are described in detail. In Chapter 3, a mathematical model of the VRP under consideration is formulated and the NP-hardness of the problem is described. Chapter 4 proposes a solution methodology of the hierarchical approach using a genetic algorithm for the VRP under consideration. Computational results by the hierarchical approach using the genetic algorithm are summarized in Chapter 5. Chapter 6 proposes an endosymbiotic evolutionary algorithm as a novel approach to the VRPs under

consideration. The results of various computational experiments by the EEA are provided in Chapter 7. Finally, chapter 8 concludes the dissertation with a discussion of the results and implications, and introduces ideas for future research.

# 2. LITERATURE REVIEW

## 2.1 Vehicle routing problems (VRPs)

### 2.1.1 Description of the traditional VRP

The vehicle routing problem is one of the most important problems in the fields of transportation and logistics. However, it is hard to solve this problem since the VRP belongs to the category of NP-hard combinatorial optimization problems. The VRP was originally introduced by Dantzig and Ramser [1959] and has been widely studied since. Dantzig and Ramser described a dispatching problem of gasoline trucks and proposed a mathematical model and an algorithmic approach. Fisher [1994] described an extended problem in which a vehicle has a series of stops to deliver products to customers. Every customer is assigned to exactly one vehicle in a specific order. The capacity of vehicles is also considered to minimize the total cost. The traditional VRP consists of a set of customers with known demands at predetermined locations and a set of vehicles with a homogeneous capacity. The vehicles start from and return to a single depot. The VRP is to serve all customers without any vehicle being overloaded, while minimizing the total traveling distance. The traveling distance can be easily converted to the traveling time or cost.

Figure 2.1 shows an example of traditional VRPs consisting of three vehicles, nine customers, and a single depot. In Figure 2.1, node 0 in the box denotes the depot, nodes 1 to 9 in circles indicate the customers, and the arrows represent vehicle routes for deliveries to customers.

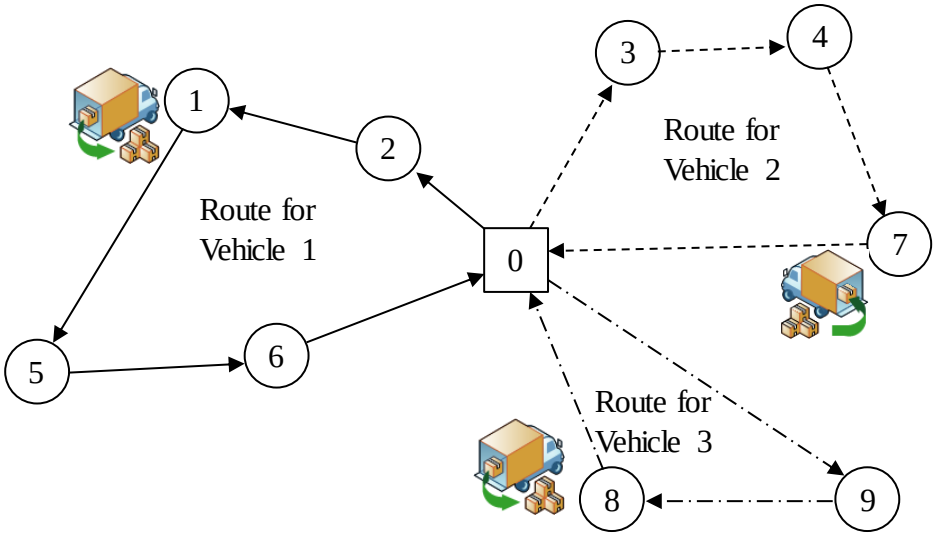



Figure 2.1 An example of a traditional VRP with three vehicles, nine customers, and a single depot.

### 2.1.2 Variations of traditional VRPs

The basic concept of the VRP is that a fleet of vehicles delivers products from a single depot to customers. From this simple concept, variations have emerged for decades. Each variation has its own additional constraints or requirements. Some well-known VRPs are explained by Toth and Vigo [2002]:

**Capacitated vehicle routing problem (CVRP):** The CVRP is the simplest and the most studied problem. In the CVRP, all customers have known demands and known locations for the delivery. The delivery for a customer cannot be split. In other words, the demand of a customer must be satisfied via only one visit. All vehicles are assumed to have the same loading capacity. They depart from a single depot at the beginning and return to the depot at the end. The service or unloading time at each customer may or may not be considered. The objective is to minimize the total traveling distance or time for all vehicles to serve all customers.

**Distance-constrained vehicle routing problem (DVRP):** The DVRP is a variant of the CVRP. Each route of a vehicle is constrained by a maximum length of distance or time. Because of the distance constraint, the total traveling distance in each route cannot exceed the maximum prescribed length.

**Vehicle routing problem with time windows (VRPTW):** The VRPTW is another variant of the CVRP. In the VRPTW, the distance constraint may or may not be considered. Each customer has a time interval, referred to as a time window. The visit of a vehicle to a customer must occur within his or her time window. In case of early arrival at a customer's location, the vehicle is allowed to wait until the beginning of the customer's time window. The time windows are defined by assuming that all vehicles start from a depot at the beginning.

**Vehicle routing problem with pickup and delivery (VRPPD):** In a VRPPD, vehicles are required not only to deliver products to a set of delivery locations, but also to pick goods or wastes up at a set of pickup locations. Unlike other VRPs, products to be delivered are not provided at the depot; rather, they must be picked up. For multiple pickups, the loading capacity of a vehicle must be considered in the problem. Time windows for the pickup and the delivery at each location may or may not be considered in the problem.

Even if a significant amount of research has been done in the area of VRPs, the application of their results to actual practice has been confronted with many difficulties due to the limitations of the simplified VRP models. Therefore, there have been efforts to understand the real-world

constraints and requirements that accompany to specific applications. This research has not been limited to a single basic type of the VRPs introduced earlier; rather, these projects have tended to contain characteristics of multiple traditional VRP models. For example, Prive et al. [2006] suggested a VRP for the distribution of soft drinks and the collection of recyclable containers. They considered the heterogeneous vehicle fleet, vehicle capacity, time windows, pickup, and delivery. Hence, the corresponding VRP is a combination of the CVRP, VRPTW, and VRPPD. Another example is a VRP for school buses. Ripplinger [2005] proposed a rural school VRP. The school bus has a limited number of seats, which is a characteristic of the CVRP. Each school bus must deliver students within a specific time; this is a characteristic of the DVRP. In addition, the school buses do not need to return to school. The route for returning to the original location (the depot) is not important in this example. Unlike other traditional VRPs, the route of each school bus in this VRP does not make a closed loop but, rather, a Hamiltonian loop. Thus, this type of VRPs is called an open vehicle routing problem (OVPR) [Repoussis et al., 2007], and the path is a Hamiltonian path. An example of routes in an OVRP is illustrated in Figure 2.2.

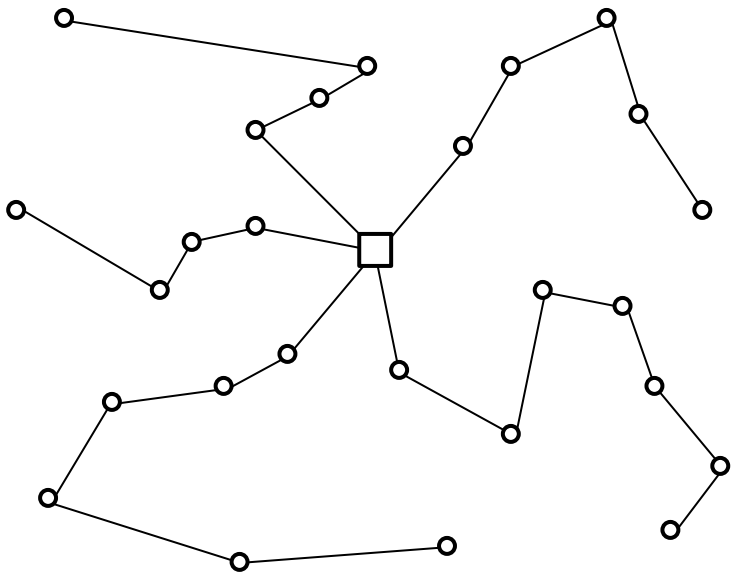


Figure 2.2 An example of an open vehicle routing problem.

### 2.1.3 Solution methods for VRPs

The VRP is one of the well-known and most studied combinatorial optimization problems in academia and industry. For given VRPs, we need to determine the optimal set of routes to be completed by a fleet of vehicles to serve a set of customers. In real life, a VRP can contain many complications such as asymmetric distances, stochastic distance, multiple depots, heterogeneous vehicles, different time windows of customers, and so on. These potential complications make the problem more intractable to solve.

Since Dantzig and Ramser [1959] proposed a heuristic algorithm for VRPs, more than hundreds of papers have been published. Various mathematical models and solution methods have been proposed by Mole [1979], Laporte [1992], Desrosiers et al. [1995], Homberger and Gehring [2005], Krumke et al. [2008], and Eksioglu et al. [2009]. Ford-Bellman-Moore's algorithm was used to solve the multiple traveling salesman problems (TSPs) with time windows by Solomon and Desrosiers [1988]. The problem has also been extended to a vehicle routing problem with time windows (VRPTW). Dynamic programming models are suggested for VRPTWs by Christofides et al. [1981], Baker [1983], and Dumas et al. [1995]. The set partitioning method with a relaxation of linear programming is used to solve a VRPTW by Desrochers et al. [1992].

The VRP is a NP-hard problem, which is hard to solve in a polynomial time [Bodin et al., 1983]. No optimal algorithm that can solve NP-hard problems in a polynomial time has been found [Falkenauer, 1996]. Finding optimal solutions of NP-hard problems is usually very time-consuming and sometimes even impossible. Due to this characteristic, it is not realistic to use optimal solution methods to solve large problems. As an optimal solution method, the branch-and-bound method has been applied to problems with a small number of customers

[Pereira et al., 2002], but for problems of large size the computational limitation of memory buffers and computing resources exists. Hence, many other approaches based on heuristics, approximation algorithms that aim at finding good feasible solutions quickly, have been introduced [Laporte et al., 2000; Prescott-Gagnon et al., 2009].

Many models and algorithms have been proposed to find the optimal solution or near-optimal solutions of different types of VRPs. A thorough classification of VRPs was introduced by Desrochers et al. [1990]. Laporte and Novert [1987] presented an extensive survey that was entirely devoted to exact methods for VRPs. Other surveys were reported by Christofides et al. [1979], Magnanti [1981], Bodin et al. [1983], Christofides [1985], Laporte [1992], Golden et al. [1995], Fisher [1995], Toth and Vigo [1998], and Golden et al. [1998]. They can be broadly divided into two kinds: *classical heuristics* (mostly between 1960 and 1990) and *metaheuristics* from 1990 onward [Laporte et al., 2000].

Classical heuristics can in turn be classified into three groups: construction methods, two-phase methods, and improvement methods [Laporte and Semet, 1999]. Construction methods gradually build a feasible solution by selecting arcs, based on minimizing cost. Two-phase methods divide the problem into two stages: the first stage involves clustering customers into feasible routes while disregarding their order and the second stage constructs routes for each cluster. One of these two-phase methods is the sweep algorithm described by Laporte et al. [2000]. Improvement methods start with a feasible solution and try to improve it by exchanging arcs or nodes within or between the routes. The local search algorithm developed by Savelsbergh [1985] and Aarts and Lenstra [1996] belongs to the category of improvement heuristics. The advantage of classical heuristics is that they have a polynomial computation time [Laporte et al., 2000; Cordeau et al., 2002]. On the other hand, they perform only a limited search in the solution space.

During the past few decades, various meta-heuristics, such as tabu search (TS), simulated annealing (SA), and genetic algorithm (GA), have been applied to solve VRPs quickly and effectively [Laporte et al., 2000]. TS and SA move from one solution to another in the neighborhood until termination criteria are satisfied. Many different TS heuristics have been proposed with varying success. Rochat and Taillard [1995] used a TS heuristic to solve some benchmark VRPs. Osman [1993] obtained similar results using a SA. GA maintains a population of good solutions that are recombined to produce new solutions. Berger and Barkaoui [2003], Jeon et al. [2007], Yu et al. [2011] presented a hybrid genetic algorithm (HGA) to solve the VRP. Renaud et al. [1996] reported that such heuristics require substantial computing times and several parameter settings. The detail of GA, which is used in both the proposed hierarchical approach and endosymbiotic evolutionary algorithm, is described in the following sections.

## 2.2 The genetic algorithm (GA)

### 2.2.1 Background of the genetic algorithm

The theory of natural selection, proposed by Charles Darwin in 1859, states that individuals with certain favorable characteristics are more likely to survive in nature and consequently pass their characteristics on to their offsprings. Individuals with less favorable characteristics will gradually disappear from the population. In nature, the genetic inheritance is stored in individual chromosomes made of genes. The characteristics of every organism are controlled by the genes, which are passed on to the offspring when the organisms reproduce. Occasionally a mutation causes changes in the chromosomes. Due to natural selection, the

population will gradually converge toward improvement of the species, as the number of individuals having the favorable characteristics increases.

The GA is a randomized global search algorithm that solves intractable problems by imitating genetic processes observed during natural evolution. The "survival of the fittest" nature of this algorithm lends itself favorably to being extremely robust in its search for optimality [Gen and Cheng, 2000]. Fundamentally, the GA evolves a population of bit strings, chromosomes, or individuals, where each individual encodes a solution to a particular problem. This evolution takes place through the application of genetic operators, which mimic the phenomena such as reproduction and mutation observed in nature. The characteristics of the GA differ from those of other heuristics and can be described as follows [Rawlins, 1991; Gen and Cheng, 2000]:

- The GA works with coding of the solutions instead of the solutions themselves. Therefore, a well-designed coding or an efficient representation of the solutions is required.
- The GA searches for good solutions from a group of solutions. This is different from other meta-heuristics like the simulated annealing (SA) and the tabu search (TS), which start with a single solution and move to another solution by some transitions. Therefore, the GA performs a multi-directional search in the solution space, reducing the probability of finishing at a local optimum.
- The GA requires only the objective function value that measures the fitness of individuals, while many other algorithms require continuity or differentiability. Many real-life examples contain discontinuous search space.
- The GA is nondeterministic; i.e., it is stochastic in natural decisions, making the GA more robust.
- The GA is a heuristic because it does not know when it has found an optimal solution.

### 2.2.2 Procedure of the genetic algorithm

The procedure of the traditional GA is described as follows. In the first step, the GA starts from a randomly generated initial population, which is a set of solutions. Davis [1987] suggested that, for research purposes, much can be learned by initializing a population randomly. Moving from a randomly created population to a well-adapted population is a good test of the algorithm. Through this step, important features of the final solution will have been produced by the search and recombination mechanism of the algorithm, rather than the initialization process. In order to generate and search for an optimal solution, a function that evaluates the survivability of each solution in the population is required in the initialization process. This function is called the *fitness function*, and it evaluates each solution in accordance with its fitness value. The fitness function is the most critical part of the algorithm, as it is the one that decides how much time the algorithm takes to find the optimal solution.

The second step, a reproductive process, allows parent solutions to be randomly selected from the population. Typically, a lower selection pressure is indicated at the start of a search in favor of wide exploration of the search space, while a higher selection pressure is recommended at the end to narrow the search space [Gen and Cheng, 2000]. Offspring solutions are made by the reproductive processes using a crossover operator. The offspring solutions that are produced inherit some of the characteristics from each parent. Then a random mutation is applied to the offsprings with a certain probability. Gen and Cheng [2000] proved that the mutation operator can sometimes play a more crucial role than crossover. Therefore, the crossover and mutation operators need to be well designed in accordance with the problem at hand.

Finally, generation update takes place in the third step. The evaluation of the solutions can be related to the objective function value. In the VRPs, the total traveling distance and the level of violation of any constraint can be considered in the fitness function. Analogous to biological processes, offsprings with relatively good fitness levels are more likely to survive and reproduce, with the expectation that fitness levels throughout the population will improve as they evolve. More details can be found in Reeves [1993].

### 2.2.3 Encoding method for a solution

The preliminary component involves choosing the right coding schema for the representation of solutions to the problem. Diverse encoding methods have been suggested for different problems to provide efficient implementation of GAs. Depending on the symbols used for the bits of the individual, the encoding methods can be classified into:

- **Binary encoding** uses binary numbers (0 or 1) as the symbols for a bit in an individual. This is the most common encoding method because it is easy to create and manipulate. A wide range of problems can use binary encoding, one-point crossover, and mutation without modification [Davis, 1987]. For efficiency, however the other coding methods introduced below are more favorable in the real world.

- **Real number encoding** uses real numbers for a bit in an individual. This encoding is appropriate for function optimization problems. It has been widely confirmed that real number encoding performs better than binary encoding on optimization problems, as Eshelman and Schaffer [1993], Michalewicz [1996], and Walters and Smith [1995] reported.

- **Integer or literal permutation encoding** is useful for combinatorial optimization problems. Since the essence of combinatorial optimization problems is the search for a best permutation or combination of items subject to constraints, literal permutation encoding can be used for this type of problem. For more complex real-world problems, an appropriate data structure encoding is suggested for the bits of an individual in order to capture the nature of the problem [Gen and Cheng, 2000].

- **Data structure encoding:** According to the computer data structure, encoding methods can be classified into two types: one-dimensional encoding and multidimensional encoding. In most practices, one-dimensional encoding has been widely used, but some complex problems require multidimensional encoding. Cohoon and Paris [1986] used two-dimensional encoding for circuit placement problems. Anderson et al. [1991] used a two-dimensional grid type of encoding. Lim [2007] used two-dimensional encoding for vehicle routing problems with heterogeneous vehicles from multiple depots, allowing multiple visits.

### 2.2.4 Selection

The selection directs the genetic search toward promising regions in the solution space. Population diversity and selective pressure are the two most important factors in the genetic search [Michalewicz, 1996]. An increase in selective pressure decreases the population diversity, and vice versa; the two factors have a strong inverse relationship. Therefore, it is important to maintain the balance when determining a selection method for the GA. Four commonly used selection methods are as follows:

- **Roulette wheel selection:** In roulette wheel selection, the probability of being chosen is the individual's fitness divided by the sum of fitness of the whole population. Each individual is assigned a slice of a circular roulette wheel, the size of the slice being proportional to the individual's fitness. The wheel is spun N times, where N is the number of individuals in the population. On each spin, the individual under the wheel's marker is selected to be in the pool of parents for the crossover.

- **Tournament selection:** This selection method randomly chooses a set of individuals and picks out the best individual for reproduction among chosen individuals. The number of individuals in a competition is called the "tournament size." A common tournament size is two, and this is called a binary tournament. A random number r is then generated between 0 and 1. If $r < k$, where k is a parameter between 0 and 1, then the fitter of the two individuals is selected to be a parent. Otherwise, the less fit individual is selected. The two individuals are then returned to the original population for the next round of selection.

- **Elitism:** This is an addition to other selection methods that forces the GA to retain a number of good individuals in each generation. Without elitism, good individuals may be lost if they are not selected to reproduce or if they are destroyed by crossover or mutation.

- **Scaling:** The scaling method has been proposed to prevent premature convergence to local optima. The scaling method maps raw fitness values of all individuals in a population to the scaled fitness values, which are positive real values. The selection process may be performed based on the scaled fitness values. Many scaling methods have been proposed in the literature on GAs. Scaling parameters are known to be problem-dependent [Gen and Cheng, 2000]. One of the commonly used scaling methods in GAs is linear scaling, which adjusts the fitness values of all individuals such that the best individual gets a fixed

number of expected offsprings, thus preventing it from reproducing too many times [Gen and Cheng, 2000].

### 2.2.5 Crossover

An important genetic operator is the crossover, which simulates a reproduction by parents. It works on a certain number (occasionally, a pair) of solutions and recombines them in a certain way, generating one or more offspring. The offsprings share some of the characteristics from the parents through the crossover. In that way, the good characteristics of the current generation are passed on to following generations.

Many different crossover operators have been introduced in the literature. The functionality of the crossover depends on the encoding method, and the performance depends on how well it is adjusted to the problem. Commonly used crossover methods for VRPs are as follows [Gen and Cheng, 2000]:

- **Point crossover:** Among point crossovers, one-cut-point crossover is the simplest method. It selects one cut-point randomly in an individual, as shown in Figure 2.3. The selected point is indicated by an arrow. P1 represents the first parent and P2 represents the second parent. The one-cut-point crossover takes the pre-cut section from P1 as a proto-child and fills up the offspring by taking in order each legitimate gene from P2 to generate an offspring, as shown in Figure 2.3.

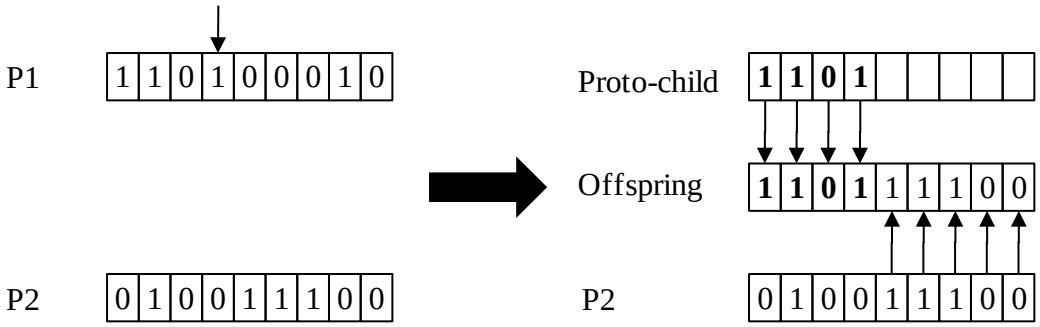



Figure 2.3 The one-cut-point crossover.

Two-cut-point and multi-cut-point crossovers are more advanced methods than one-cut-point crossover. Two-cut-point crossover is illustrated in Figure 2.4, where two points are randomly selected at P1 and the genes between two selected points are passed on to the offspring. Then, it takes each legitimate gene in the order shown in P2.

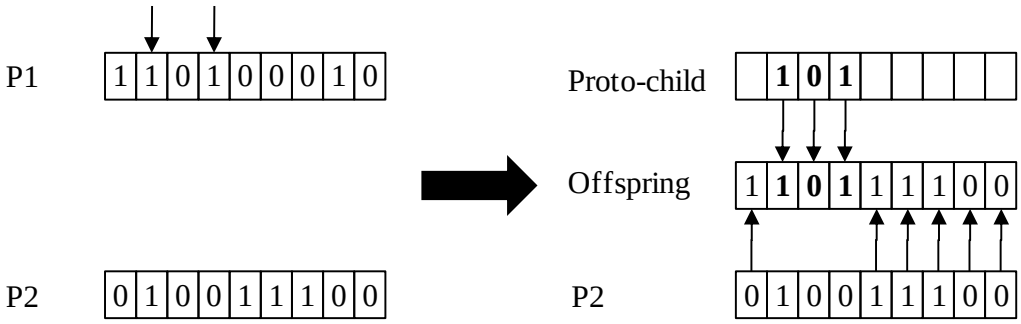



Figure 2.4 The two-cut-point crossover.

Multi-cut-point crossover is more complicated than two-cut-point crossover. The number of the cut-points is randomly chosen, and then the cut-points are selected according to the chosen number. An example of multi-cut-point crossover is illustrated in Figure 2.4. Four points are selected for the cut sections from P1. Then, each legitimate gene is taken in the order shown in P2.

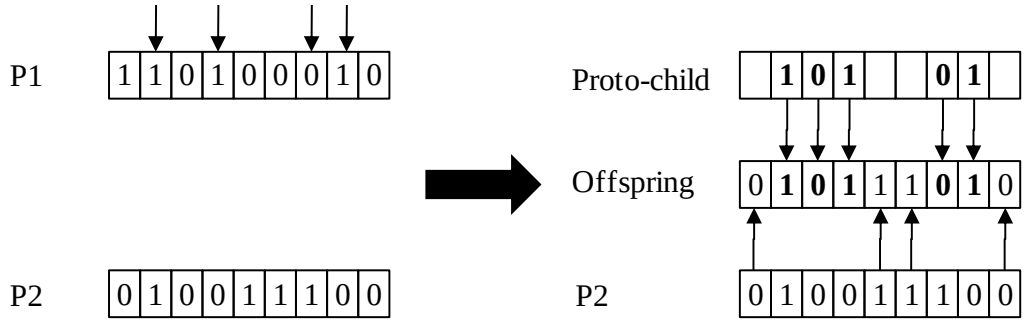



Figure 2.5 The multi-cut-point crossover.

- **Partial-mapped crossover (PMX):** PMX is a variant of two-cut-point crossover for binary string representation, and can be used for integer or literal permutation encoding. The PMX uses a special repair procedure to resolve the illegitimacy. An illustration of PMX is shown in Figure 2.6. Two positions along the genes from both P1 and P2 are randomly selected. The sub-genes defined by the two positions are called the mapping sections. The mapping section in P1 is copied to the proto-child at the same positions and others are copied in order from P2. From the mapping section of P1 and P2, the mapping relationship is determined. The genes that are not in the mapping section of P1 are changed according to the mapping relationships in the proto-child. Genes without the mapping relationship are simply copied in the proto-child.

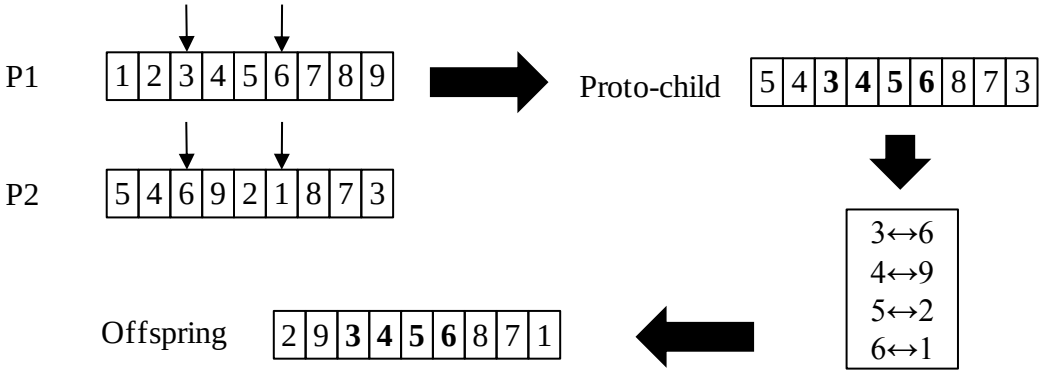



Figure 2.6 The partial-mapped crossover (PMX).

- **Order crossover (OX)** can be viewed as a variant of the PMX with a different repair procedure. An illustration of OX is shown in Figure 2.7. Sub-genes are taken from P1 by randomly choosing two points. The sub-genes are copied into the corresponding position of each gene in a proto-child. The corresponding genes in P2 are deleted and the remaining genes in P2 are placed into the proto-child from left to right in the same order as in P2.

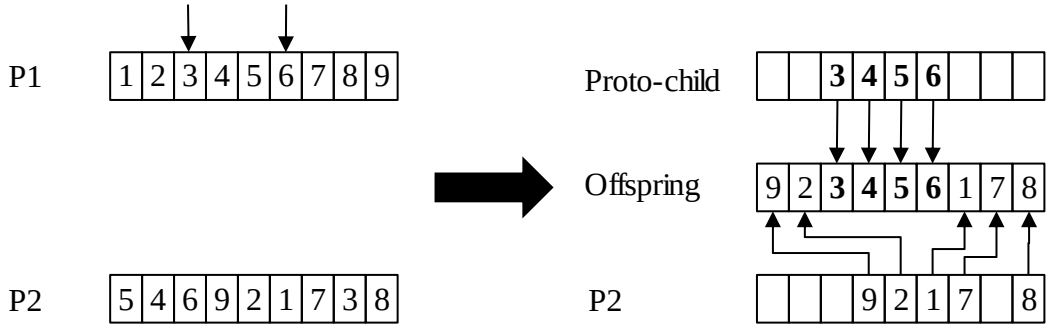



Figure 2.7 The order crossover (OX).

- **Uniform crossover** is accomplished by selecting two-parent solutions and randomly taking each gene from one parent to form the corresponding position of the child. Uniform crossover is illustrated in Figure 2.8. Each gene for the offspring is randomly selected from either parent and then copied to the offspring. This process is repeated until all genes of the offspring fill up completely.

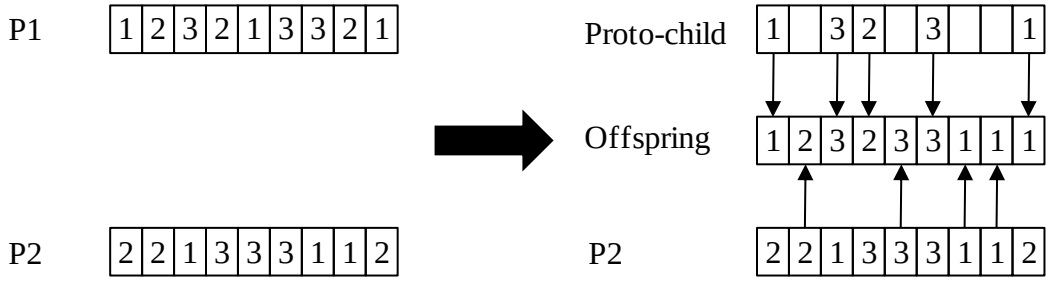



Figure 2.8 The uniform crossover.

- **Position-based crossover** is a variant of uniform crossover for permutation encoding together with a repair procedure, which can also be viewed as a variant of the OX where the genes are copied inconsecutively. A position-based crossover is illustrated in Figure 2.9. A set of genes in P1 is selected and copied into the corresponding positions of the proto-child. The corresponding genes in P2 are deleted. The remaining genes in P2 are placed into the proto-child from left to right in the same order as in P2.

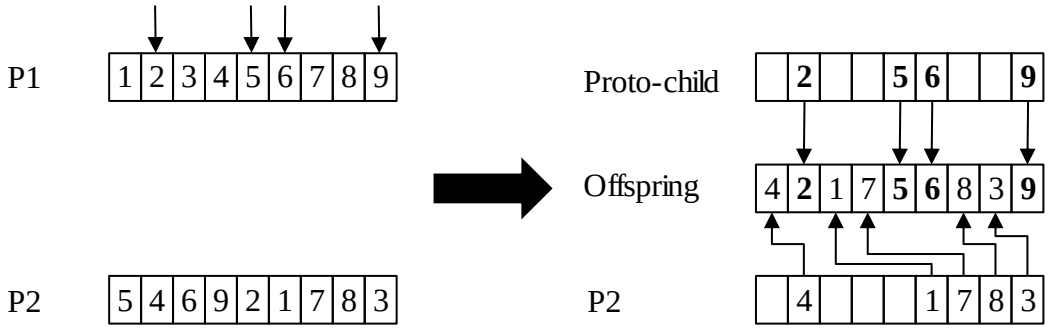



Figure 2.9 The position-based crossover.

- **Order-based crossover** is a slight variant of position-based crossover in which the order of genes at the selected position of one parent is imposed on the corresponding genes of the other parent, as shown in Figure 2.10.

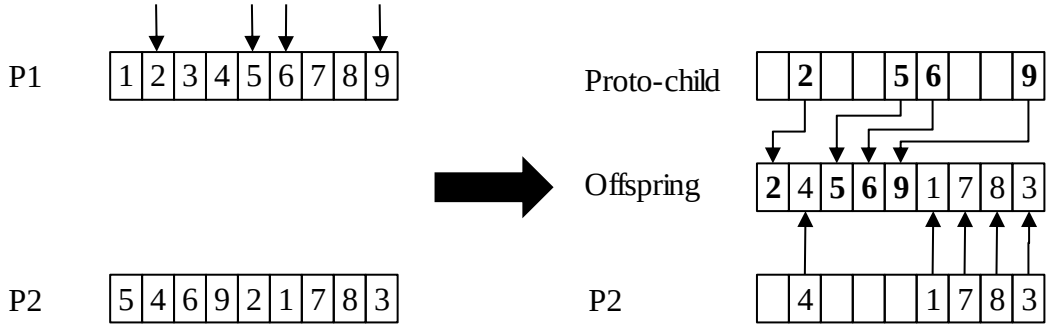



Figure 2.10 The order-based crossover.

- **Cycle crossover (CX):** Like the position-based crossover, CX takes some genes from one parent and selects the remaining genes from the other parent. The difference is that the genes from the first parent are not selected randomly, and only those genes that create a cycle according to the corresponding positions between parents must be selected. CX is illustrated in Figure 2.12. The cycle defined by the corresponding positions of genes between parents is created. The genes in the cycle to offspring with the corresponding positions of P2 are copied into the proto-child. The genes in P2 that are already in the cycle are deleted. The offspring is created by copying the remaining genes in P2.

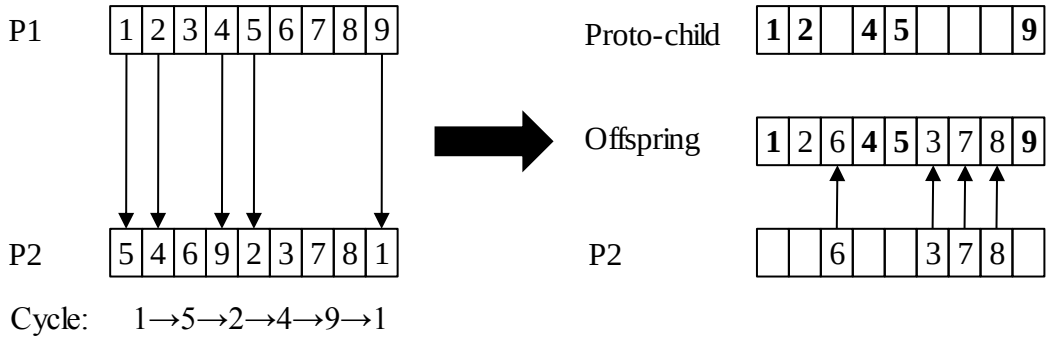



Figure 2.11 The cycle crossover.

### 2.2.6 Mutation

To explore different solutions beyond the neighborhood and avoid local optima, a mutation procedure needs to be implemented. In the GA, mutation plays an important role by either replacing the individuals lost from the population during the selection process, so that they can be tried in a new context, or providing the individuals that were not present in the initial population. Commonly used mutation methods are explained as follows [Gen and Cheng, 2000]:

- **Inversion mutation:** The inversion mutation selects two positions within an individual at random and then inverts the sub-genes between these two positions, as illustrated in Figure 2.12.


Figure 2.12 The inversion mutation.

- **Insertion mutation:** The insertion mutation randomly selects a gene and inserts it in a random position, as illustrated in Figure 2.13.


Figure 2.13 The insertion mutation.

- **Reciprocal exchange mutation:** The reciprocal exchange mutation selects two positions at random and then swaps the genes on these positions, as illustrated in Figure 2.14.

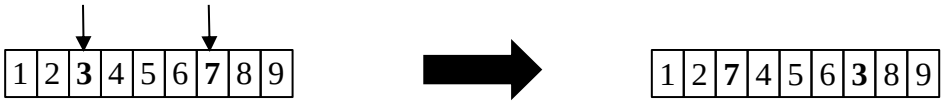



Figure 2.14 The reciprocal exchange mutation.

- **Point mutation:** The point mutation selects a position at random and changes the gene in the position to a certain gene, as illustrated in Figure 2.15.

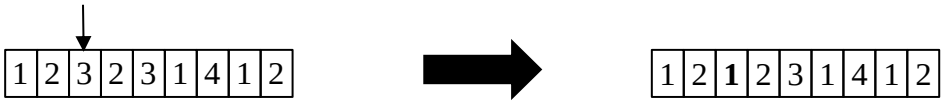



Figure 2.15 The point mutation.

### 2.2.7 Termination condition

The GA repeats selecting parents, performing the crossovers, and executing the mutations until termination criteria are met. The most frequently used stopping criterion is a maximum number of generations [Gen and Chang, 2000]. Another notable termination strategy is the population convergence criterion. The GA forces much of the entire population to converge to a single solution. When the sum of the deviations among individuals becomes smaller than a specified threshold, the algorithm is terminated. The algorithm can also be terminated due to a lack of improvement in the best solution over a predetermined number of generations. For each criterion, a threshold needs to be carefully selected. Several strategies can be used in conjunction with each other.

## 2.3 The Taguchi method

To fine-tune the performance of algorithms or processes, many parameters must be set carefully. The technique of investigating all possible combinations in experimental conditions involving multiple factors is known as *Design of Experiment*. The method of experimental design constitutes the preset values of parameters to obtain the optimized output as it allows the designer to determine the significant parameters over the others. The Taguchi method was introduced to search effectively for the optimal parameters.

The Taguchi method for parameter designs is an important tool in the category known as robust design. Robust design is an engineering methodology for optimizing the product and process conditions that are minimally sensitive to the causes of variations, and that produce

high-quality products with low development and manufacturing costs. The orthogonal array and the signal-to-noise ratio (SNR) are two major tools used in the Taguchi method. Additional details can be found in Taguchi et al. [2000] and Wu [2000].

The Taguchi method uses matrices called orthogonal arrays to determine which combinations of factor levels to use for each experimental run. An orthogonal array is a fractional factorial matrix, which assures a balanced comparison of levels of any factor. It is a matrix of numbers arranged in rows and columns, where each row represents the level of the factors in each run and each column represents a specific factor that can be changed from each run. The symbol for three-level orthogonal arrays is $L_n(3^k)$, where $n$ is the number of experimental runs, 3 is the number of levels for each factor, and $k$ is the number of factors. The letter $L$ comes from Latin, since the orthogonal arrays were associated with Latin square designs from the outset.

The SNR is the ratio of the signal over the noise, which measures the strength of signal with the existence of noises. The higher SNR means that the process or design is more robust. There are several SNRs available depending on the type of characteristics or outputs: nominal-is-best, smaller-the-better or larger-the-better. Further details can be found in Taguchi et al. [2000] and Wu [2000]. Taking the case of the smaller-the-better characteristic, suppose that we have a set of experiment runs $x_1, x_2, \ldots, x_n$. Since the value of the SNR is large for favorable situations, the following formulation for the smaller-the-better characteristic is used:

$$\mathrm{SNR} = -10 \log_{10}\left(\frac{1}{n}\sum_{i=1}^{n} x_i^2\right).$$

Since the objective of VRPs is to minimize the total traveled distance, the smaller-the-better is an appropriate measure in this thesis. The proposed GA with different process parameters shows different performances.

## 2.4 The endosymbiotic evolutionary algorithm (EEA)

A concept of endosymbiotic hypothesis to explain biological theories was proposed by Margulis [1981], and it is widely accepted in the area of biology. This concept is used to design an endosymbiotic evolutionary algorithm (EEA), which is one of the symbiotic evolutionary algorithms rooted at the biological hypothesis used to explain that the mitochondria and chloroplasts are the result of years of collaborative evolutions. The concept is initiated by the endocytosis of bacteria and blue-green algae, which, instead of becoming digested, become symbiotic. It hypothesizes that prokaryotes enter into and become parasitic on eukaryotes. The prokaryotes live together in symbiosis and evolve into a eukaryote.

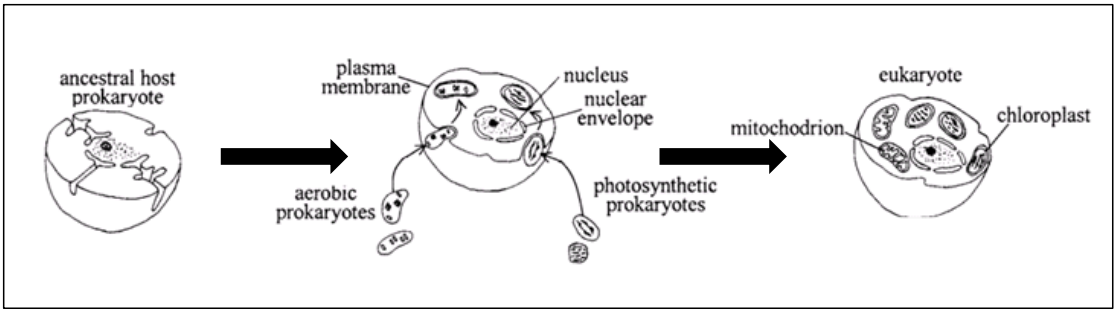



Figure 2.16 The endosymbiotic evolution.

Figure 2.16, adapted from Campbell et al. [1996], illustrates the concept of endosymbiotic evolution. Larger species engulfed smaller ones that continued to live and functioned in the

larger host cell. Both are called symbionts, or prokaryotes. They then evolved with this mutual assistance into a better form of life, which is called an *endosymbiont* or *eukaryote*. EEA is a probabilistic meta-heuristic algorithm that mimics this evolution.

The EEA is one of symbiotic evolutionary algorithms (SEAs). SEAs have been studied since the 1990s and are known as effective tools to solve complex problems in dynamic situations where multiple subproblems are interwoven. The original problem is split into subproblems, and each subproblem has a population consisting of a set of elements. An element in the population for a subproblem can be a part of a complete solution to the original problem. Several variants of SEAs have been proposed by Potter [1997], Moriarty and Miikkulainen [1997], and Kim et al. [2003]. It was reported that SEAs perform better than hierarchical approaches using the genetic algorithm. However, most existing SEAs tend to allow an individual entity to evolve independently without considering symbiotic partners, which are, as partial solutions to other subproblems in the original problem, evaluated together in the fitness function for complete solutions. If two or more cooperative, interacting species evolve only in an independent fashion, the deviation of symbiotic partners during the evolution increases, hindering consistent searches toward good solutions and resulting in a slow convergence.

In order to overcome this drawback, the endosymbiotic evolutionary algorithm was proposed [Kim et al., 2001, 2006]. In EEA, not only does an individual entity evolve independently, but symbiotic partners of the entity also are allowed to evolve together. That is, by allowing some species to adapt for a certain period without changing their symbiotic partners, the search for the solution is performed more efficiently and effectively. In EEA, unlike other SEAs, if a species mates with symbiotic partners that highly appreciate the partnership, it evolves with them into a better form of life.

# 3. MIXED-INTEGER NONLINEAR PROGRAMMING MODEL

In this chapter, a mathematical model of the VRP for the delivery and installation is presented. The objective of this model is to find the optimal routing and scheduling solution considering not only the shortest traveling time of all vehicles but also the smallest cost of vehicles in operation. The solution must be able to provide routes of delivery and installation vehicles. The problem under consideration is formulated as a mixed-integer nonlinear programming (MINP) model, as explained in the following sections.

## 3.1 Model assumptions

The VRP under consideration can be represented as a network, where nodes are customers or a single depot and the links are roads linking any pair of nodes. In a network, customers are indexed from 1 to N while the index 0 denotes a single depot. Customer *i* has a known demand $m_i$ ($i = 1, \dots, N$) and its location is known. There is a set of customers, A, requiring both delivery and installation ($|A| \leq N$). There are two types of vehicles, one for delivery and one for installation, respectively. The complete list of assumptions used in this thesis is as follows:

- Each vehicle starts from and returns to the depot.
- The demand of each customer is known.
- The demand of each customer must be satisfied by a single vehicle.
- The demand of a customer is less than the capacity of a delivery vehicle.
- All delivery vehicles have a homogeneous capacity.
- All installation vehicles have no delivery capacity.

- The same amount of installation time is required for every installation customer.
- The locations of all customers and the depot are known.
- The traveling time between any pair of locations is known.
- The traveling time matrix is symmetric. That is, the traveling time from location i to j is equal to the traveling time from location j to i.

## 3.2 Notations

The parameters and decision variables used in this thesis are as follows:

***Parameters:***

| | |
|---|---|
| N | Number of customers |
| K | Number of delivery vehicles |
| S | Number of installation vehicles |
| A | Set of customers requiring both delivery and installation |
| $T_{ij}$ | Traveling time between location i and location j |
| FCO | Fixed cost per vehicle |
| $D_i$ | Demand of the customer at location i |
| CAP | Capacity of delivery vehicles |
| $R_i$ | Time to complete the installation at location i, $i \in A$ |
| OPT | Available operation time per shift for vehicles |
| SVL | Maximum allowable time between delivery and installation (i.e., service level) |

***Decision Variables:***

$x_{ijp}$ $\begin{cases} 1, & \text{if the delivery vehicle k travels from location i to location j} \\ 0, & \text{otherwise} \end{cases}$

$y_{ijq}$ $\begin{cases} 1, & \text{if the installation vehicle s travels from location i to location j} \\ 0, & \text{otherwise} \end{cases}$

$e_i$ Arrival time of the delivery vehicle at location i, $i \in A$

$f_i$ Arrival time of the installation vehicle at location i, $i \in A$

$w_i$ Waiting time of the installation vehicle at location i, $i \in A$

$u_{ip}$ Subtour prevention variables for $x_{ijp}$

$v_{iq}$ Subtour prevention variables for $y_{ijq}$

## 3.3 Mathematical model

The mathematical model of the VRP under consideration for delivery and installation vehicles is given as follows:

$$\text{Minimize } Z = \sum_{p=1}^{K}\sum_{i=0}^{N}\sum_{j=0}^{N} T_{ij}x_{ijp} + \sum_{q=1}^{S}\sum_{i=0}^{N}\sum_{j=0}^{N} T_{ij}y_{ijq} + \sum_{i=0}^{N} w_i + FCO\left(\sum_{k=1}^{K}\sum_{j=1}^{N} x_{0jk} + \sum_{s=1}^{S}\sum_{j=1}^{N} y_{0js}\right)$$

Subject to

$$\sum_{p=1}^{K}\sum_{j=1}^{N} x_{ijp} \le K \qquad \text{for } i = 0 \qquad (1)$$

$$\sum_{j=1}^{N} x_{ijp} \le 1 \qquad \text{for } i = 0, \forall p \qquad (2)$$

$$\sum_{j=1}^{N} x_{ijp} - \sum_{j=1}^{N} x_{jip} = 0 \qquad \text{for } i = 0, \forall p \qquad (3)$$

$$\sum_{p=1}^{K}\sum_{j=0}^{N} x_{ijp} = 1 \qquad \text{for } i = 1 \ldots N \qquad (4)$$

$$\sum_{p=1}^{K}\sum_{i=0}^{N} x_{ijp} = 1 \qquad \text{for } j = 1 \ldots N \qquad (5)$$

$$\sum_{j=0}^{N} x_{ijp} - \sum_{j=0}^{N} x_{jip} = 0 \qquad \text{for } i = 1 \ldots N, \ \forall p \qquad (6)$$

$$\sum_{i=1}^{N} D_i \left( \sum_{j=0}^{N} x_{ijp} \right) \le CAP \qquad \text{for } \forall p \qquad (7)$$

$$\sum_{i=0}^{N}\sum_{j=0}^{N} T_{ij}x_{ijp} \le OPT \qquad \text{for } \forall p \qquad (8)$$

$$u_{ip} - u_{jp} + (N+1)x_{ijp} \le N \qquad \text{for } i \ne 0, \ j \ne 0, i \ne j, \forall p \qquad (9)$$

$$\sum_{i=1}^{N} x_{iip} = 0 \qquad \text{for } \forall i, \forall p \qquad (10)$$

$$x_{ijp} = \{0,1\} \quad \text{for } \forall i, \forall j, \forall p \tag{11}$$

$$\sum_{q=1}^{S}\sum_{j=1}^{N} y_{ijq} \le S \quad \text{for } i=0,\ j \in A \tag{12}$$

$$\sum_{j=1}^{N} y_{ijq} \le 1 \quad \text{for } i = 0, \forall q \tag{13}$$

$$\sum_{j=1}^{N} y_{ijq} - \sum_{j=1}^{N} y_{jiq} = 0 \quad \text{for } i = 0, \forall q \tag{14}$$

$$\sum_{q=1}^{S}\sum_{j=0}^{N} y_{ijq} = 1 \quad \text{for } i \in A \tag{15}$$

$$\sum_{q=1}^{S}\sum_{i=0}^{N} y_{ijq} = 1 \quad \text{for } j \in A \tag{16}$$

$$\sum_{j=0}^{N} y_{ijq} - \sum_{j=0}^{N} y_{jiq} = 0 \quad \text{for } i \in A, \ \forall q \tag{17}$$

$$v_{iq} - v_{jq} + (N+1)y_{ijq} \le N \quad \text{for } i \ne 0,\ j \ne 0,\ i \ne j, \forall q \tag{18}$$

$$\sum_{i=1}^{N} y_{iiq} = 0 \quad \text{for } i \in A, \ \forall q \tag{19}$$

$$y_{ijq} = \{0,1\} \quad \text{for } \forall i, \forall j, \forall s \tag{20}$$

$$e_0 = f_0 = w_0 = r_0 = 0 \tag{21}$$

$$f_i - e_i \le SVL \quad \text{for } i \in A \tag{22}$$

$$w_i \ge 0, \ w_i \ge e_i - f_i \quad \text{for } i \in A \tag{23}$$

$$\sum_{p=1}^{K}\sum_{i=0}^{N} x_{ijp}\left(e_i + T_{ij}\right) - e_j = 0 \quad \text{for } \forall j \tag{24}$$

$$\sum_{q=1}^{S}\sum_{i=0}^{N} y_{ijq}\left(f_i + w_i + R_i + T_{ij}\right) - f_j = 0 \quad \text{for } j \in A \tag{25}$$

$$\sum_{i=0}^{N}\sum_{j=0}^{N} y_{ijq}\left(T_{ij} + w_i + R_i\right) = OPT \quad \text{for } \forall q \tag{26}$$

The mathematical model for the VRP under consideration is formulated as a mixed-integer nonlinear programming (MINP) model. The objective function of the given MINP consists of three parts. The first part of the objective function is the sum of the shortest traveling times of the vehicles, which is the major cost in the problem. The second part of the objective function is the sum of the waiting time of installation vehicles occurring due to the synchronization of two different types of vehicles. The last part is the fixed cost of vehicles in operation. For the last part, the fixed cost per vehicle (FCO) can be considered as a certain penalty. The traveling distance or the transportation cost can also be used to optimize the model for the different purposes.

The constraints can be classified into three different sets. The first set of constraints concerns the VRP for delivery vehicles (constraints (1) through (11)); the second set concerns the VRP for installation vehicles (constraints (12) through (20)); and the third set concerns the synchronization for both types of vehicles (constraints (21) through (26)). Constraints (1) and (12) constrain the numbers of delivery and installation vehicles, respectively, by limiting the number of vehicles that can depart from the depot. Constraints (2), (3), (13) and (14) ensure that all vehicles must return to the depot. Constraints (4) through (6) require that each customer can accept only one visit by a delivery vehicle. Constraints (15) through (17) require that each customer who needs the installation service can accept a visit by only one installation vehicle. Constraint (7) assures that the sum of demand of all customers on a vehicle route cannot exceed the loading capacity of a delivery vehicle. Constraints (8) and (26) ensure that the duration of each vehicle's shift cannot be longer than an available operation time per shift for delivery and installation vehicles, respectively. Constraints (9) and (18) eliminate the possible subtours. Constraints (11) and (20) define the binary integer decision variables which represent the travels of corresponding vehicles between locations. Constraints

(21) through (25) guarantee the quality of service by defining the service level for customers requiring both delivery and installation.

In order to ensure fulfillment of the planned service level, which is a unique characteristic of the VRP in this thesis, the waiting time of the installation vehicles for the customers requiring both delivery and installation has been calculated as well. If an installation vehicle arrives at a customer's location earlier than a delivery vehicle, it needs to wait for the delivery vehicle to arrive before installation can start. In addition, an installation vehicle will not necessarily leave immediately after the installation at one location; it may stay longer to avoid waiting at the next location or so that its driver can make other arrangements. Hence, the waiting time of an installation vehicle at customer location i ($w_i$) is defined as the amount of time spent by the installation vehicle before or after the installation at location i. This thesis also evaluated another mathematical programming model by replacing constraint (23) with a different constraint, $\max\{0, e_i - f_i\} - w_i = 0$, in order to remove the waiting time after the installation at a location. Since the third set of constraints includes nonlinear ones, the mathematical programming model for the VRP under consideration is more complicated than other traditional VRPs.

## 3.4 Complexity of the problem

In the computational complexity theory, a traditional VRP is defined as one of NP-hard problems. No polynomial time algorithm is known for any NP-hard problem. The computational time of a NP-hard problem increases exponentially as the size of the problem grows. The traveling salesman problem (TSP), a well-known NP-hard problem [Garey and

Johnson, 1979], is a special case of VRPs. In the traditional VRP, there are given numbers of customers and vehicles. Each customer has its location and demand, and each vehicle has the same loading capacity. If we consider the case of a VRP that has only one vehicle and zero demand for all customers, this restricted VRP is exactly the same as a simple TSP. In other words, such a VRP is a generalized case of the TSP. Therefore, the traditional VRP is NP-hard in a strong sense.

The MINP for the problem can be viewed as a combined model of two VRPs. One is a VRP for delivery vehicles, which shows characteristics of the CVRP, and the other is a VRP for installation vehicles, which shows characteristics of the VRPTW. The CVRP is the simplest of VRPs, and the VRPTW is an extension of the CVRP. Both of these are traditional types of VRPs that have been studied extensively and are defined as NP-hard problems. Additionally, the formulated mathematical model contains nonlinear constraints for the relation between two VRPs, thereby making the problem more complicated. The problem can be regarded as a combination of two NP-hard problems, and hence as NP-hard in a strong sense.

# 4. HIERARCHICAL APPROACH USING THE GENETIC ALGORITHM FOR SYNCHRONIZATION OF THE DELIVERY AND THE INSTALLATION

Since the VRP under consideration is one of NP-hard problems, it will be hard to use existing mathematical approaches to solve such a problem of large size within polynomial computation times. In order to effectively and efficiently find high-quality solutions in a reasonably small amount of time, a hierarchical approach using the genetic algorithm is proposed in this chapter.

## 4.1 Hierarchical approach to the vehicle routing problem

The problem under consideration is more complicated than other traditional VRPs because there are two different types of vehicles, delivery and installation vehicles, which must be synchronized to guarantee the quality of service. Therefore, it is necessary to develop a systematic approach not only to find routes and schedules for delivery and installation vehicles but also to synchronize both types of vehicles. A search of routes and schedules for delivery vehicles and a search of routes and schedules for installation vehicles can be defined as subproblems of the original problem. In order to obtain good solutions of the original problem, a hierarchical approach dealing with these two subproblems is developed in this thesis. The procedure of the proposed hierarchical approach is illustrated in Figure 4.1.

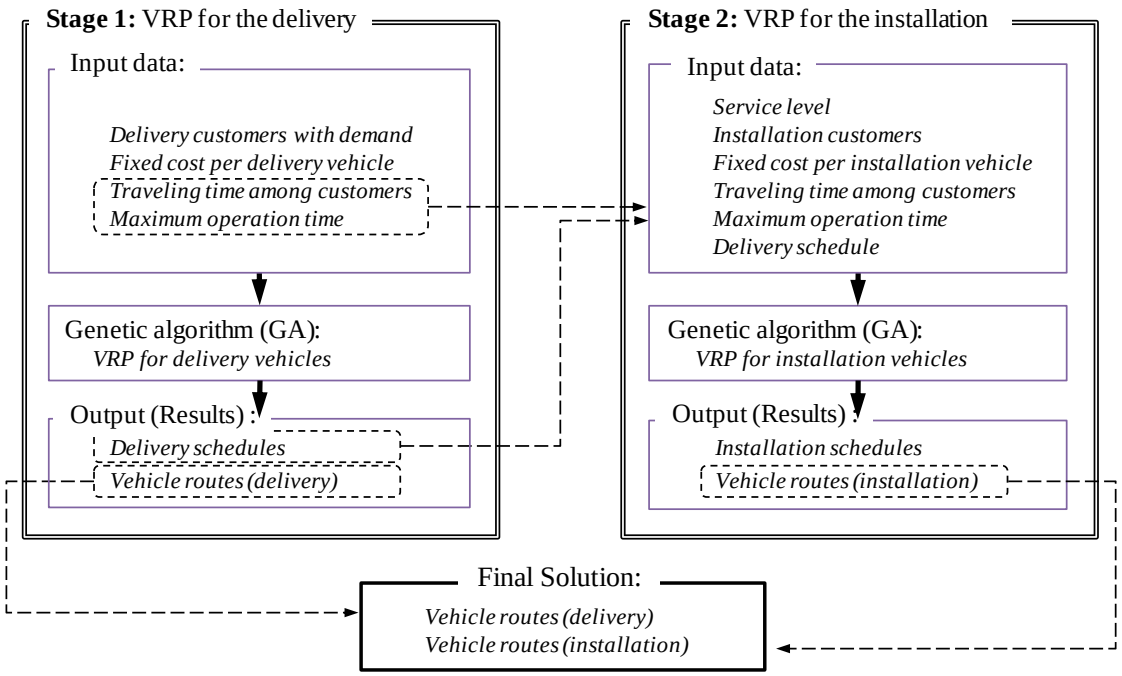



Figure 4.1 The procedure of the proposed hierarchical approach.

The proposed hierarchical approach divides the original problem into two stages, each of which contains a subproblem. The subproblem in Stage 1 is a VRP for delivery vehicles, and the subproblem in Stage 2 is a VRP for installation vehicles. From the subproblem in Stage 1, a set of routes and schedules is generated. The generated set of routes and schedules is a partial solution of the original problem and is then used as a part of the input data for the subproblem in Stage 2. Based on the partial solution from the subproblem in Stage 1, a set of routes and schedules for installation vehicles is determined to solve the subproblem in Stage 2. The set of routes and schedules for installation vehicles is the other partial solution of the original problem. Therefore, the synchronization of the two types of vehicles is automatically completed while solving the subproblem in Stage 2. Finally, the combination of the two partial solutions constitutes the solution of the original problem.

The subproblem in Stage 1 has the characteristics of capacitated vehicle routing problems (CVRP), which were briefly described in Section 2.2.1. The subproblem assumes that all delivery vehicles have an identical loading capacity. They must return to the depot within a specific time, called the *maximum operation time*. The limited loading capacity and the maximum operation time need to be considered when customers are assigned to delivery vehicles. All customers must be visited only once by a single delivery vehicle. It is assumed in this thesis that unloading times for all customers are negligible, though this assumption can be modified without loss of generality. Fixed cost per delivery vehicle is considered in the proposed algorithm to minimize the number of delivery vehicles in operation, but this factor could be omitted. The algorithm for the subproblem in Stage 1 determines routes and schedules for delivery vehicles and their arrival times at each customer's location. The arrival times of delivery vehicles at the customers who require both delivery and installation are later fed to the subproblem in Stage 2 in order to be considered for synchronization of delivery and installation vehicles.

As mentioned previously, the subproblem in Stage 2 includes the characteristics of the VRP with time windows (VRPTW). Unlike the subproblem in Stage 1, the loading capacity of installation vehicles is not considered, since the installation requires only the service to be rendered, not the goods. Similar to the subproblem in Stage 1, all installation vehicles must return to the depot within a specified maximum operation time. Installation vehicles must visit all customers who require both delivery and installation. The customers must be visited only once, by a single installation vehicle, within a specified time after a delivery vehicle has arrived at that customer, so as to fulfill the service guarantee. This specified time is called the *service level*. Hence, each customer requiring both delivery and installation has a time window following delivery within which he or she expects the arrival of an installation vehicle. It is assumed that the time windows for all customers are identical, though this provision could be

easily relaxed without loss of generality. If an installation vehicle arrives earlier than a delivery vehicle at a customer location, the installation vehicle must wait there until it can start the installation service. The waiting times of installation vehicles or customers can be considered as a penalty. It is assumed that all installation service requires the same amount of installation time for each customer. Fixed cost per installation vehicle is considered in the proposed algorithm to minimize the number of installation vehicles used. The algorithm for the subproblem in Stage 2 determines routes and schedules for all installation vehicles.

Finally, to complete the solution of the original problem, a set of routes and schedules for all delivery and installation vehicles is decided through this hierarchical approach using the genetic algorithm. The fitness function in the GA for Stage 1 considers the traveling times and fixed costs of all delivery vehicles, while the one for Stage 2 considers the traveling times, waiting times, installation times, and fixed costs of all installation vehicles.

## 4.2 Procedure of the genetic algorithm for subproblems

The proposed hierarchical approach uses genetic algorithms to solve subproblems in Stages 1 and 2. As the characteristics of both subproblems are different, the GAs for the subproblems have some differences. However, both GAs for the subproblems follow the basic procedure illustrated in Figure 4.2. The GA consists of several processes, such as the process for the initialization, the fitness function, the selection, the crossover, the mutation, and the local search.

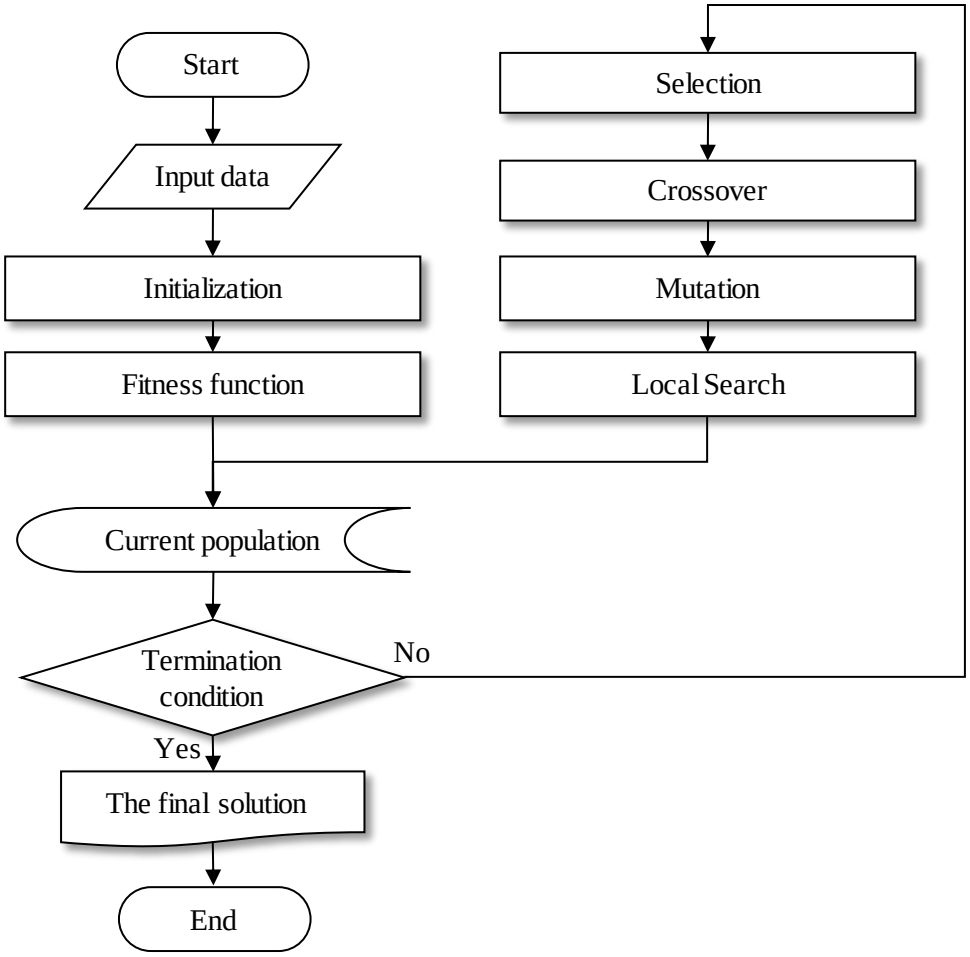



Figure 4.2 The procedure of the proposed genetic algorithm.

The GA creates a randomly generated initial population, which is a set of feasible solutions (or "individuals"). The randomly generated initial population evolves into a well-adapted population over generations. Through the adaptive search and recombination mechanisms, the GA obtains high-quality solutions at the end. In order to assess how good the solutions are, a function that effectively evaluates the survivability of individuals in the population is required. This function is called a *fitness function*. All solutions in the population are evaluated by the fitness function and are ranked according to their fitness values. If a solution has a high fitness value, it has more chance to survive in the next generation and to produce offsprings. Otherwise, it can be easily eliminated from the population.

The problem considered in this thesis consists of two subproblems. Each subproblem has a different fitness function. From the population of the current generation, offspring solutions for the next generation are produced by the crossover and mutation operations in the algorithm. The crossover operation allows parent solutions to be randomly selected from the population and to produce offspring solutions which inherit some of the characteristics from them. The mutation operation is applied to current solutions with a certain probability to generate mutant solutions. The purpose of the mutation operation is to allow the algorithm to avoid local optima and to try to search in other directions. Then, a local search procedure is applied to see the possibility of improving the best solution in the population. The population continuously evolves as these operations are repeated to create the next generation of possible solutions until the algorithm meets a certain termination condition.

In this thesis, the ranking replacement strategy is used to construct the population for the next generation in the proposed algorithm (Chu and Beasley, 1998). At the beginning of the evolution in each generation, the population at the end of the previous generation is duplicated to the population of the current generation. In the course of crossover procedures, newly generated offspring competes with all individuals in the current population. That is, let $\pi_{offspring}$ and $\pi_w$ be the fitness values of the new offspring and the worst individual in the current population, respectively. If $\pi_{offspring} > \pi_w$, the corresponding offspring will replace the worst individual in the current population.

## 4.3 Components of the proposed genetic algorithm

The GA consists of several components, including the population, the individual, the selection operation, the crossover operation, and the mutation operation. Using a different combination of components may have an impact on the GA's performance in terms of algorithm speed and solution quality. In the following subsections, the components of the proposed GA are described in detail.

### 4.3.1 Genetic representations

The subproblems in Stages 1 and 2 have different characteristics and constraints, which were already explained in Section 4.1. As a result there are different genetic representations for the two subproblems. Sections 4.3.1.1 and 4.3.1.2 describe their genetic representations, used in the proposed GAs for subproblems in Stages 1 and 2, respectively.

#### 4.3.1.1 Genetic representation of the VRP for delivery vehicles

The subproblem in Stage 1 can be defined as a CVRP for delivery vehicles. A one-dimensional array is used to represent an individual or a solution, as shown in Figure 4.3.

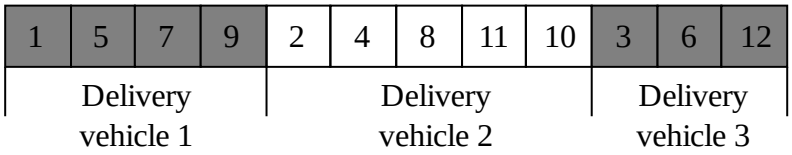



Figure 4.3 The genetic representation for the subproblem in Stage 1.

Numbers in the boxes indicate customers requiring the delivery, whether or not they want the installation as well. Since each customer is to be visited by a single delivery vehicle, a

customer must be shown only once in the array. The alternating shaded areas in Figure 4.3 represent different groups of customers served by different delivery vehicles. The decoding procedure, which is a part of the fitness function evaluation, for the subproblem in Stage 1 determines which delivery vehicle visits which customers, along with the visiting sequences for each delivery vehicle. A simple greedy method is used in the decoding procedure to assign customers to delivery vehicles. In order to identify a group of customers to be served by a delivery vehicle, the customers' demands, the loading capacity of the vehicle, and the maximum operation time are considered.

Figure 4.3 shows an example of a genetic representation and the decoding procedure for the subproblem in Stage 1. There are 12 customers requiring the delivery, and the customers are served by three delivery vehicles. Let the depot denote the location 0. The routes of three delivery vehicles are {0, 1, 5, 7, 9, 0}, {0, 2, 4, 8, 11, 10, 0}, and {0, 3, 6, 12, 0}. The arrival times of delivery vehicles at the customer locations are calculated from the routes, using the distances between the locations and the possible unloading times.

#### 4.3.1.2 Genetic representation of the VRP for installation vehicles

The subproblem in Stage 2 can be defined as a VRPTW for installation vehicles only. A one-dimensional array is used to represent an individual or a solution for the subproblem in Stage 2, as shown in Figure 4.4.

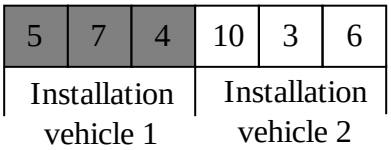



Figure 4.4 The genetic representation for the subproblem in Stage 2.

Numbers in the boxes indicate customers requiring both delivery and installation. Since the customers are to be visited by a single installation vehicle, numbers must appear only once in the array. The alternating shaded areas in Figure 4.4 represent different groups of customers served by different installation vehicles. The decoding procedure for the subproblem in Stage 2 determines which installation vehicle visits which customers, along with visiting sequences for each installation vehicle. As in the subproblem in Stage 1, a simple greedy method is used in the decoding procedure to assign customers to installation vehicles. However, unlike the subproblem in Stage1, in order to assign a group of customers to an installation vehicle, the time windows for customers requiring installation, as well as the maximum operation time, must be considered.

Figure 4.4 shows an example of the genetic representation and the decoding procedure for the subproblem in Stage 2. There are six customers requiring installation, and the customers are served by two installation vehicles. Let the depot denote the location 0. The routes of the two installation vehicles are {0, 5, 7, 4, 0} and {0, 10, 3, 6, 0}. The arrival times of installation vehicles at the customer locations are calculated from the distance between the locations, the installation times, and the waiting times.

### 4.3.2 Initialization, fitness function, and selection

A population in the GA consists of a set of individuals. The population must be initialized at the beginning. In order to generate the initial population, all individuals are randomly generated. Each individual in the population must be evaluated by its survivability in the problem through a fitness function. The evaluation value of the individual is called the *fitness*

*value*. The individual having a higher fitness value than others will have a greater chance to survive during the evolution in the GA.

In the subproblem in Stage 1, the sum of traveling times of delivery vehicles, the sum of unloading times at customers, and the sum of fixed costs of used delivery vehicles are considered to calculate the fitness value of an individual. Let $\tau_{a,t}$ be the sum of traveling times of delivery vehicles in individual *a* at generation *t*, $\Lambda$ be the sum of unloading times at customers in individual *a*, and $\delta_{a,t}$ be the sum of fixed costs of delivery vehicles used in individual *a* at generation *t*. According to the assumptions of the problem, the unloading time at any customer is identical and the number of customers is known. The sum of unloading times at customers ($\Lambda$) can be easily calculated and left as a constant, since all individuals would have the same value. Therefore, the sum of unloading times can be ignored in the evaluation of individuals. The fitness function of individual *a* at generation t for the subproblem in Stage 1 ($\pi_{a,t}$) is defined as follows:

$$\pi_{a,t} = \frac{1}{\tau_{a,t} + \delta_{a,t}(+\Lambda)}$$

In the subproblem in Stage 2, the sum of traveling times of installation vehicles, the sum of installation times at customers, the sum of waiting times of installation vehicles, and the sum of fixed costs of used installation vehicles are considered to calculate the fitness value of an individual. Let $\varphi_{b,t}$ be the sum of traveling times of installation vehicles in the individual *b* at generation *t*, I be the sum of installation times at customers in individual *b*, $\omega_{b,t}$ be the sum of waiting times of installation vehicles in the individual *b* at generation *t*, and $\gamma_{b,t}$ be the sum of fixed costs of used installation vehicles in individual *b* at generation *t*. The sum of installation times at customers (I) can be easily calculated and left as a constant, since all individuals have

the same value. Therefore, the installation time can be ignored in the evaluation of individuals. The fitness function of individual *b* for the subproblem in Stage 2 ($\rho_{b,t}$) is defined as follows:

$$\rho_{b,t} = \frac{1}{\varphi_{b,t} + \omega_{b,t} + \gamma_{b,t}(+\mathrm{I})}$$

After the evaluation of all individuals in the population has been completed, the ranking replacement strategy is applied to generate the new population. In GAs for both subproblems, higher fitness values are more desirable to generate high-quality solutions.

An appropriate selection method is also one of the important operations in the proposed GA to produce offsprings for the next generation. The selection method fundamentally gives a greater chance of being chosen for reproduction to individuals having higher fitness values. The roulette wheel selection method is used in the algorithm proposed in this thesis. The probability for an individual to be selected is calculated as the fitness value of the individual divided by the sum of the fitness values of all individuals in the population. The roulette wheel selection method is known as an acceptable selection method [Gen and Cheng, 2000].

### 4.3.3 Crossover

Crossover is one of the important reproduction procedures in GAs. A hybrid order crossover procedure has been implemented to efficiently and effectively reproduce new offsprings from two parents in the current population. The proposed crossover operator is a hybrid of the order crossover and the one-cut-point crossover, which are both described in Section 2.2.5,

respectively. A number of pairs of individuals equal to the size of the population are randomly selected by the selection method. The selected pairs may participate or not participate in the reproduction. For each pair, a random number between 0 and 1 is generated, and if the number is over a given probability, called the *crossover rate*, then the corresponding pair (P1 and P2) proceed to the crossover procedures, which are as follows:

Step 1: A vehicle is randomly chosen from one parent (P1).

Step 2: The corresponding genes, which are a series of location indices for the chosen vehicle, are copied into an offspring in the same order as they appear in P1, and the corresponding genes are deleted from P1.

Step 3: The remaining genes in P1 are rearranged in order.

Step 4: A gene is randomly selected from the remaining genes in P1.

Step 5: The genes in the pre-cut section of P1 are added into the offspring in the same order in which they appear in P1.

Step 6: The indices of already-inherited genes from P1 are deleted in the other parent (P2).

Step 7: The remaining genes in P2 are copied into the offspring in the order in which they appear in P2.

Figure 4.5 shows an example of the proposed hybrid crossover operator. The genes (2, 4, 8, 11, and 10) for the second vehicle in Parent 1 are copied to the offspring and deleted from P1 (Steps 1 and 2). Then, remaining genes in P1 are rearranged in the same order, after which the gene (9) is randomly selected as a cutting point (Steps 3 and 4). The genes (1, 5, 7, and 9), the pre-cut section of P1, are added into the offspring in the same order as they appear in P1 (Step 5). The corresponding genes in Parent 2, which are in bold and underlined, are deleted. Finally,

all remaining genes in Parent 2 are copied to the offspring in the same order as in Parent 2 (Step 6 and 7). The genes from the route of the selected vehicle are passed on to the offspring without violating any constraint. Therefore, the good genes from the selected parents can be preserved in the next generation. However, an offspring that may require more vehicles to serve customers can be produced as well.

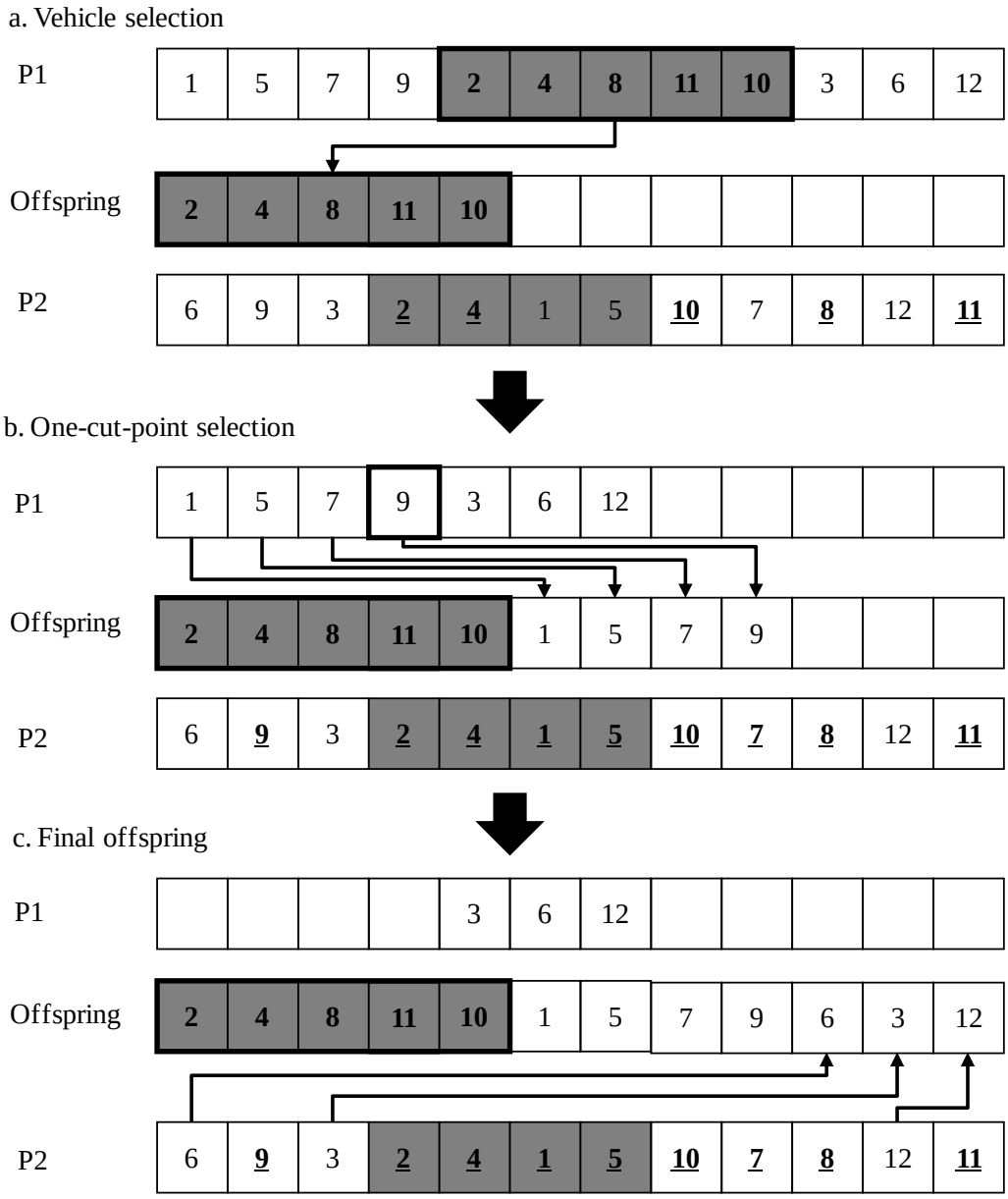



Figure 4.5 The modified crossover operation.

Various crossover operators were tried to obtain the best performance. Overly complicated operators might restrict the random evolutions and take too much computational time. It is thus best to use simple, effective operators. When only the order crossover operator was used in the proposed GA, a premature convergence to local optima was observed. Individuals in the final population have different genetic representations but the same assignment of customers to vehicles. The proposed hybrid crossover operator improves both the inheritance from parents and the diversity of the population, while avoiding the premature convergence of the population.

### 4.3.4 Mutation

For the mutation procedure, an exchange mutation operator is used to prevent individuals in the population from becoming too similar to each other and the GA from settling down at a local optimum. Some individuals are randomly selected at a given probability, called *mutation rate*. The exchange mutation operator selects two genes randomly from an individual and exchanges them. Figure 4.6 shows an example of the exchange mutation. Two genes (7 and 4) are randomly selected in the original individual, and their locations are swapped in the mutant.

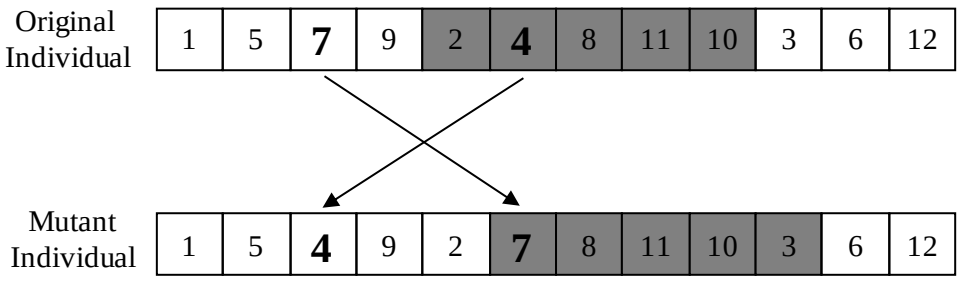



Figure 4.6 An example of the exchange mutation operation.

### 4.3.5 Local search

If the optimal solution of the problem is in the complex solution space, it is sometimes difficult to find good solutions with the GA, which is a general meta-heuristic algorithm. Therefore, in many heuristic algorithms for combinatorial optimization problems, including VRPs, local search methods are employed to improve the solution quality according to the characteristics of the problem and to increase the performance of the algorithms. To improve the quality of solutions for those complex problems, various local search procedures have been studied and implemented along with the GA (Freisleben & Merz, 1996; Prins, 2004).

The proposed algorithm employs the 2-opt exchange local search procedure to improve the routes of each vehicle by untangling the twisted routes. The 2-opt exchange local search procedure is a well-known local search method. The operator of the 2-opt exchange local search procedure searches for a better solution among the neighbors of the current best individual. To probe among the neighbors, the local search operator extracts a set of genes that represent the route of the first vehicle in the best individual. The operator selects two genes in the route and then inverts the sub-genes between these two genes. All possible combinations that can be generated by selecting a pair of genes in the route are considered to find better solutions. When the local search procedure completely finishes searching better routes in a set of genes for the first vehicle, the best route of the first vehicle is copied to a new individual. Then, a set of genes for the route of the next vehicle is considered. Finally, if the fitness value of the new individual produced by the local search operation is better than the fitness value of the best individual in the current population, the new individual is admitted to the population in place of the worst individual in the population. An example of a 2-opt move operation is shown in Figure 4.7.

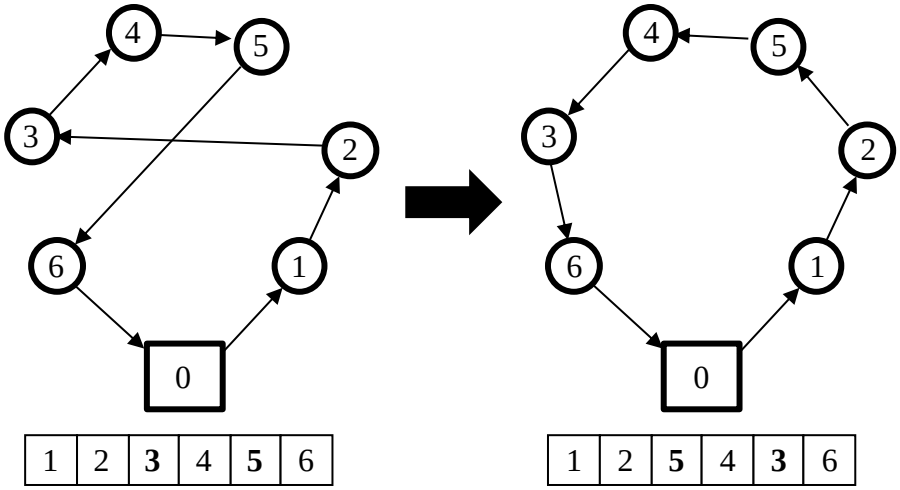



Figure 4.7 The 2-opt exchange local search.

In Figure 4.7, the route of a vehicle is {0, 1, 2, 3, 4, 5, 6, 0}. The proposed local search operation selects genes 3 and 5 and then inverts the sub-genes between two genes. Finally, the proposed local search operation finds a better route for the vehicle, namely {0, 1, 2, 5, 4, 3, 6, 0}.

### 4.3.6 Termination conditions

The proposed GA terminates when the number of generations reaches a specified limit or no improvement of the best solution is observed over a specified number of generations, which is defined as the *improvement interval*. The individual with the highest fitness value in the final generation is interpreted as the best known solution to the problem.

# 5. COMPUTATIONAL EXPERIMENTS OF THE HIERARCHICAL APPROACH USING THE GENETIC ALGORITHM

The proposed hierarchical approach using the genetic algorithm was implemented to effectively solve the VRP under consideration and programmed in Visual Basic programming language with the Microsoft Visual Studio.NET Framework version 1.1. Computational results of the hierarchical approach using the genetic algorithm are compared with results of the MINP model, which was implemented and solved by Lingo version 10.0, a commercially available optimization software for non-linear programming models. Computational experiments were carried out on a personal computer with 3.4 GHz Pentium 4 CPU and 2.0 GB RAM.

## 5.1 Effectiveness of the hierarchical approach using the genetic algorithm

### 5.1.1 Comparison of the MINP approach and the hierarchical approach using the genetic algorithm

In order to show the effectiveness of the proposed hierarchical approach using the genetic algorithm, two test problems were tried by the MINP approach and the hierarchical approach using the genetic algorithm. The results from these two approaches were compared.

Two test problems, V-d6-i3 and V-d8-i4, were randomly generated. In V-d6-i3, there are 6 customers requiring the delivery while 3 customers require the installation as well. In V-d8-i4, there are 8 customers requiring the delivery while 4 customers require the installation as well. Table 5.1 shows the details of the problem parameters used to generate two test problems randomly.

Table 5.1 The problem parameters for two test problems.

| Parameters | V-d6-i3 | V-d8-i4 |
|---|---|---|
| Number of customers | 6 | 8 |
| Number of customers for installation | 3 | 4 |
| Amount of customers' demands | 2-10 | 2-10 |
| Number of delivery vehicles | 3 | 3 |
| Capacity of a delivery vehicle | 20 | 20 |
| Number of installation vehicles | 2 | 2 |
| Service level (min) | 60 | 60 |
| Maximum operation time (min) | 480 | 480 |
| Fixed cost per vehicle (min/vehicle) | 100 | 100 |

In addition, the hierarchical approach using the genetic algorithm has used the algorithmic parameters listed in Table 5.2 to solve test problems and show its effectiveness. The GAs for the subproblems in Stages 1 and 2 used the same parameters during the operation.

Table 5.2 Algorithmic parameters of the genetic algorithm.

| Parameters | Values |
|---|---|
| Size of the population | 100 |
| Crossover rate | 0.8 |
| Mutation rate | 0.1 |
| Maximum number of generations | 5,000 |
| Improvement interval | 200 |

The MINP approach and the hierarchical approach using the genetic algorithm were applied to the two test problems, and the results are given in Table 5.3. The MINP approach solved the test problems to generate optimal solutions, which became the target values to demonstrate the effectiveness of the hierarchical approach using the genetic algorithm. Since the VRP under consideration is an NP-hard problem, the calculation time of the MINP approach would be exponentially increased as the size of the problem grows. In the MINP approach, V-d6-i3 and V-d8-i4 contain 144 and 344 variables, respectively. Lingo takes 452 seconds to obtain the optimal solution of V-d6-i3 and 283 hours to obtain the optimal solution of V-d8-i4. Hence, if the size of the problem grows bigger than V-d8-i4, it would be extremely hard or time-consuming to find the optimal solutions using the MINP approach.

Unlike the MINP approach, the proposed hierarchical approach using the genetic algorithm finds high-quality solutions in a reasonable amount of time. Practically, the hierarchical approach obtains optimal or good solutions of the test problems in just a few seconds. Table 5.3 shows the sum of traveling times of delivery and installation vehicles for two test problems from the MINP approach and the hierarchical approach using the genetic algorithm.

Table 5.3 The results of two approaches for small test problems.

| Problem name | MINP approach with Lingo | The hierarchical approach using GA |
|---|---|---|
| V-d6-i3 | 238.42 | 238.42 |
| V-d8-i4 | 371.47 | 377.01 |

Both approaches provide same results for V-d6-i3 but different results for V-d8-i4. The hierarchical approach using the genetic algorithm found the same solution for V-d6-i3 that the MINP models found, so that the solution is equal to the optimal solution. However, the hierarchical approach using the genetic algorithm obtained a different solution from the optimal solution obtained by the MINP approach. The gap between the solutions from the two approaches may occur because of the characteristics of the hierarchical approach. The final solution of the hierarchical approach consists of two partial solutions from two subproblems in Stages 1 and 2. Since the partial solution of the subproblem in Stage 2 depends on the partial solution of the subproblem in Stage 1, the proposed algorithm may not be able to search for the solutions efficiently. Therefore, the partial solution of the subproblem in Stage 1, which covers the deliveries, is regarded as a local optimum in the original problem. Due to the fact that the hierarchical approach obtains the final solution based on a local optimum, the final solutions for the original problem are not necessarily a global optimum. Unfortunately, this is a natural limitation of the proposed hierarchical approach. For reference, the Lingo program code of the MINP approach for V-d6-i3 is presented in Table A.1 of Appendix A.

### 5.1.2 Computational results for a large problem

The hierarchical approach using the genetic algorithm is proposed to solve the VRP under consideration, with variables of any size, in a reasonable amount of time. This section summarizes computational experiments of the hierarchical approach using the genetic algorithm for a test problem of large size. For this purpose, a test problem of relatively large size, V-d100-i50-a, was randomly generated. In V-d100-i50, there are 100 customers requiring the delivery while 50 customers require the installation as well. All customers are randomly located in a 100x100 bounded square field, and a single depot is located in the center (50, 50) of the field. Other parameters of the test problem and algorithmic parameters of the GAs for the subproblems in Stage 1 and 2 are shown in Table 5.4.

Table 5.4 Problem parameters and algorithmic parameters.

| Parameters | Values |
|---|---|
| Amount of customers' demands | 2-6 |
| Capacity of a delivery vehicle | 20 |
| Service level (min) | 60 |
| Installation time (min) | 10 |
| Fixed cost per vehicle (min/vehicle) | 100 |
| Size of the population | 200 |
| Crossover rate | 0.8 |
| Mutation rate | 0.05 |
| Maximum number of generations | 10,000 |
| Improvement interval | 1000 |

Due to the stochastic properties of the evolutionary algorithm, the average performance of the algorithm is our interest. Therefore, the experiments were performed as described in this paragraph. The GA for the subproblem in Stage 1 solved the test problem five times. Let V-d100-i50-a(1) through V-d100-i50-a(5) be the results of the five runs, respectively. Figure 5.1 shows the progress of the five runs from the GA for the subproblem in Stage 1. Lines in the figure represent the changes of the value of the best solution during the evolution. As the number of generation increases, the value of the best solution of the GA for the subproblem in Stage 1 decreases. As the algorithm evolves from the first to the thousandth generation, the value of the best solution for the algorithm rapidly drops down. Then, the good solution for the subproblem in Stage 1 is found after approximately 2,500 generations.

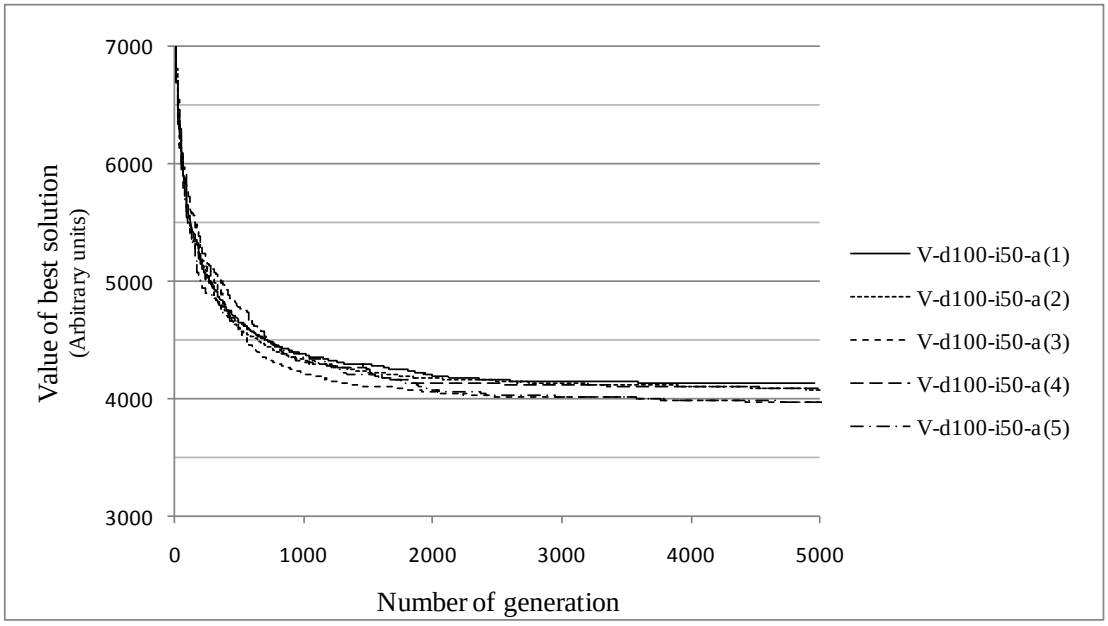



Figure 5.1 Results of the subproblem in Stage 1.

Hence, the GA for the subproblem in Stage 1 obtains five solutions. Each solution contains different arrival times of delivery vehicles at customers' locations. Based on the arrival times

of delivery vehicles in each solution, the time windows for the installations are determined. The time windows are one of the important, unique factors required to synchronize the two types of vehicles; they are fed to the GA for the subproblem in Stage 2 as a part of the input data. Therefore, five different sets of time windows for the customers requiring installation are generated from the results of Stage 1, V-d100-i50-a(1) through V-d100-i50-a(5).

Using the five sets of time windows, five sets of input data for the GA for the subproblem in Stage 2 were prepared and fed into the GA for the subproblem in Stage 2 five times each to again obtain the average performance. Figure 5.2 through 5.6 show the results of the GA for the subproblem in Stage 2 with the five input data. The five lines in each figure represent the five runs of the GA for the subproblem in Stage 2 with a set of input data, which were prepared using a single result from the GA for the subproblem in Stage 1. Lines in each figure represent the changes of the value of the best solution during the evolution. As the number of generations increases, the value of the best solution for the GA for the subproblem in Stage 2 decreases. As the algorithm evolves from the first to the 500th generation, the best fitness value of the algorithm rapidly drops down. A good solution for the subproblem in Stage 2 is found after approximately 700 generations.

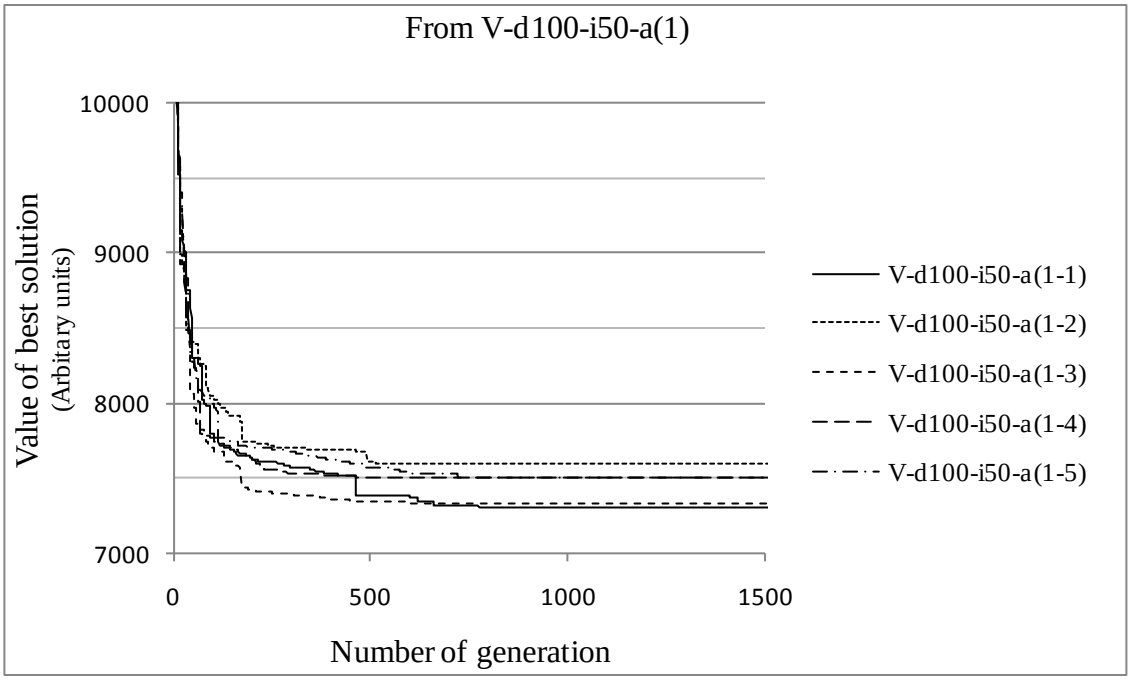



Figure 5.2 Results of the subproblem in Stage 2 based on V-d100-i50-a(1).

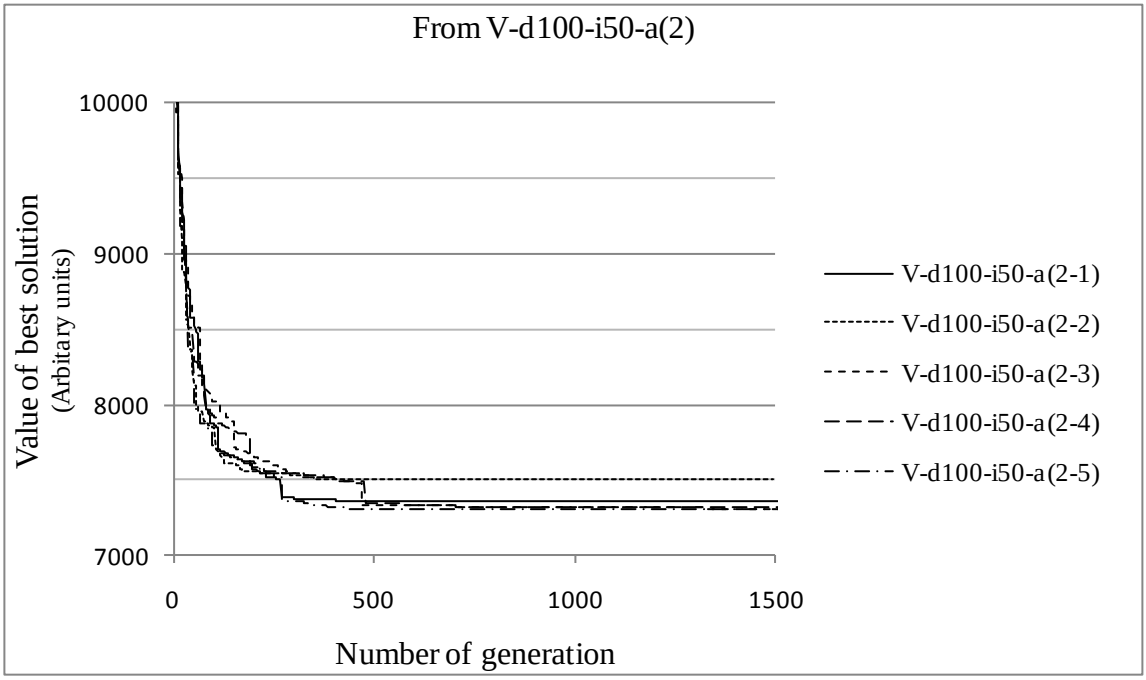



Figure 5.3 Results of the subproblem in Stage 2 based on V-d100-i50-a(2).

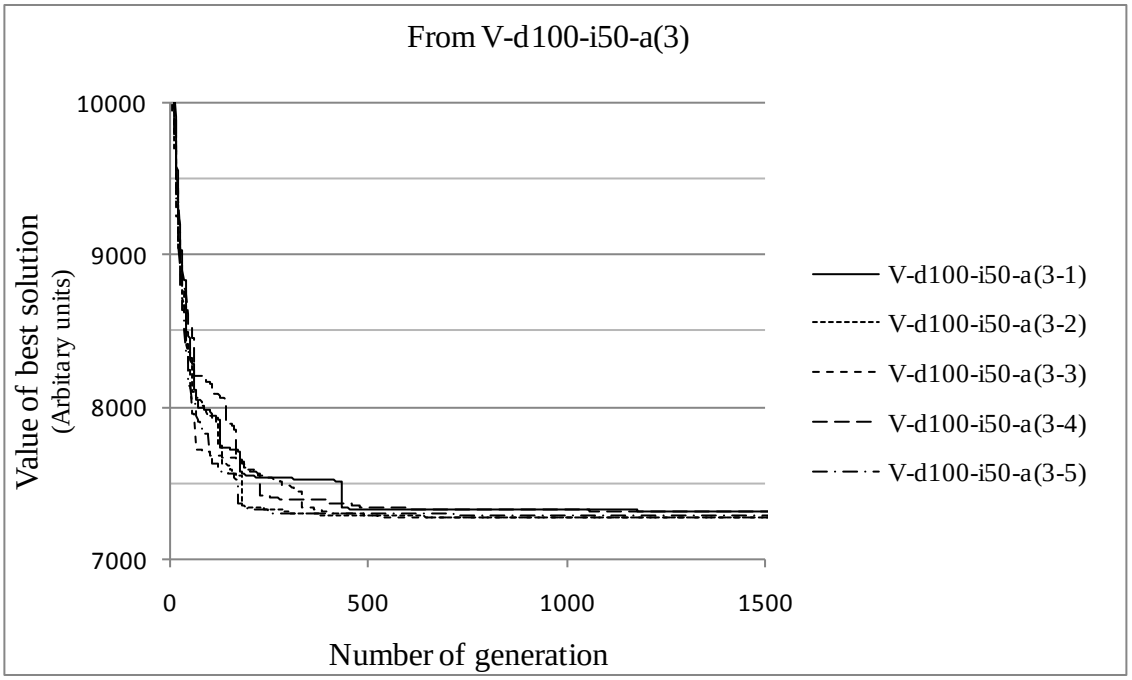



Figure 5.4 Results of the subproblem in Stage 2 based on V-d100-i50-a(3).

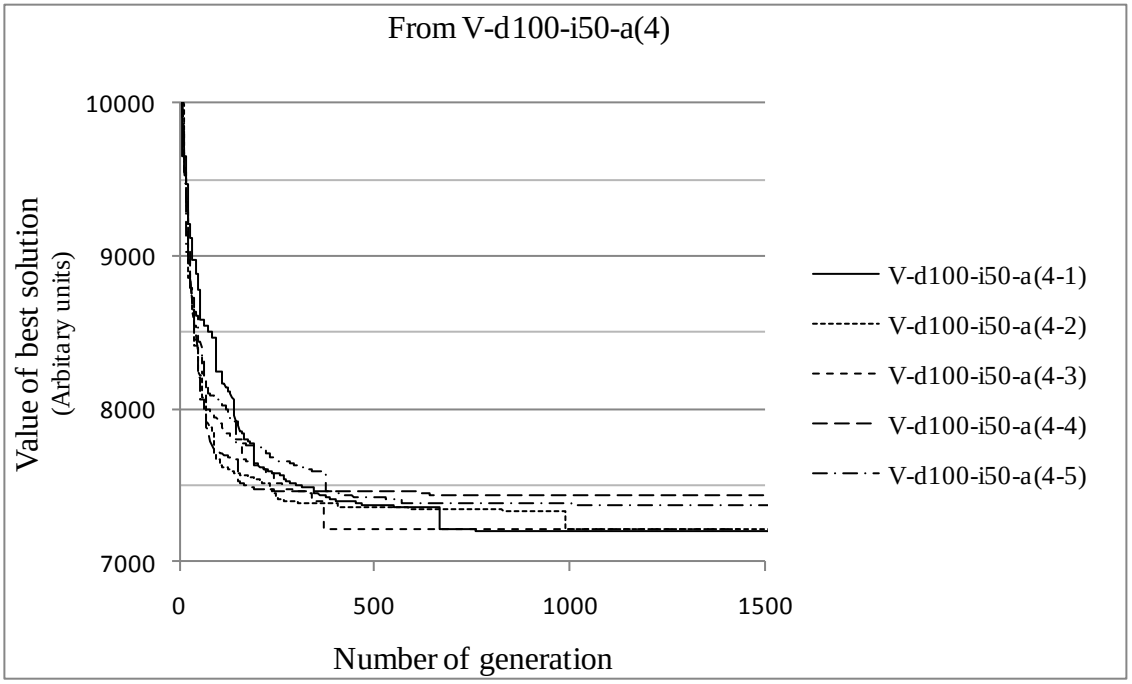



Figure 5.5 Results of the subproblem in Stage 2 based on V-d100-i50-a(4).

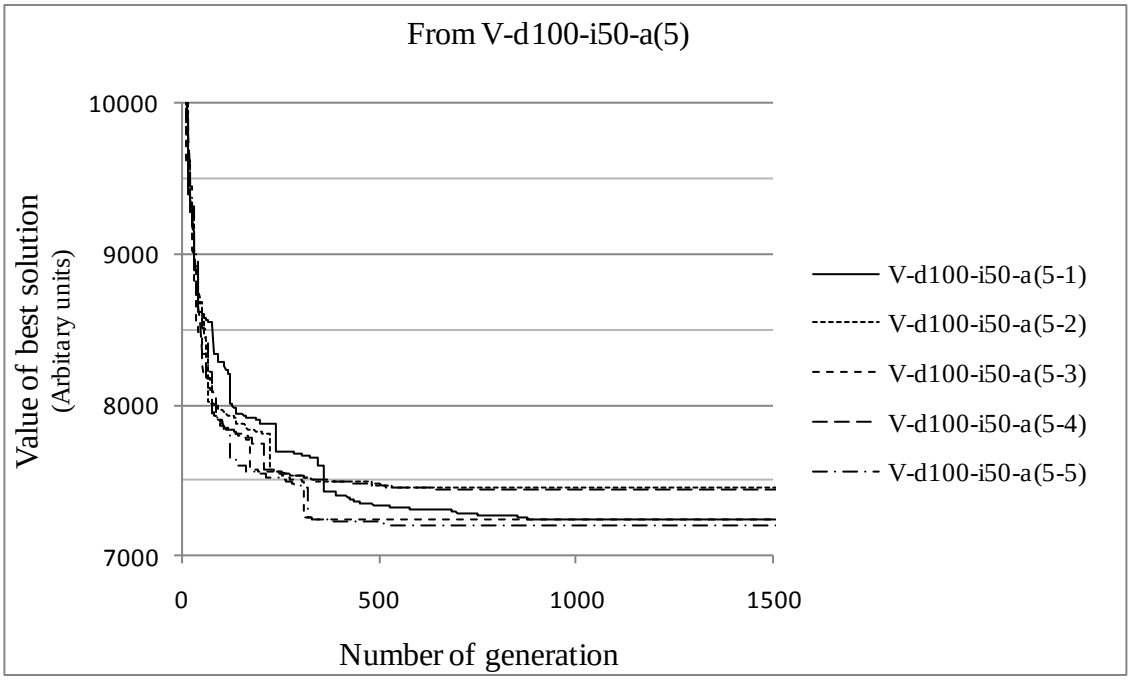



Figure 5.6 Results of the subproblem in Stage 2 based on V-d100-i50-a(5).

Finally, from the hierarchical approach using the genetic algorithm for the subproblems in Stages 1 and 2, 25 final solutions were obtained. For simplicity, Table 5.3 shows a summary of the 25 experimental results. The table consists of final solutions for the subproblems in Stages 1 and 2 and for the original problem, which is naturally the sum of the solutions from both subproblems.

Table 5.5 Results of GAs on V-d100-i50-a.

| Experiments | Subproblem in Stage 1 | Subproblem in Stage 2 | Original problem (Stages 1+ 2) |
|---|---|---|---|
| 1 | 4140.03 | 7304.43 | 11444.46 |
| 2 | | 7591.71 | 11731.74 |
| 3 | | 7334.92 | 11474.95 |
| 4 | | 7505.58 | 11645.61 |
| 5 | | 7496.76 | 11636.79 |
| 6 | 4029.14 | 7355.62 | 11384.76 |
| 7 | | 7509.45 | 11538.59 |
| 8 | | 7307.80 | 11336.94 |
| 9 | | 7323.06 | 11352.20 |
| 10 | | 7308.49 | 11337.63 |
| 11 | 3971.28 | 7313.23 | 11284.51 |
| 12 | | 7276.10 | 11247.38 |
| 13 | | 7270.78 | 11242.06 |
| 14 | | 7272.49 | 11243.77 |
| 15 | | 7288.16 | 11259.44 |
| 16 | 4030.77 | 7206.32 | 11237.09 |
| 17 | | 7215.20 | 11245.97 |
| 18 | | 7218.96 | 11249.73 |
| 19 | | 7433.16 | 11463.93 |
| 20 | | 7370.35 | 11401.12 |
| 21 | 3976.87 | 7245.86 | 11222.73 |
| 22 | | 7457.32 | 11434.19 |
| 23 | | 7237.73 | 11214.60 |
| 24 | | 7433.15 | 11410.02 |
| 25 | | 7204.65 | 11181.52(*) |

(*) The best solution obtained by the proposed hierarchical approach

In Table 5.3, the best solution of the original problem is not equal to the sum of the best solutions for each subproblem in Stages 1 and 2. The best solution value of the subproblem in Stage 1 is 3971.28, the value used for the third set of experiments (experiments 11-15). The routes and schedules of delivery vehicles from this experiment were fed to the subproblem in Stage 2, but the best solution for the original problem did not use this best solution for the subproblem in Stage 1. Rather, the best solution for the original problem was achieved in experiment 25. Considering these computational experiments, it is necessary to develop a way

to consider the tradeoff between the two subproblems in order to obtain the best possible solution of the original problem.

## 5.2 Robustness of the hierarchical approach

In practice, the VRP under consideration would have case-specific problem parameters reflecting the situation. Some problem parameters may have great impacts on the performance and effectiveness of the proposed algorithm as well as on the solutions of the VRP under consideration. The impacts of the problem parameters on the performances and solutions may not be identical, either. Significant degradation in the performance of the proposed algorithm resulting from variation of the problem parameters is not desirable. In other words, the robust algorithm is desirable even if the problem characteristic varies. In order to verify the robustness of the proposed algorithm, the Taguchi method was used to generate an orthogonal combination of the experiments. The purpose of applying the Taguchi method in this section is to show the robustness of the proposed algorithm for the problem with various conditions rather than optimizing the process parameters.

### 5.2.1 Design of experiments by the Taguchi method

Experimental design experiments using the Taguchi method were conducted using the "orthogonal array," within which all experiments are balanced with respect to all control factors while still requiring the minimum number of experiments to be performed. The Taguchi method is a derivative of the fractional factorial design method. In this section, the $L_9$ orthogonal array for four factors with each at three levels is used. The four control factors in

this section are the ratio of installations to deliveries (factor A), the service level (factor B), the installation time (factor C), and the fixed cost per vehicle (factor D). The ratio of installations to deliveries is determined by the customers' demands. Three different ratios are considered; 10%, 30%, and 50%. The service level is generally predetermined by the service policy of the company. The better the quality of service required, the smaller is the permissible service level. It is clear that more installation vehicles will be required to satisfy the demands of customers requring the installation in narrower time windows. Three service levels are considered: 60, 120, and 180 minutes. An identical installation time for all customers requiring the installation is assumed in the VRP under consideration. The installation time may be determined according to the recommended installation time of the products. That is, it is assumed that different amounts of installation times are required for different types of products. However, for simplicity, a single predetermined installation time is applied to all installations in these sample problems. Three different installation times are considered: 10, 35, and 60 minutes. Finally, the fixed cost per vehicle may have an impact on the performance and the solutions as well. When the fixed cost per vehicle is high, reducing the number of vehicles may be more important in order to find the best solution. The fixed cost per vehicle is converted to a certain amount of time as a penalty in the problem modeling. Three different fixed costs per vehicle are used: 50, 100, and 150 minutes per vehicle. Table 5.6 summarizes the four factors and their levels.

Table 5.6 Four factors for the Taguchi method.

| Factor | Level |
|---|---|
| A: Ratio of installations to deliveries (%) | A1 = 10, A2 = 30, A3 = 50 |
| B: Service level (min) | B1 = 60, B2 = 120, B3 = 180 |
| C: Installation time (min) | C1 = 10, C2 = 35, C3 = 60 |
| D: Fixed cost per vehicle (min/vehicle) | D1 = 50, D2 = 100, D3 = 150 |

Based on these four factors and three levels of each factor, an $L_9$ orthogonal array is used to generate the nine sets of experimental conditions. Table 5.7 shows the $L_9$ orthogonal array and nine experimental runs with the experimental conditions.

Table 5.7 The $L_9$ orthogonal array and the nine runs with the experimental conditions.

| $L_9$ ($3^4$) Orthogonal array | | | | | Actual values | | | |
|---|---|---|---|---|---|---|---|---|
| Run no. | A | B | C | D | Ratio of installations to deliveries *(%)* | Service level *(min)* | Installation time *(min)* | Fixed cost per vehicle *(min/vehicle)* |
| 1 | 1 | 1 | 1 | 1 | 10 | 60 | 10 | 50 |
| 2 | 1 | 2 | 2 | 2 | 10 | 120 | 35 | 100 |
| 3 | 1 | 3 | 3 | 3 | 10 | 180 | 60 | 150 |
| 4 | 2 | 1 | 2 | 3 | 30 | 60 | 35 | 150 |
| 5 | 2 | 2 | 3 | 1 | 30 | 120 | 60 | 50 |
| 6 | 2 | 3 | 1 | 2 | 30 | 180 | 10 | 100 |
| 7 | 3 | 1 | 3 | 2 | 50 | 60 | 60 | 100 |
| 8 | 3 | 2 | 1 | 3 | 50 | 120 | 10 | 150 |
| 9 | 3 | 3 | 2 | 1 | 50 | 180 | 35 | 50 |

A = Ratio of installation customers (%), B= Service Level (min)

C=Installation time (min), D= Fixed cost per vehicle (min/vehicle)

In order to generate test problems randomly, the problem parameters listed in Table 5.8 have been used.

Table 5.8 The problem parameters of the test problems in this section.

| Parameters | Values |
|---|---|
| Field size | 100×100 |
| Location of the single depot | (50,50) |
| Locations of customers | Random |
| Amount of demands | 2-10 |
| Loading capacity of a delivery vehicle | 20 |

The hierarchical approach using the genetic algorithm solved the test problem under the identical conditions, using the algorithmic parameters given in Table 5.9. The identical algorithm parameters were applied to the GAs for the subproblems in Stages 1 and 2.

Table 5.9 The algorithm parameters for the experiment.

| Parameters | Value |
|---|---|
| Size of the population | 200 |
| Crossover rate | 0.8 |
| Mutation rate | 0.05 |
| Maximum number of generations | 10,000 |
| Improvement interval | 1,000 |

### 5.2.2 Results of the experiments with regard to robustness

In each run of Table 5.7, five test problems were generated randomly and solved using the identical algorithmic parameters. As mentioned before, the hierarchical approach using the genetic algorithm consists of two stages. It is important to understand that the robustness of the proposed algorithm is proved only when the average performances are robust. Hence, in Stage 1, five test problems were solved five times each. The five test problems (V-d100-i10-a through V-d100-i10-e) were solved five times each for run number 1 in Table 5.7, resulting in 25 solutions. After Stage 1, five solutions per problem were obtained. These solutions were then fed to the GA for the subproblem in Stage 2 as a part of the input data. Then, the GA for the subproblem in Stage 2 solved, five times each, the subproblems based on input data from the end of Stage 1. Thus, each run in Table 5.7 produced 125 solutions. Therefore, 1,125

solutions of the original problem were obtained through the nine experimental runs. The results of these 1,125 solutions are reported in Tables B.1 through B.9 in Appendix B.

The results of the solutions from the experiments are normalized for analysis because the control factors themselves may have a direct impact on the solutions. The sum of the average traveling time per customer requiring delivery from the subproblem in Stage 1 and the average traveling time per customer requiring installation from the subproblem in Stage 2 is used for this Taguchi analysis as the normalized signal. Since the objective of the VRP under consideration is to find the shortest traveling times of all vehicles while satisfying all customers' demands, the smaller result is more desirable in the Taguchi analysis.

Let $y_i$ be the normalized signal from the experiment $i$. The signal-to-noise ratio (SNR) for this smaller-the-better problem is calculated as

$$\text{SNR} = -10 \log_{10} \left( \frac{1}{n} \sum y_i^2 \right)$$

By ANOVA, the relative contributions of each considerable factor are calculated in Table 5.10. The relative contributions of the four control factors used in this Taguchi analysis are fairly similar. The most significant factor is the ratio of customers requiring installation (factor A), which impacts 32.83 % on the results of the experiment. The service level and the installation time impacts 25.37% and 28.97%, respectively, which are not much less than the contribution of the ratio of customers requiring installation. The least significant factor is the fixed cost per vehicle. If all solutions for the same problem contained the same number of vehicles, the contribution of the fixed cost per vehicle must be zero. Therefore, the fact that this factor does

have some impact shows that there are solutions that use different numbers of vehicles. However, the impact of this factor is smaller than that of the other three factors.

It is concluded that the proposed algorithm solves the VRP under consideration consistently well even with the existence of various settings of four control factors, which are problem parameters as well.

Table 5.10 Taguchi analysis for the robustness of the proposed algorithm.

| | Levels | A | B | C | D |
|---|---|---|---|---|---|
| S/N | 1 | -124.524 | -123.877 | -115.747 | -116.934 |
| | 2 | -118.071 | -118.637 | -120.112 | -120.456 |
| | | -116.783 | -116.865 | -123.52 | -121.988 |
| Relative contribution (%) | | 32.83 | 25.37 | 28.98 | 12.81 |

## 5.3 Conclusion of the hierarchical approach using the genetic algorithm

The hierarchical approach using genetic algorithm was proposed to effectively solve the VRP under consideration. The approach divides the original problem into two subproblems. The subproblems were defined as a VRP for delivery vehicles and a VRP for installation vehicles. For each subproblem, an efficient genetic representation, an appropriate method to construct the population, and a set of genetic operators were proposed and developed.

In order to show the effectiveness of the hierarchical approach using the genetic algorithm, the computational results from the hierarchical approach using the genetic algorithm and the MINP model were compared. In the experiment of two small test problems, the hierarchical approach achieved optimality for one test problem and a relatively good solution for the other. It is important to note that a small increase of the problem size resulted in an excessive increase in computational time for the MINP model. The hierarchical approach efficiently found good solutions to the larger test problem in a reasonable amount of time, while the MINP approach could not find even a feasible solution. It is extremely hard to find the optimal solution of a large problem with the MINP approach, even if a great amount of calculation time is provided.

It is also observed that the hierarchical approach exposes its natural limitation. The result of the subproblem in Stage 2, the routes and schedules of installation vehicles, entirely depends on the results of the subproblem in Stage 1, the routes and schedules of delivery vehicles, since the time windows for the installation vehicles' arrival at customers requiring the installation are determined by the results of the subproblem in Stage 1. Due to the hierarchical approach, the best solution of the subproblem in Stage 1 may hinder the subproblem in Stage

2 to find better solutions in the solution space of the original problem. In other words, the best solution of the subproblem in Stage 1 does not guarantee the global optimality. Furthermore, the solution space that can be searched by the hierarchical approach is restricted. In order to consider two subproblems at the same time, an advanced evolutionary algorithm, known as the endosymbiotic evolutionary algorithm, is considered in the next chapter.

# 6. ENDOSYMBIOTIC EVOLUTIONARY ALGORITHM FOR SYNCHRONIZATION OF THE DELIVERY AND THE INSTALLATION

The hierarchical approach using the genetic algorithm was proposed in Chapter 4 to effectively solve the VRP under consideration. However, since subproblems created from the original problem cannot be considered simultaneously during the solution process, the hierarchical approach has a natural flaw. As mentioned in Chapter 4, the hierarchical approach breaks down the original problem into two subproblems and solves them one by one. The subproblem in Stage 1 is solved independently, and then the output from the subproblem in Stage 1 is fed into the subproblem in Stage 2 as a part of the input data. Hence, the solution space where the hierarchical approach can search is restricted by the result of the subproblem in Stage 1. In order to consider the original problem as a whole and to search the solution space with less limitation, an endosymbiotic evolutionary algorithm (EEA) for the VRP under consideration is presented in this chapter.

## 6.1 Endosymbiotic evolutionary algorithm for the vehicle routing problem under consideration

The EEA is one type of symbiotic evolutionary algorithm (SEA) that can consider multiple subproblems at the same time. When the original problem is interwoven by multiple subproblems and we want to solve the original problem as a whole instead of considering them individually, SEAs such as the EEA can be a good option to search for the solutions of multiple subproblems concurrently. The SEAs consider each subproblem along with its

symbiotic partners, which are corresponding subproblems, in the original problem during the evolution. The concurrent consideration of subproblems may reduce the probability that the algorithm dwells on the local optima. The SEAs try to produce good solutions in balance among cooperative subproblems.

The EEA maintains multiple different populations, each of which is composed of a set of corresponding symbiotic partners for each subproblem. The EEA for the VRP under consideration breaks down the original problem into two different subproblems and one combined subproblem; one is the VRP for delivery vehicles, another is the VRP for installation vehicles, and the last is the VRP considering both delivery and installation vehicles. Hence, three distinct populations (POP-D, POP-I, and POP-DI) for two subproblems and a combined subproblem are maintained. The populations for the subproblems regarding delivery and installation vehicles in the proposed EEA are referred as POP-D and POP-I, respectively. These two populations play the roles of corresponding symbionts in the endosymbiotic theory, and individuals in those populations represent partial solutions of the original problem. Both populations evolve in such a direction that corresponding symbionts from both populations cooperate with each other to find the better solutions to the original problem. Since individuals in POP-D and POP-I are merely partial solutions, only when corresponding symbionts are combined appropriately and feasibly into the endosymbionts does evaluation of the solutions to the original problem become possible.

The population POP-DI plays the role of the endosymbiont in this algorithm. An endosymbiont carries the genes of all symbionts, which are partial solutions. Therefore, individuals in POP-DI represent solutions to the original problem. The individuals in POP-DI compete for survival with new offsprings that are generated through the mating of individuals from POP-D and POP-I. Eventually, POP-DI evolves toward a better population that contains

better individuals in terms of the original problem. Figure 6.1 shows the concept of the proposed EEA.

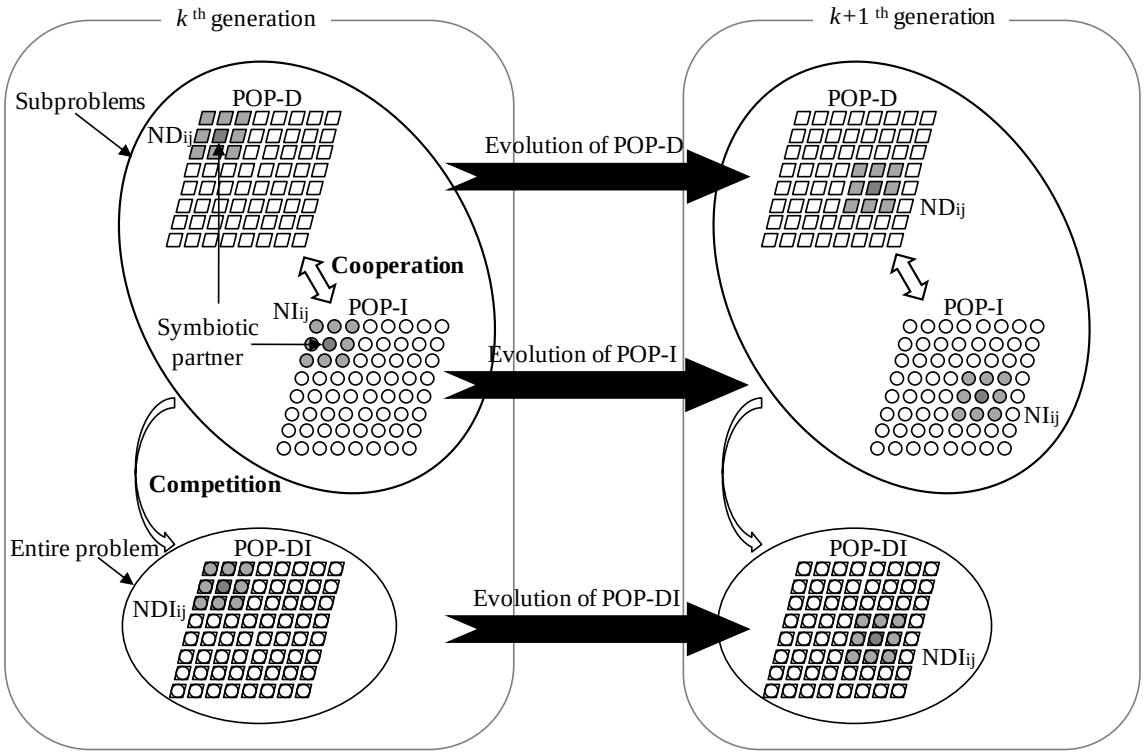



Figure 6.1 The concept of the proposed EEA.

In the algorithm, individuals in POP-D and POP-I are separately working as partial solutions but cooperate with each other while evolving. Furthermore, individuals in POP-DI, representing solutions of the original problem, compete with those created by combination of individuals from POP-D and POP-I. In order to improve the search efficiency, the algorithm uses localized interactions among the populations instead of randomized and scattered interactions. The hierarchical approach does not have any interaction between populations, which work as pools of potential solutions; however, the EEA needs appropriate interactions for the cooperation and the competition among populations to obtain even better solutions.

Thus, topological locations of corresponding individuals or symbionts for the interactions are defined as follows.

Each population forms a two-dimensional structure of toroidal grid with the same number of individuals, and individuals in the population are mapped into the cells of the grid. An individual in the population has its own location index $(x, y)$ and is surrounded by 8 neighbor individuals. The individual has corresponding individuals in the same geographical location at the toroidal grids of other two populations. Hence, when an arbitrary location $(i, j)$ is selected at a generation, the neighborhoods of individuals including $(i, j)$ at center of the 3×3 grid and 8 neighbor individuals in the POP-D, POP-I, and POP-DI are generated as $ND_{ij}$, $NI_{ij}$, and $NDI_{ij}$, respectively. Only the individuals in these three sets of neighborhoods are considered for the interactions among the three populations at the generation. Figure 6.2 shows an example of toroidal grids with individuals at the cells and a neighborhood around individual $(i, j)$.

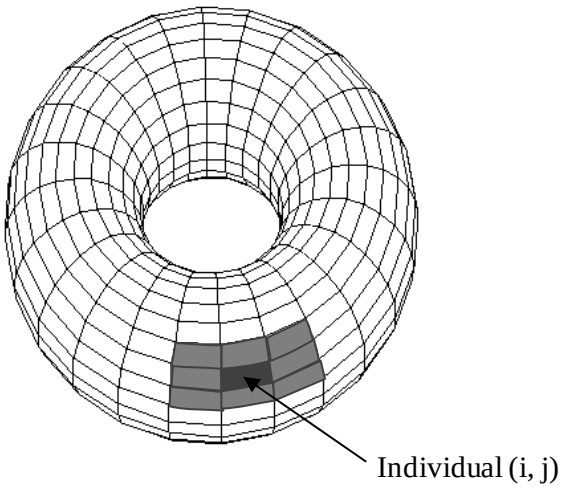



Figure 6.2 A toroidal grid and a neighborhood.

The neighborhoods of $ND_{ij}$ from POP-D and $NI_{ij}$ from POP-I cooperate to find a good solution of the problem. Since each neighborhood contains 9 individuals, 81 (9×9) combinations are considerable as the candidate solutions of the problem. The best combination among them is compared with the current best solution in the algorithm that competes with 9 individuals in $NDI_{ij}$ from POP-DI as well. Based on the interactions among the sets of neighborhoods, the parallel search with partial solutions from POP-D and POP-I in the subproblems and the integrated search with entire solutions from POP-DI in the original problem are carried out simultaneously in all generations. The details of the proposed EEA are described in Table 6.1. In Step 5, populations follow the GA processes explained in the next section.

Table 6.1 The procedures of the proposed EEA.

| | |
|---|---|
| Step 1: | **Initialization:** Generate individuals in of POP-D, POP-I and POP-DI, randomly. Set best solution value, $f_{best} = -\infty$. |
| Step 2: | **Construction of neighbors:** Select an arbitrary location $(i, j)$ and set up the neighborhoods, $ND_{ij}$, $NI_{ij}$, and $NDI_{ij}$, in each population. |
| Step 3: | **Cooperation between subproblems:**<br>Step 3.1: Evaluate the fitness of all possible combinations that can be produced by the concatenation of individuals in $ND_{ij}$ and $NI_{ij}$. Step 3.2: Let $d_p i_p$ be the best combination among those evaluated in Step 3.1, which becomes a candidate endosymbiont. If $f(d_p i_p) > f_{best}$, then update $f_{best} = f(d_p i_p)$ and keep $d_p i_p$ as the current solution. |
| Step 4: | **Competition between entire problem and the best solution:**<br>Step 4.1. Evaluate the fitness of individuals in $NDI_{ij}$ and label the individuals with the best and the worst fitness as $di_v$ and $di_w$, |

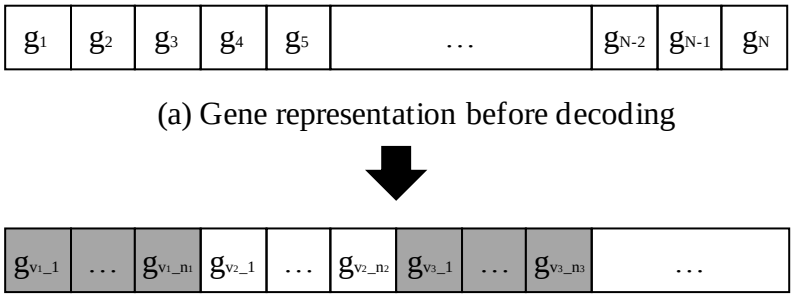
| | |
|---|---|
| | respectively. If $f(di_v) > f_{best}$, then update $f_{best} = f(di_v)$ and keep $di_v$ as the current solution.<br>Step 4.2. If $f(d_p i_p) > f(di_w)$ , then replace $di_w$ with $d_p i_p$ in $NDI_{ij}$. The symbionts, $d_p$ and $i_p$, remain in POP-D and POP-I, respectively. |
| Step 5: | **Evolution:** Perform the evolution of individuals in POP-D, POP-I and POP-DI. |
| Step 6: | If the termination criteria of evolution are met, then stop. Otherwise, go back to Step 2 and repeat the process. |

## 6.2 Genetic representations and operations

As already mentioned, in the proposed EEA not only the population for the entire problem but also the populations for the subproblems evolve to find better solutions through competition and cooperation. The populations in the proposed EEA go through their own GA operations to evolve. A population contains a set of individuals that are generated randomly in a certain genetic representation. Each population might require its unique genetic representation for the corresponding problem. Genetic representations for individuals are very important because they are domain-specific and have large impacts on the performance of the genetic evolution. Genetic representations and genetic operations for the subproblems and the entire problem under consideration in this thesis are described in the following subsections.

### 6.2.1 Genetic representations and operations for the subproblems

Genetic representations for POP-D and POP-I in the proposed EEA use identical genetic representations, i.e., the one-dimensional array, as in the hierarchical approach using the genetic algorithm that is proposed in Section 4.3.1. Figure 6.3 illustrates the genetic representation of an individual for POP-D.




(a) Gene representation before decoding

(b) Gene representation after decoding

Figure 6.3 The genetic representation for POP-D.

Let N be the number of customers for the VRP under consideration and $n_i$ be the number of customers who will be assigned and served by delivery vehicle i. An individual consists of N genes ($g_1$, $g_2$, …, $g_N$) in Figure 6.3(a). The genes contain the indices of customers requiring delivery. Since each customer is to be visited by only one delivery vehicle, the index of a customer must be shown only once in the genetic representation. The decoding procedure determines the routes and schedules of delivery vehicles in operation. In order to assign customers to delivery vehicles, the algorithm's greedy method considers the customers' demand, the loading capacity of delivery vehicles, and the maximum operation time. After the decoding process, the individual is interpreted as Figure 6.3(b). A set of customers, indicated by genes from $g_{v_1\_1}$ to $g_{v_1\_n_1}$, is allocated to the first delivery vehicle ($v_1$). The remaining

customers are assigned to other delivery vehicles in a similar manner. The alternating shades in Figure 6.2(b) represent groups of customers served by different delivery vehicles.

The genetic representation for POP-I is developed in a similar manner. As in the genetic representation for POP-D, a greedy method has been used to assign customers to installation vehicles. However, unlike the genetic representation of the subproblem for the delivery, the greedy method considers the arriving time of delivery vehicles for customers, the installation service time per customer, the service level, and the maximum operation time.

A pair of individuals from POP-D and POP-I is required to make a complete solution for the original problem, because an individual from either subproblem is only a partial solution to the original problem. In the hierarchical approach, the GA for each subproblem has its own fitness function. In other words, there are two fitness functions in the approach; one assesses individuals in the subproblem for the delivery, and the other assesses individuals in the subproblem for the installation. However, the EEA has a single fitness function that requires a completely formed solution for the original problem. Therefore, an individual solution from either subproblem is not complete to be evaluated separately.

For the fitness value of a completely formed solution, which consists of individuals from both POP-D and POP-I, the traveling times of delivery vehicles and their fixed costs are first calculated for the individual from POP-D. While the traveling times of delivery vehicles are calculated, the arrival times of delivery vehicles at customers are also determined. Based on the arrival time of delivery vehicles at customers, the time windows for the arrival of installation vehicles and their fixed costs are calculated for the individual taken from POP-I. Then the traveling schedules of installation vehicles are determined, including the traveling times and waiting times of installation vehicles. Let $\tau_i$ be the sum of traveling times of

delivery vehicles that are included in individual *i* of POP-D; $\varphi_{i,j}$ be the sum of traveling times of installation vehicles in individual *j* of POP-I (which are calculated based on the information on individual *i* of POP-D); $\omega_{i,j}$ be the sum of waiting times of installation vehicles in individual *j* of POP-I (again calculated based on individual *i* of POP-D); $\delta_i$ be the sum of fixed costs of delivery vehicles under operation in individual *i* of POP-D; $\gamma_{i,j}$ be the sum of fixed costs of installation vehicles under operation in individual *j* of POP-I; $\Lambda$ be the sum of unloading times by delivery vehicles at all customers; and I be the sum of installation times at customers requiring the installation. Since traveling times between any pair of customers are known, the traveling times of vehicles can be calculated from vehicles' visiting sequences. As mentioned previously, the waiting time occurs only when an installation vehicle arrives at a customer before a delivery vehicle arrives there, and it is calculated as the time lapse from the arrival of the installation vehicle to the completion of the unloading from the delivery vehicle. Fixed costs of delivery and installation vehicles are also computed by multiplying the known fixed cost per vehicle times the number of vehicles in operation. The sum of unloading times at customers ($\Lambda$) and the sum of installation times at customers (I) can be ignored since they are constants that can be omitted. Therefore, the fitness function for a combination of the individual *i* from POP-D and the individual *j* from POP-I , $\pi_{i,j}$ , is defined as follows:

$$\pi_{i,j} = \frac{1}{\tau_i + \varphi_{i,j} + \omega_{i,j} + \delta_i + \gamma_{i,j}(+\Lambda + \mathrm{I})}$$

The hybrid order crossover operation and the exchange mutation operation, which were described in Sections 4.3 and 4.4, are used for the subproblems of the delivery and the installation. The proposed hybrid order crossover operation has been successfully implemented to effectively reproduce new offsprings for the next generation from two parents

in the current generation. The proposed crossover operation considers both the inheritance from parents to offsprings and the diversity of individuals in the population, while avoiding a premature convergence of the population. The proposed exchange mutation operation prevents the evolution of the EEA from dwelling at potential local optima by generating effective mutants.

### 6.2.2 Genetic representation and operations for the entire problem

This section describes the characteristics of the individuals in POP-DI. An individual in POP-DI is a candidate solution for the original problem and consists of two different genomes; one has an identical form of the individual in POP-D, and the other has an identical form of the individual in POP-I. Therefore, an individual in POP-DI is a completely formed solution for the VRP under consideration in this thesis. Two genomes for the delivery and the installation are concatenated in a row. The genome for the delivery is placed at the head of the individual and is followed by the genome for the installation. An example of individuals for the POP-DI is shown in Figure 6.4. The genome at the head and the genome at the tail use identical genetic representations of the subproblems for the delivery and the installation, which are explained in Section 4.3.1 and Section 6.2.1, respectively.

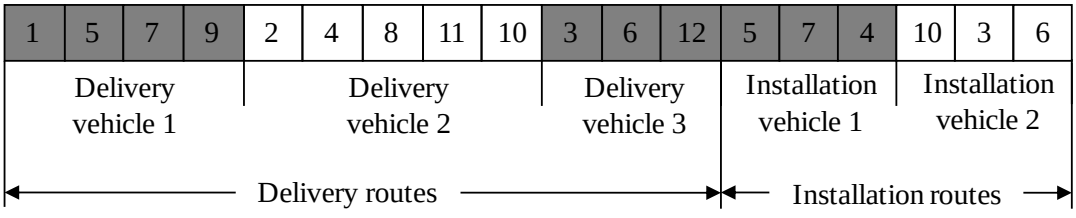



Figure 6.4 The genetic representation for an individual in POP-DI.

The individual shown in Figure 6.4 indicates that the corresponding genetic representation provides the routes for three delivery and two installation vehicles. The concatenation of two different genomes results in a one-dimensional array, but two separate genomes within an individual in POP-DI are heterogeneous. Therefore, the genetic operations for POP-DI are carried out separately in two-fold: one for the head and the other for the tail. The hybrid order crossover operation, which has been explained earlier, is executed over the head and then over the tail. The exchange mutation operation also needs to be performed in two-fold in a similar way. A detailed description of this operation is omitted to avoid redundancy.

The fitness of an individual in POP-DI is evaluated sequentially over the head and the tail. The genome at the head is decoded to generate schedules and routes for delivery vehicles. Based on the schedules for delivery vehicles, the genome at the tail is decoded to generate schedules and routes for installation vehicles. Finally, the integrated schedules and routes for both types of vehicles become the complete solution of the individual in POP-DI. From the schedules and the routes of delivery and installation vehicles, the sum of traveling times of delivery vehicles, the sum of traveling times of installation vehicles, and the sum of waiting times of installation vehicles are calculated. Furthermore, from the results of the decoding procedure, the number of delivery and installation vehicles to be used is also determined.

The evaluation of the fitness values for individuals in POP-DI is conducted in a similar manner to those for POP-D and POP-I in Section 6.2.1. Let $\gamma_{k,}$ be the sum of traveling times of delivery vehicles from the genome at the head of individual $k$, $\varphi_k$ be the sum of traveling times of installation vehicles from the genome at the tail of individual $k$, $\omega_{k,t}$ be the sum of waiting times of installation vehicles in individual $k$, $\delta_k$ be the sum of fixed costs of delivery vehicles in operation in individual $k$, $\gamma_k$ be the sum of fixed costs of installation vehicles in operation in individual $k$, $\Lambda$ be the sum of unloading times by delivery vehicles for all customers, and I be

the sum of installation times at customers requiring installation. Since traveling times between any pairs of customers are known, the traveling times of vehicles can be calculated from vehicles' visiting sequences. As already noted, waiting time occurs only when an installation vehicle arrives at a customer before a delivery vehicle arrives there, and it is calculated as the time lapse from the arrival of the installation vehicle to the completion of the unloading from the delivery vehicle. Fixed costs of delivery and installation vehicles are also computed by multiplying the known fixed cost per vehicle times the number of vehicles in operation. The sum of unloading times at all customers ($\Lambda$) and the sum of installation times at customers requiring the installation (I) are constant and can be omitted. The fitness of the individual *k* in POP-DI ( $\pi_k$ ) is defined as follows.

$$\pi_k = \frac{1}{\tau_k + \varphi_k + \omega_k + \delta_k + \gamma_k (+\Lambda + \mathrm{I})}$$

The fitness function for the original problem is designed to reduce not only the sum of traveling times of all vehicles but also the sum of waiting times of the installation vehicles while evolving.

### 6.2.3 Initialization of the populations

An EEA, which considers the cooperation of its subproblems and the competition among individual evolutions and cooperative evolutions simultaneously, is proposed to more effectively search the solution space of the VRP under consideration than the hierarchical approach. Since the proposed EEA not only conducts an effective and efficient search in global solution space but also considers a balanced evolution of subproblems' solution space,

it requires more calculation time to find high-quality solutions then the hierarchical approach. The fitness evaluation in the proposed EEA requires a complete solution for the problem, which consists of two partial solutions (for the delivery and the installation). The complete solution can be simply chosen from POP-DI or created by the concatenation of partial solutions from POP-D and POP-I. The genetic representations of individuals in the subproblems and the entire problem were already detailed in Sections 4.3.1 and 6.2.1.

Any individual in the algorithm forms a one-dimensional array. An individual in POP-D or the delivery portion of an individual in POP-DI contains a sequence of customer numbers requiring delivery. Since each customer is to be visited by a single delivery vehicle, a customer must be shown only once in the array. The customers' demands, the loading capacity of the vehicle, and the maximum operation time are considered simultaneously to decode the solution of the delivery portion. An individual of POP-I or the installation portion of an individual in POP-DI contains a sequence of customer numbers requiring installation as well. As in the delivery portion, a customer must be shown only once in the solution. The delivery schedules for customers from the delivery portion, the size of time windows for the installation, and the maximum operation time are considered together to decode the solution of the installation portion.

In the hierarchical approach, the subproblem for delivery vehicles is solved by first finding the best possible sequence of customers to be assigned to a set of delivery vehicles, considering the visiting sequence and the loading capacity of vehicles. Then these assignments are fed into the subproblem for installation vehicles as part of the input data. Therefore, the subproblem for installation vehicles is automatically subject to a single set of assignments for delivery vehicles. Due to the nature of this hierarchical approach, the calculation time for each subproblem is proportional to the number of genes in the individuals. Hence, both initial

populations for the two subproblems have been randomly generated without any special consideration.

However, the proposed EEA combines the individuals from POP-D and POP-I (more exactly, from $ND_{ij}$ and $NI_{ij}$) to evaluate their fitness. For each individual in POP-D (or POP-I), the EEA creates and considers a number of combinations simultaneously, along with the corresponding symbiotic partners in POP-I (or POP-D, respectively). Hence, a significant increase in the calculation time for initial evolutions of either POP-D and POP-I is reported. To overcome these time-consuming combinatorial operations in the proposed algorithm, a simple local search is used to produce an initial POP-D with better individuals. This local search generates good individuals with shorter traveling times of delivery vehicles, so the EEA shows a faster convergence toward optimal solutions. It is applied to initial solutions in POP-D and the delivery portion of initial solutions in POP-DI. Initial solutions in POP-I and the installation portion of initial solutions in POP-DI are randomly generated without any local search.

The proposed local search for delivery vehicles proceeds as follows. To generate an initial solution for delivery vehicles, a customer is randomly selected and assigned to a delivery vehicle. From this customer, the nearest customer not already assigned to any delivery vehicle is selected and assigned to the current vehicle of our concern, as far as the vehicle can accommodate the demand of the selected customer and complete all assigned deliveries in the maximum operation time. If the demand of the selected customer exceeds the remaining capacity of the delivery vehicle of concern, the selected customer is assigned to a new vehicle and the process continues until all customers are assigned for a delivery.

The proposed local search for the initialization of the POP-D and POP-DI improves the convergence speed significantly by avoiding the time-consuming search for unnecessary combinations of symbiotic partners in the proposed EEA. The initial solutions created by the proposed local search produce geographical clusters of customers, and its fitness values are higher on average than those of randomly generated individuals.

### 6.2.4 Termination condition

The proposed EEA terminates when the number of generations reaches a specified maximum number. The individual with has the highest fitness in the final generation is interpreted as the best known solution to the problem. The best known solution to the original problem can be obtained from either the best individual in POP-DI or the best combination of two individuals in POP-D and POP-I.

# 7. COMPUTATIONAL EXPERIMENTS FOR THE ENDOSYMBIOTIC EVOLUTIONARY ALGORITHM

The proposed EEA was implemented to solve the VRP under consideration and programmed in Visual Basic language with the Microsoft Visual Studio.NET Framework 1.1 version. Computational results of the EEA were compared with those obtained by the MINP model and the hierarchical approach using the genetic algorithm. The MINP model was implemented and solved by Lingo version 10.0, commercially available optimization software for nonlinear programming models, and the hierarchical approach using the genetic algorithm was programmed in Visual Basic programming language. All computational experiments were carried out on a personal computer with 3.4 GHz Pentium 4 CPU and 2.0 GB RAM.

## 7.1 Effectiveness of the proposed endosymbiotic evolutionary algorithm

The EEA has been proposed to facilitate an efficient search in a wider solution space of the VRP that is the subject of this study, while considering the balance of the two subproblems in the algorithm at the same time. In order to show the effectiveness of the proposed EEA, two test problems previously solved by the MINP approach and the hierarchical approach have been attempted by the proposed EEA as well. The results from three approaches are compared in this section. The MINP approach and the hierarchical approach were described in Chapters 3 and 5, respectively. Furthermore, the proposed EEA provides some results for test problems of larger sizes, which cannot be solved by the MINP approach.

### 7.1.1 Comparison of the results of small problems from the MINP model and the endosymbiotic evolutionary algorithm

As already mentioned, only test problems of small sizes can be solved by the MINP approach. For purposes of comparison, two test problems (V-d6-i3 and V-d8-i4) were attempted using the proposed EEA to show its effectiveness. Those test problems have already been described in Section 5.1.1, where we demonstrated the effectiveness of the hierarchical approach. In V-d6-i3, there are six customers requiring the delivery while three customers require installation as well. In V-d8-i4, there are eight customers requiring the delivery while four customers require installation as well. The details of the problem parameters used to generate two test problems have already been described in Table 5.1 and are thus omitted here. The proposed EEA uses the algorithm parameters listed in Table 7.1 to solve test problems and show its effectiveness. Both genetic operations for POP-D, POP-I and POP-DI use the identical parameters during the operation.

Table 7.1 Parameters of the EEA for the test problems.

| Parameter | Value |
|---|---|
| Size of the population | 100 (10×10) |
| Crossover rate | 0.8 |
| Mutation rate | 0.1 |
| Maximum number of generations | 2,000 |

Since the VRP under consideration is an NP-hard problem, the calculation time using the MINP approach increases exponentially as the size of the problem grows. By way of reference, the MINP approach for V-d8-i4 required 283 hours to obtain the optimal solution even when

the state-of-the-art optimization software, Lingo, was used. However, the proposed EEA finds high-quality solutions in a reasonable amount of time.

Table 7.2 shows the sum of the traveling times of delivery and installation vehicles in the best solutions from the MINP approach, the hierarchical approach, and the proposed EEA. In contrast to the hierarchical approach, the proposed EEA successfully obtains the optimal solutions for both test problems, as only the MINP approach achieved the optimality previously. The hierarchical approach could not achieve the optimality at all times. In this study, there is a gap between the best solutions from the hierarchical approach and the optimal solution for V-d8-i4. This gap may be caused by the natural limitation of the hierarchical approach, as discussed in Section 5.1.1.

Table 7.2 The results of two small test problems.

| Test problem | The MINP model | The proposed EEA | The hierarchical approach |
|---|---|---|---|
| V-d6-i3 | 238.42 | 238.42 | 238.42 |
| V-d8-i4 | 371.47 | 371.47 | 377.01 |

### 7.1.2 Endosymbiotic evolutionary algorithm with larger problems

The EEA is proposed to effectively solve not only small problems but also large problems in a reasonable amount of time. This section summarizes computational experiments using the proposed EEA for problems of medium size (V-d30-i10) and large size (V-d100-i50). In these two test problems, all customers are randomly located in a 100x100 bounded square field and

a single depot is located in the center (50, 50) of the field. Other problem parameters of the test problems and algorithmic parameters of the proposed EEA are shown in Table 7.3. The genetic operations for POP-D, POP-I and POP-DI have used the identical crossover rates and mutation rates throughout the course of the algorithm.

Table 7.3 The parameters of the EEA for problems of larger sizes.

| Parameter | | Value |
|---|---|---|
| Amount of demands | | 2-6 |
| Loading capacity of a delivery vehicle | | 20 |
| Service level (min) | | 60 |
| Installation time (min) | | 10 |
| Fixed cost per vehicle (min/vehicle) | | 100 |
| Size of the population | | 100 (10×10) |
| Crossover rate | | 0.8 |
| Mutation rate | | 0.1 |
| Maximum number of generations | V-d30-i10 | 3,000 |
| | V-d100-i50 | 10,000 |

V-d30-i10 consists of 30 customers requiring the delivery while 10 customers require the installation. Due to the stochastic properties of the evolutionary algorithm, the average performance of the algorithm is our interest. The proposed EEA solved the test problem five times, and the progress of these five trials of the EEA for V-d30-i10 is illustrated in Figure 7.1. Lines in the figure represent the change in the best solutions during the evolution. As the number of generations increases, the score of the best solution of the algorithm decreases. As the proposed EEA evolves through the first 1,000 generations, the score of the best fitness

value of the algorithm drops rapidly. A good solution is found in approximately 1,200 generations.

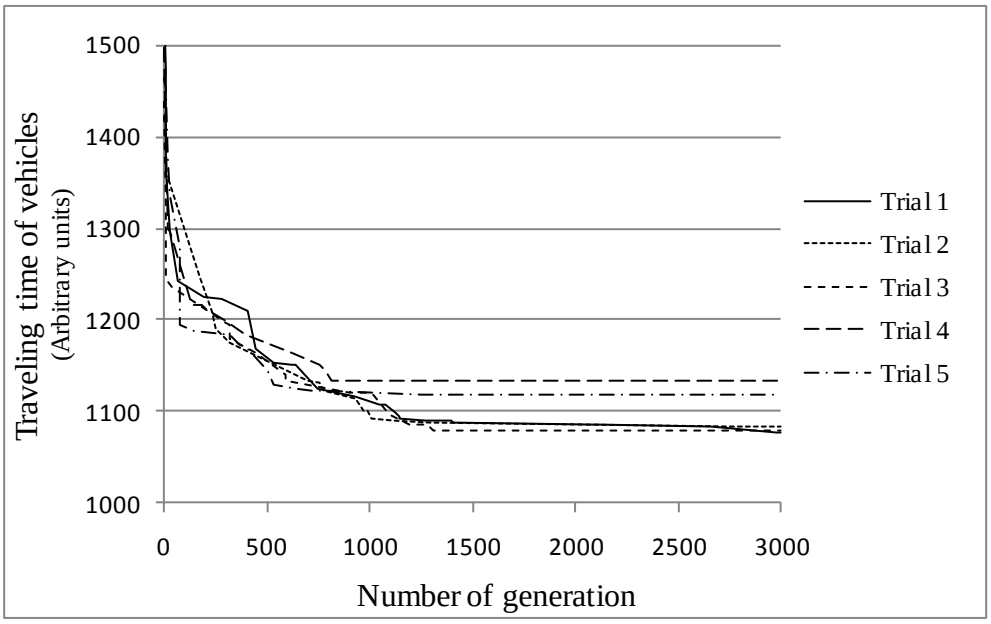



Figure 7.1 Five progresses of the best solutions for V-d30-i10.

Figure 7.2 shows the routes of delivery and installation vehicles as decoded from one of the best solutions by the EEA. This solution requires eight delivery vehicles and three installation vehicles to satisfy all constraints and restrictions of the test problem. The sum of the traveling times of delivery and installation vehicles is 1062.98. The routes from and to the depot are omitted for simplicity in Figure 7.2.

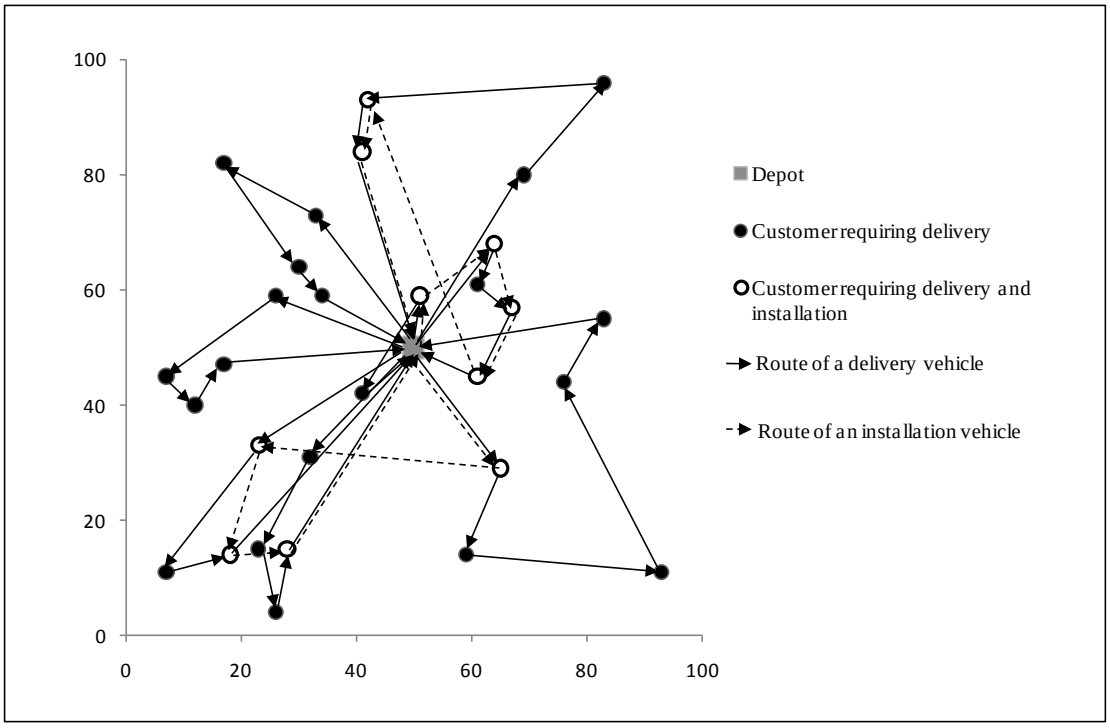



Figure 7.2 Routes of vehicles in the best solution for V-d30-i10.

A test problem of large size, V-d100-i50, consists of 100 customers requiring delivery while 50 customers require installation. The algorithmic parameters in Table 7.3 have been used for this problem as well. In this experiment, the EEA solved the test problem 10 times to obtain an unbiased average performance, since it is more complicated and harder to solve than the smaller problems. The progresses of 10 trials for V-d100-i50 by the proposed EEA are illustrated in Figure 7.3. Lines in the figure represent the change of the value of the best solutions during the evolution. As the algorithm evolves through its first 4,000 generations, the value of the best solution rapidly drops down. In most trials, the proposed EEA found a good solution within approximately 5,000 generations. As Figure 7.3 shows, trial 7 provided the best results among the ten trials.

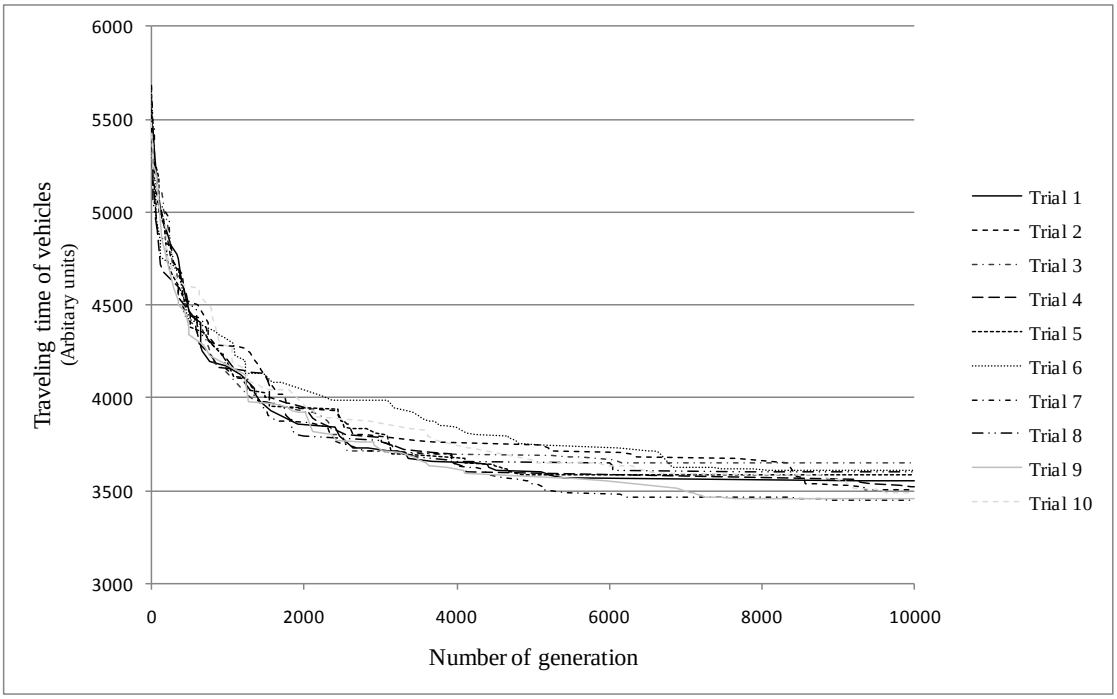



Figure 7.3 Ten results of the test problem V-d100-i50.

Section 5.2.1 showed the results of the hierarchical approach using the genetic algorithm for V-d100-i50. Since the hierarchical approach solves the test problem in two stages, the final solutions for the original problem cannot be obtained until the subproblem in Stage 2 has been completely solved. The EEA and the hierarchical approach have different structures and procedures to solve the VRP under consideration. Hence, in order to compare the performance of the two different approaches, the algorithmic parameters in the hierarchical approach were modified to solve the problem under similar conditions to those of the proposed EEA. Since the EEA for V-d100-i50 finds good solutions within approximately 5,000 generations, the maximum number of generations, as the termination condition for the EEA, is set at 5,000 generations for comparison. The maximum number of generations for each stage in the hierarchical approach is constrained at 2,500 generations, respectively, which seems reasonable considering the computational experiments in Section 5.2.1. The improvement

interval for each subproblem, which is another termination condition for the hierarchical approach, is ignored in the test. The identical crossover rates (0.8) and mutation rates (0.1) are used for both approaches during the operation.

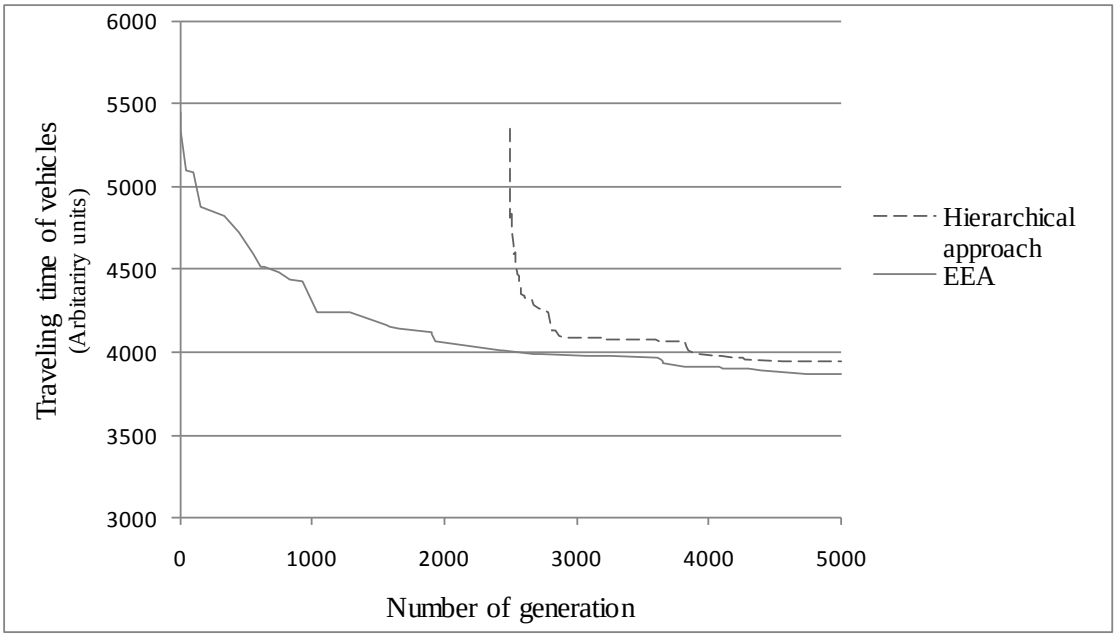



Figure 7.4 Progresses of the EEA and the hierarchical approach for V-d100.

Figure 7.4 shows the progress of the best solutions in each generation for V-d100-i50 from the EEA and the hierarchical approach. Lines in the figure represent the change in the value of the best solution in both approaches during the evolution. Until the 2,500th generation, the hierarchical approach does not have any solution of the original problem, as explained in the previous paragraph. The hierarchical approach searches for a good solution for the subproblem in Stage 1, which is only a partial solution of the original problem. Based on the final partial solution from Stage 1, which is the best solution of Stage 1 at the 2,500th generation, the hierarchical approach obtains solutions for the original test problem. At the 5,000th generation in the hierarchical approach, as shown in Figure 7.4, the final solution of the hierarchical

approach for the test problem is determined as the sum of the partial solutions from Stages 1 and 2. As shown in Figure 7.4, the proposed EEA shows better performance than the hierarchical approach.

## 7.2 Performance of the proposed endosymbiotic evolutionary algorithm

In this section, the computational experiments with problems of various sizes and parameters are reported to show the effectiveness and efficiency of the proposed EEA. The problems of various sizes and problem parameters were randomly generated for testing purposes. The generated problems were solved by the MINP approach, by the hierarchical approach using the genetic algorithm, and finally by the proposed EEA. The computational results from the three approaches are presented and compared in this section.

### 7.2.1 Test problems

All customers are randomly located in a 100x100 bounded square field and a single depot is located in the center (50, 50) of the field. For all test problems, it assumed that the delivery demand of each customer is between 2 and 10; the loading capacity of delivery vehicles is 20; the installation time and the service level are 10 and 60, respectively; and the maximum operation time for both types of vehicles is 480, which can be considered as 8 hours per day. Fifteen test problems, including the four test problems used in previous sections, have been generated to verify the effectiveness and efficiency of the proposed EEA. The numbers of customers and vehicles for each problem are presented in Table 7.4.

Table 7.4 Numbers of customers and vehicles for the test problems.

| Problem size | Problem number | Problem name | Configuration | | | | |
|---|---|---|---|---|---|---|---|
| | | | Number of customers | Number of installation customers | Number of delivery vehicles | Number of installation vehicles | Ratio of installation customers |
| Small | 1 | V-d6-i3 | 6 | 3 | 2 | 1 | 50% |
| | 2 | V-d8-i4 | 8 | 4 | 3 | 2 | 50% |
| | 3 | V-d10-i5 | 10 | 5 | 3 | 2 | 50% |
| | 4 | V-d12-i6 | 12 | 6 | 4 | 3 | 50% |
| Medium | 5 | V-d30-i10 | 30 | 10 | 8 | 3 | 33% |
| | 6 | V-d30-i20 | 30 | 20 | 8 | 4 | 66% |
| | 7 | V-d50-i5 | 50 | 5 | 12 | 2 | 10% |
| | 8 | V-d50-i10 | 50 | 10 | 13 | 3 | 20% |
| | 9 | V-d50-i20 | 50 | 20 | 13 | 5 | 40% |
| Large | 10 | V-d80-i8 | 80 | 8 | 20 | 3 | 10% |
| | 11 | V-d80-i20 | 80 | 20 | 18 | 5 | 25% |
| | 12 | V-d80-i40 | 80 | 40 | 19 | 9 | 50% |
| | 13 | V-d100-i10 | 100 | 10 | 22 | 3 | 10% |
| | 14 | V-d100-i25 | 100 | 25 | 24 | 7 | 25% |
| | 15 | V-d100-i50 | 100 | 50 | 27 | 10 | 50% |

The test problems are classified according to the size of the problem; there are four small problems, five medium-sized problems, and six large problems. Table 7.5 shows the numbers of variables, integer variables, and constraints for the test problems in the MINP model. These numbers can be considered as a rough estimate of the complexity of the problems. It is natural that the number of variables should increase rapidly as the size of the problem grows.

Table 7.5 The number of variables in the MINP model for the test problems.

| Problem size | Problem number | Problem name | Statistics of the MINP model | | |
|---|---|---|---|---|---|
| | | | Number of variables | Number of integer variables | Number of constructions |
| Small | 1 | V-d6-i3 | 144 | 129 | 132 |
| | 2 | V-d8-i4 | 344 | 325 | 293 |
| | 3 | V-d10-i5 | 498 | 475 | 430 |
| | 4 | V-d12-i6 | 916 | 889 | 782 |
| Medium | 5 | V-d30-i10 | 8,374 | 8,321 | 7,688 |
| | 6 | V-d30-i20 | 9,845 | 9,772 | 9,041 |
| | 7 | V-d50-i5 | 31,957 | 31,894 | 30,286 |
| | 8 | V-d50-i10 | 34,929 | 34,856 | 33,068 |
| | 9 | V-d50-i20 | 36,861 | 36,768 | 34,824 |
| Large | 10 | V-d80-i8 | 133,186 | 133,087 | 128,568 |
| | 11 | V-d80-i20 | 121,966 | 121,843 | 117,634 |
| | 12 | V-d80-i40 | 141,831 | 141,668 | 136,550 |
| | 13 | V-d100-i10 | 227,138 | 227,015 | 220,754 |
| | 14 | V-d100-i25 | 252,284 | 252,131 | 244,924 |
| | 15 | V-d100-i50 | 304,840 | 304,637 | 275,427 |

### 7.2.2 Performance comparison

The test problems were solved by the MINP approach, the hierarchical approach, and the proposed EEA. Table 7.6 shows the results of the test problems from these three solution methods. The CPU time using the MINP approach was limited to a maximum of two hours, so that the Lingo finds the feasible solutions or optimal solutions. Lingo was not able to obtain a feasible solution within two hours on all occasions. Since the hierarchical approach and the EEA are meta-heuristic algorithms, they have been tried multiple times for each problem in order to obtain the average performance. In the hierarchical approach, for each problem, the subproblem in Stage 1 is solved four times and then the subproblem in Stage 2 is solved four

times for each result from Stage 1. Hence each test problem has 16 solutions for the hierarchical approach. For the EEA, 16 trials are made to make as much computational effort as the hierarchical approach. The best solutions and the averages of the 16 solutions are summarized in Table 7.6.

Table 7.6 The performance comparison for the MINP approach, the hierarchical approach using the genetic algorithm, and the endosymbiotic evolutionary algorithm.

| Problem | MINP approach | | Hierarchical approach using the genetic algorithm | | | Endosymbiotic evolutionary algorithm | | |
|---|---|---|---|---|---|---|---|---|
| | Best solution | Remark | Best solution | Average solution | Standard deviation | Best solution | Average solution | Standard deviation |
| 1 | 238.42 | Optimal | 238.42 | 238.69 | 0.279 | 238.42 | 238.42 | 0.000 |
| 2 | 381.96 (371.47*) | Feasible | 377.01 | 392.58 | 8.812 | 371.47 | 371.47 | 0.037 |
| 3 | 340.51 | Feasible | 328.12 | 331.80 | 3.801 | 314.89 | 317.49 | 15.149 |
| 4 | 435.77 | Feasible | 365.10 | 368.65 | 3.802 | 350.95 | 356.74 | 32.138 |
| 5 | N/A | Unknown | 1149.30 | 1171.27 | 17.605 | 1081.76 | 1178.65 | 63.296 |
| 6 | N/A | Unknown | 1428.58 | 1466.34 | 32.232 | 1321.83 | 1385.47 | 52.925 |
| 7 | N/A | Unknown | 1607.56 | 1643.53 | 34.632 | 1413.44 | 1449.25 | 31.540 |
| 8 | N/A | Unknown | 1763.66 | 1876.23 | 68.743 | 1617.09 | 1674.83 | 42.214 |
| 9 | N/A | Unknown | 2074.82 | 2100.72 | 22.525 | 1907.06 | 1988.87 | 65.769 |
| 10 | N/A | Unknown | 2552.65 | 2699.44 | 101.219 | 2358.51 | 2454.02 | 75.289 |
| 11 | N/A | Unknown | 2832.26 | 2963.08 | 80.211 | 2537.11 | 2632.88 | 65.175 |
| 12 | N/A | Unknown | 3061.47 | 3203.59 | 91.116 | 2999.86 | 3142.44 | 67.203 |
| 13 | N/A | Unknown | 3045.43 | 3176.53 | 133.936 | 2642.29 | 2713.33 | 47.469 |
| 14 | N/A | Unknown | 3258.53 | 3372.02 | 103.569 | 3140.25 | 3221.60 | 69.835 |
| 15 | N/A | Unknown | 3777.79 | 3957.88 | 144.300 | 3696.68 | 3914.91 | 89.420 |

* the local optimal value found after 283 hours

The MINP approach obtained an optimal solution only for the smallest problem (Problem 1) within two hours. During the two-hour CPU time, the MINP approach was able to identify feasible solutions for Problems 2, 3 and 4. For Problem 2, the best solution found in up to two hours of CPU time is shown, with the optimal solution shown in parenthesis. As mentioned earlier, the optimal solution for Problem 2, the second-smallest one, was obtained after 283

hours of CPU time, which reveals the excessive combinatorial complexity of the problem under consideration. For the larger problems (Problems 5 through 15), the MINP approach failed to find feasible solutions at all.

The hierarchical approach and the EEA obtained optimal or near-optimal solutions for all test problems. In Problem 1, both the hierarchical approach and the EEA obtained the same optimal solution as the MINP approach did. In Problem 2, the EEA found the optimal solution that was obtained by the MINP approach after 283 hours of CPU time. As shown in Table 7.6, the EEA provided better solutions for all problems than the hierarchical approach.

## 7.3 Conclusion of the endosymbiotic evolutionary algorithm

In this chapter, an EEA was proposed to solve the VRP that has the unique characteristics of delivery and installation in the electronics industry. The problem under consideration in this thesis contained two subproblems, each of which is defined as a VRP for delivery vehicles and a VRP for installation vehicles, respectively. The proposed EEA was designed to deal efficiently with both subproblems at the same time. The proposed EEA consists of three populations of the same size for two subproblems and the original problem. The algorithm searches for good solutions through not only cooperation between partial solutions for two subproblems, but also the competition among solutions that are concatenated from the partial solutions for subproblems and solutions for the original problem, while the populations for the two subproblems and the original problem evolve. In order to improve the effectiveness and efficiency of the search, the EEA introduces a concept of neighborhoods in populations and localized interactions among the neighborhoods rather than scattered and wide spread interactions all over the populations. In the EEA, populations for subproblems and the original problem help each other to evolve together toward better solutions. An effective genetic

representation, an initial population construction method, and efficient genetic operations were proposed and developed in this thesis.

The proposed EEA simultaneously considers both the synchronization of delivery and installation vehicles and the minimization of the objective function. In order to show the effectiveness and the efficiency of the proposed EEA, two test problems of small sizes were solved by the MINP approach, the hierarchical approach using the genetic algorithm, and the EEA, and their computational results were compared. The optimal solutions of these test problems were obtained by the MINP approach (though in one case 283 hours of computational time were required). The hierarchical approach achieved optimality for one case, but the EEA obtained the optimal solutions for both, showing better performance than the hierarchical approach.

In the experiments using test problems of medium and large sizes, the EEA consistently produced better solutions than the hierarchical approach. In Section 7.1.2, the convergence speed of the proposed EEA for a test problem of large size was studied and compared with that of the hierarchical approach. Furthermore, test problems of various sizes were solved by three different approaches: the MINP approach, the hierarchical approach using the genetic algorithm, and the EEA. The number of variables in the MINP models for those test problems demonstrated the complexity of the VRP under consideration. Since the proposed EEA deals with subproblems and the original problem at the same time, it can search a wider solution space and obtain better solutions than the hierarchical approach in most test problems.

Based on the computational experiments in this study, it is conjectured that the EEA can be an effective way to solve these very complicated problems, which are interwoven with various subproblems that can be NP-complete or NP-hard themselves.

# 8. CONCLUSIONS

In this thesis, a new type of VRP, drawn from the electronics industry, was identified and introduced. Since material handling has become more and more important in the competitive business environment, electronics manufacturers has extended their involvement in post-sale services, including both delivery and installation, to satisfy various customers' demands. The problem under consideration deals with two types of customers: some require only delivery, while others require both delivery and installation. In order to satisfy both types of demands, two different types of vehicles (delivery and installation vehicles) have been separately operated. Delivery vehicles have a limited loading capacity to carry goods, while installation vehicles do not carry any goods. All vehicles start from a single depot at the beginning and return to the depot within a specific time. Customers are to be visited only once by a single delivery vehicle and (if needed) a single installation vehicle. In addition, there is a service quality requirement, measured by the time lapse between delivery and installation at a customer location. The installation vehicle must visit a customer within the predetermined maximum allowable time after the delivery vehicle's visit to that customer. Therefore, the synchronization of both types of vehicles is needed to ensure the guaranteed quality of service for customers requiring both delivery and installation. Installation vehicles can visit customers earlier than delivery vehicles, resulting in waiting times for installation vehicles at the corresponding customer locations. The minimization of the sum of traveling times of all vehicles, while maintaining adequate synchronization of delivery and installation vehicles, is a main focus in this thesis. Three different approaches have been developed and implemented to solve the problem.

First, the VRP for the delivery and the installation was clearly defined and mathematically formulated by the mixed integer nonlinear programming (MINP) approach. The problem under consideration is by far more complicated than traditional VRPs, which are known as NP-hard problems, since the problem in this thesis contains two VRPs (one for delivery and one for installation vehicles) and the synchronization of both types of vehicles is required besides. In particular, this synchronization has introduced nonlinear constraints in the mathematical modeling. The MINP approach can be used to generate optimal solutions but, due to the complexity of this problem, is able to solve only small test problems whose sizes are eight customers or fewer. The solutions obtained by the MINP approach are provided to assess the effectiveness and performance of other approaches as optimal solutions. Larger test problems cannot be solved within a limited time, since their calculation times exponentially increase as the problem sizes grow.

A hierarchical approach using the genetic algorithm was proposed as the second approach to the problem in this thesis. The approach divides the VRP under consideration into two subproblems: a VRP for delivery vehicles (Stage 1) and a VRP for installation vehicles (Stage 2). After a partial solution for delivery vehicles is obtained from the first subproblem in Stage 1, the other partial solution for installation vehicles is obtained from the second subproblem in Stage 2. Since the schedule of delivery vehicles from the subproblem in Stage 1 is fed as a part of the input data for the second subproblem in Stage 2, the synchronization requirements are automatically achieved during the solution process of the subproblem for installation vehicles in Stage 2. The final solution to the original problem can be formed by concatenation of both partial solutions of the two subproblems. Genetic algorithms are used to find partial solutions of subproblems in both stages.

The proposed hierarchical approach solved test problems of small sizes, and their results were compared with those achieved by the MINP approach to show the effectiveness and efficiency of the hierarchical approach. The optimal or near-optimal solutions of the test problems were found in a few seconds by the hierarchical approach. In addition, the approach also solved test problems of large sizes with computational ease. The GAs in the hierarchical approach quickly obtained high-quality solutions for the subproblems in each stage. According to the computational results for the test problems, the proposed hierarchical approach is able to generate optimal or near-optimal solutions for test problems of various sizes in a reasonable amount of time. However, it was also observed that the hierarchical approach contains a natural limitation. The subproblem for installation vehicles cannot be solved without the result of the subproblem for delivery vehicles. Due to the hierarchical nature of the proposed approach, the partial solution from the subproblem for delivery vehicles may restrict global searches for better solutions in the solution space of the problem under consideration. In other words, the best solution of the subproblem in Stage 1 does not guarantee global optimality. This observation motivated a further search for an approach that could deal with both subproblems at the same time.

As a result of efforts to improve the hierarchical approach, an endosymbiotic evolutionary algorithm (EEA) was proposed and developed as the third approach to the VRP under consideration. The proposed EEA can solve the problems free of the limitation inherent in the hierarchical approach. Unlike the hierarchical approach, the EEA maintains three different populations of potential solutions for two subproblems, defined as the VRP for delivery vehicles, the VRP for installation vehicles, and the original whole problem (consisting of both a solution for delivery vehicles and one for installation vehicles). These three populations cooperate with one another to find a good solution of the problem. Through this cooperation, best combinations of solutions from subproblems are produced as a complete solution to the

original problem and become candidates to be admitted to the population of the original problem. A complete solution consists of two sections, one for delivery vehicles and the other for installation vehicles. It can be created from the concatenation of individuals in the population for subproblems or from the population for the original problem. The newly generated best combination of solutions from symbiotic solutions for subproblems competes with the solutions in the population for the original problem. Through this cooperation and competition, the EEA tries to search for high-quality solutions in a much wider solution space than the hierarchical approach. In order to improve the effectiveness and efficiency of the search, the proposed EEA introduces the concept of neighborhoods in populations and localized interactions among the neighborhoods rather than scattered and widespread interactions among all the solutions in the populations. In the proposed EEA, populations for subproblems and the original problem not only use the localized interactions for cooperation and competition but also help each other to evolve together toward better solutions. An effective genetic representation, an initial population construction method, and efficient genetic operations were proposed and developed.

In order to show the EEA's ability to find optimal or near-optimal solutions, two test problems of small size were solved by the MINP approach, the hierarchical approach using the genetic algorithm, and the EEA. The optimal solutions of the test problems were obtained by the MINP approach. The proposed hierarchical approach was able to obtain the optimal solution for one test problem, while the proposed EEA achieved optimality for both test problems. In subsequent experiments with test problems of medium and large sizes, the EEA consistently produced better solutions than the hierarchical approach. The convergence speed of the proposed EEA for a test problem of large size was studied and compared with that of the hierarchical approach in Section 7.1.2. With a given set of conditions, the proposed EEA showed faster convergence speed than the hierarchical approach.

A set of test problems of various sizes was solved to compare the performance of the proposed algorithms. Through the number of variables in the MINP models for the test problems, the relative difficulty of the problems was indirectly conjectured. Due to the fact that the VRP under consideration is an NP-hard problem, the calculation time for the problems increases exponentially as the problem size grows. Thus, it was impossible for the MINP model to obtain optimal solutions for the test problems of medium and large sizes in a reasonable time. The proposed EEA can search a wider solution space for higher-quality solutions than the hierarchical approach, since the EEA deals with both subproblems at the same time. The results of a set of test problems using all three approaches were compared. For all test problems, the hierarchical approach and the EEA both found good solutions in a reasonable amount of time. In most cases, the EEA provided better solutions than the hierarchical approach.

From the VRP under consideration and the approaches to it examined in this thesis, several issues can be considered for future research. First of all, the MINP approach was developed to find the optimal solution in Chapter 3. However, the calculation time for the approach increases exponentially as the problem size grows, since the VRP is defined as an NP-hard problem. Even when the test problem consisted of eight customers requiring delivery and four customers requiring installation, the MINP approach required unreasonable calculation time to obtain the optimal solution. Containing nonlinear constraints is the most critical reason why the MINP approach required more calculation time for the problem under consideration. Hence, a lower-bounding technique with relaxation of the nonlinear constraints can be an issue to efficiently identify optimal solutions for small-size problems.

Two meta-heuristic algorithms, the hierarchical approach using the genetic algorithm and the EEA, were applied to solve the problem. The algorithms consist of several methods and

operators, such as the genetic representation method, the initial population construction method, the crossover and mutation operators, and so on. The improvement of these components can be considered as a research issue. Different genetic operators for the selection, the crossover, and the mutation operation can be replaced to the algorithms for better performance. In addition, new genetic representation methods can be considered. In this case, appropriate fitness functions and suitable genetic operators for the genetic representation method may be required.

Instead of the hierarchical approach using the genetic algorithm and the EEA, other meta-heuristic algorithms such as the simulated annealing algorithm or ant colony algorithm could be applied. A hybrid algorithm containing multiple meta-heuristic algorithms is another possibility. It may be worthwhile to apply other algorithms and compare their performance with the results attained by the algorithms in this thesis.

Finally, the problem under consideration in this thesis is a special type of VRP with unique characteristics, relevant to the electronics industry. In order to clearly define the VRP, several assumptions and restrictions, described in Chapter 3, were used. By relaxing any of these assumptions and restrictions or by adding new ones, a more realistic model for this type of VRP as a new area of research can be developed. In addition, defining VRPs that represent the unique characteristics of other industries and developing appropriate approaches to these VRPs could be another valuable area of future research.

# BIBLIOGRAPHY

Aarts, E., and Lenstra, J. K. (1996). *Local search in combinatorial optimization.* Springer-Verlag.

Anderson, C., Jones, K., and Ryan, J. (1991). A two-dimensional algorithm for the ising problem. *Complex Systems, 5*, 327-333.

Baker, B. M., and Ayechew, M. A. (2003). A genetic algorithm for the vehicle routing problem. *Computers and Operations Research, 30*, 787-800.

Baker, E. K. (1983). An exact algorithm for the time-constrained traveling salesman problem. *Operation Research, 31*, 938-945.

Berger, J., and Barkaoui, M. (2003). A new hybrid genetic algorithm for the capacitated vehicle routing problem. *Journal of the Operational Research Society, 41*(2), 179-194.

Bodin, L. D., Golden, B. L., and Bender, A. (1983). The state of the art in the routing and scheduling of vehicles and crews. *Computers and Operations Research, 10*(2), 79-116.

Bodin, L. D., Golden, B. L., Assad, A. A., and Ball, M. (1983). Routing and scheduling of vehicles and crews, the state of the art. *Computers and Operations Research, 10*(2), 63-212.

Campbell, N. A., Mitchell, L. G., and Reece, J. B. (1996). *Biology: Concept and connections*, 2nd.ed. Benjamin/Cummings.

Christofides, N. (1985). Vehicle routing. In *The traveling salesman problem: A guided tour of combinatorial optimization* (pp. 431-448). Chichester, UK: John Wiley & Sons.

Christofides, N., Mingozzi, A., and Toth, P. (1979). The vehicle routing problem. In *Combinatorial optimization* (pp. 315-338). Chichester, UK: John Wiley & Sons.

Christofides, N., Mingozzi, A. and Toth, P. (1981). *An algorithm for the time constrained traveling salesman problem.* Report IC-OR, Imperial College of Science and Technology, London.

Chu, P. C., and Beasley, J.E (1998). Constraint handling in genetic algorithms: the set partitioning problem. *Journal of Heuristics*, *4(4)*, 323-357.

Cohoon, P., and Paris, W. (1986). Genetic placement. *Proceedings of the IEEE International Conference on Computer-Aided Design*, 422-425.

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

Dantzig, G. B., and Ramser, J. H. (1959). The truck dispatching problem. *Management Science, 6,* 80-91.

Davis, L. (1987). *Genetic algorithms and simulated annealing.* Pitman.

Desrochers, M., Desrosiers, J. and Solomon, M. (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operation Research, 40,* 342-354.

Desrochers, M., Lenstra, J. K., and Savelsbergh, M. W. P. (1990). A classification scheme for vehicle routing and scheduling problem. *Journal of the Operational Research Society, 46*, 322-332.

Desrosiers, J., Dumas, Y., Solomon, M., and Soumis, F. (1995). Time constrained routing and scheduling. In *Network routing. Handbooks in Operations Research and Management Science, 8*, North Holland.

Dumas, Y., Desrosiers, J., Gelinas, E., and Solomon, M. (1995). An optimal algorithm for the traveling salesman problem with time windows. *Operation Research, 43,* 367-371.

Eksioglu, B., Vural, A. V., and Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers and Industrial Engineering, 57,* 1472-1483.

Eshelman, L., and Schaffer, J. (1993). Real-coded genetic algorithms and interal-schemata. In L. Whitley (ed.), *Foundations of genetic algorithms* (pp. 187-202). San Francisco, CA: Morgan Kaufmann.

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, *2*, 5-30.

Fisher, M. L. (1994). Optimal solution of vehicle routing problems using minimum K-trees. *Operation Research*, *42*(4), 626-642.

Fisher, M. L. (1995). Vehicle routing. In *Network routing*. *Handbooks in Operations Research and Management Science*, *8*, North Holland.

Freisleben, B. and Merz, P. (1996). Genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. *Proceedings of the IEEE Conference on Evolutionary Computation,* 616-621.

Garey, M. R., and Johnson, D. S. (1979*). Computers and intractability: a guide to the theory of NP-completeness,* W.H. Freeman.

Gen, M., and Cheng, R. (2000). *Genetic algorithms and engineering optimization*. John Wiley & Sons.

Golden, B. L., and Addad, A. (1995). The vehicle routing problem. In *Network routing. Handbooks in Operations Research and Management Science*, *8*, North Holland.

Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I. M. (1998). Metaheuristics in vehicle routing. In *Fleet Management and Logistics* (pp. 33-56). Boston, MA: Kluwer.

Homberger, J., and Gehring, H. (2005). A two–phase hybrid meta-heuristic for the vehicle routing problem with time windows. *European Journal of Operation Research, 162,* 220-238.

Jeon, G. W., Leep, H. R., and Shim, J. Y. (2007). A vehicle routing problem solved by using a hybrid genetic algorithm. *Computers and Industrial Engineering, 53,* 680-692.

Kim, J. Y., Kim, Y., and Kim, Y. K. (2001). An endosymbiotic evolutionary algorithm for optimization. *Applied Intelligence, 15*, 117-130.

Kim, Y. K., Kim, J. Y., and Kim, Y. (2006). An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model U-lines. *European Journal of Operational Research, 168,* 838-852.

Kim, Y. K., Park, K., and Ko, J. (2003). A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. *Computers and Operations Research, 30(8),* 1151-1171.

Krumke, S. O., Saliba, S., Vredeveld, T., and Westphal, S. (2008). Approximation algorithms for a vehicle routing problem. *Mathematical Methods of Operations Research, 68,* 333-359.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research, 23,* 631-640.

Laporte, G., Gendreau, M., Potvin, J-Y., and Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research, 7,* 285-300.

Laporte, G., and Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. *Annals of Discrete Mathematics, 31,* 147-184.

Laporte, G., and Semet, F. (1999). Classical heuristics for the vehicle routing problem. *Technical Report G-98-54,* GERAD.

Lim, H. (2007). A genetic algorithm for the vehicle routing problem with heterogeneous vehicles from multiple depots, allowing multiple visits (Unpublished master's thesis). Department of Industrial and Manufacturing Engineering, Oregon State University.

Magnanti, T. L. (1981). Combinatorial optimization and vehicle fleet planning: Perspectives and prospects. *Networks, 11,* 179-214.

Margulis, L., (1981). *Symbiosis in cell evolution*. San Francisco, CA: W. H. Freeman.

Michalewicz, Z., (1996). *Genetic algorithm + data structure = evolution programs,* 3rd ed. New York: Springer-Verlag.

Mole, R. H. (1979). A survey of local delivery vehicle routing methodology. *Journal of the Operational Research Society, 30,* 245-252.

Moriarty, D. E., and Miikkulainen, R. (1997). Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation, 5*, 373-399.

Osman, I. H. (1993a). Metastrategy simulated annealing and Tabu search algorithms for the vehicle routing problem. *Operation Research, 41*, 421-451.

Pereira, F. B., Tavares, J., Machado, P., and Costa, E. (2002). GVR: a new genetic representation for the vehicle routing problem. *Proceedings of the 13th Irish International Conference on Artificial Intelligence and Cognitive Science*, 95-102.

Potter, M. A. (1997). The design and analysis of a computational model of cooperative coevolution. (Unpublished doctoral dissertation). George Mason University, Arlington, VA.

Prescott-Gagnon, E., Desaulniers, G., and Rousseau, L. M. (2009). A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks, 54*, 190-204.

Prive, J., Renaud, J., and Boctor, F. (2006). Solving a vehicle-routing problem arising in soft-drink distribution. *Journal of the Operational Research Society, 57,* 1045-1052.

Rawlins, G. J. E. (1991). *Foundations of genetic algorithms*. Morgan Kaufmann Publishers.

Reeves, C. R. (1993). *Modern heuristic techniques for combinatorial problems*. Oxford, England: Blackwell.

Renaud, J., Boctor, F. F., and Laporte, G. (1996). An improved petal heuristic for the vehicle routing problem. *Journal of the Operational Research Society, 47*, 329-336.

Repoussis, P. P., Tarantilis, C. D., and Ioannou, G. (2007). The open vehicle routing problem with time windows. *Journal of the Operational Research Society, 58*, 355-367.

Ripplinger, D. (2005). Rural school vehicle routing problem. *Transportation Research Record, 1922,* 105-110.

Rochat, Y., and Taillard, E. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics, 1,* 147-167.

Savelsbergh, M. (1985). Local search in routing problems with time windows. *Annual Conference of Operation Research, 4,* 285-305.

Solomon, M. M., and Desrosiers, J. F. (1988). Time window constrained routing and scheduling problems. *Transportation Science, 22,* 1-13.

Taguchi, G., Chowdhury, S., and Taguchi, S. (2000). *Robust engineering*. New York: McGraw-Hill.

Toth, P., and Vigo, D. (1998). Exact algorithms for vehicle routing. In *Fleet Management and Logistics* (pp. 1-31). Boston: Kluwer.

Toth, P., and Vigo, D. (2002). The vehicle routing problem. *SIAM Monographs on Discrete Mathematics and Applications,* Philadelphia, PA: SIAM.

Walters, G. A., and Smith, D. K. (1995). Evolutionary design algorithm for optimal layout of tree networks. *Engineering Optimization, 24,* 261-281.

Wu, Y. (2000). *Taguchi methods for robust design*. New York: ASME.

Yu, B., Yang, Z. Z., and Yao, B. Z. (2011). A hybrid algorithm for vehicle routing problem with time windows. *Expert Systems with Applications, 38,* 435-441.

# APPENDICES

# Appendix A: Mixed-integer nonlinear programming (MINP) model

Table A.1 Mixed-integer nonlinear programming model for the test problem, V-d6-i3

```
MIN =  19.21*X_0_1_1 + 19.21*X_0_1_2 + 9.49*X_0_2_1 + 9.49*X_0_2_2 +
25.63*X_0_3_1 + 25.63*X_0_3_2 + 20.52*X_0_4_1 + 20.52*X_0_4_2 + 12.04*X_0_5_1
+ 12.04*X_0_5_2 + 22.67*X_0_6_1 + 22.67*X_0_6_2 + 19.21*X_1_0_1 +
19.21*X_1_0_2 + 27.66*X_1_2_1 + 27.66*X_1_2_2 + 26.83*X_1_3_1 + 26.83*X_1_3_2
+ 27.02*X_1_4_1 + 27.02*X_1_4_2 + 23.19*X_1_5_1 + 23.19*X_1_5_2 +
41.87*X_1_6_1 + 41.87*X_1_6_2 + 9.49*X_2_0_1 + 9.49*X_2_0_2 + 27.66*X_2_1_1 +
27.66*X_2_1_2 + 33.54*X_2_3_1 + 33.54*X_2_3_2 + 18.03*X_2_4_1 + 18.03*X_2_4_2
+ 18.68*X_2_5_1 + 18.68*X_2_5_2 + 15.23*X_2_6_1 + 15.23*X_2_6_2 +
25.63*X_3_0_1 + 25.63*X_3_0_2 + 26.83*X_3_1_1 + 26.83*X_3_1_2 + 33.54*X_3_2_1
+ 33.54*X_3_2_2 + 45.28*X_3_4_1 + 45.28*X_3_4_2 + 15.03*X_3_5_1 +
15.03*X_3_5_2 + 39.81*X_3_6_1 + 39.81*X_3_6_2 + 20.52*X_4_0_1 + 20.52*X_4_0_2
+ 27.02*X_4_1_1 + 27.02*X_4_1_2 + 18.03*X_4_2_1 + 18.03*X_4_2_2 +
45.28*X_4_3_1 + 45.28*X_4_3_2 + 32.56*X_4_5_1 + 32.56*X_4_5_2 + 31*X_4_6_1 +
31*X_4_6_2 + 12.04*X_5_0_1 + 12.04*X_5_0_2 + 23.19*X_5_1_1 + 23.19*X_5_1_2 +
18.68*X_5_2_1 + 18.68*X_5_2_2 + 15.03*X_5_3_1 + 15.03*X_5_3_2 + 32.56*X_5_4_1
+ 32.56*X_5_4_2 + 25.63*X_5_6_1 + 25.63*X_5_6_2 + 22.67*X_6_0_1 +
22.67*X_6_0_2 + 41.87*X_6_1_1 + 41.87*X_6_1_2 + 15.23*X_6_2_1 + 15.23*X_6_2_2
+ 39.81*X_6_3_1 + 39.81*X_6_3_2 + 31*X_6_4_1 + 31*X_6_4_2 + 25.63*X_6_5_1 +
25.63*X_6_5_2 + 19.21*Y_0_1_1 + 9.49*Y_0_2_1 + 25.63*Y_0_3_1 + 19.21*Y_1_0_1
+ 27.66*Y_1_2_1 + 26.83*Y_1_3_1 + 9.49*Y_2_0_1 + 27.66*Y_2_1_1 +
33.54*Y_2_3_1 + 25.63*Y_3_0_1 + 26.83*Y_3_1_1 + 33.54*Y_3_2_1 + w_1_2 + w_2_2
+ w_3_2 + 100*X_0_1_1 + 100*X_0_1_2 + 100*X_0_2_1 + 100*X_0_2_2 + 100*X_0_3_1
+ 100*X_0_3_2 + 100*X_0_4_1 + 100*X_0_4_2 + 100*X_0_5_1 + 100*X_0_5_2 +
100*X_0_6_1 + 100*X_0_6_2 + 100*Y_0_1_1 + 100*Y_0_2_1 + 100*Y_0_3_1;


X_0_1_1 + X_0_1_2 + X_0_2_1 + X_0_2_2 + X_0_3_1 + X_0_3_2 + X_0_4_1 + X_0_4_2
+ X_0_5_1 + X_0_5_2 + X_0_6_1 + X_0_6_2 <= 2;

Y_0_1_1 + Y_0_2_1 + Y_0_3_1 <= 1;

X_0_1_1 + X_0_2_1 + X_0_3_1 + X_0_4_1 + X_0_5_1 + X_0_6_1 <= 1;
X_0_1_2 + X_0_2_2 + X_0_3_2 + X_0_4_2 + X_0_5_2 + X_0_6_2 <= 1;


X_0_1_1 - X_1_0_1 + X_0_2_1 - X_2_0_1 + X_0_3_1 - X_3_0_1 + X_0_4_1 - X_4_0_1
+ X_0_5_1 - X_5_0_1 + X_0_6_1 - X_6_0_1 = 0;
X_0_1_2 - X_1_0_2 + X_0_2_2 - X_2_0_2 + X_0_3_2 - X_3_0_2 + X_0_4_2 - X_4_0_2
+ X_0_5_2 - X_5_0_2 + X_0_6_2 - X_6_0_2 = 0;


Y_0_1_1 + Y_0_2_1 + Y_0_3_1 <= 1;


Y_0_1_1 - Y_1_0_1 + Y_0_2_1 - Y_2_0_1 + Y_0_3_1 - Y_3_0_1 = 0;


X_1_0_1 + X_1_0_2 + X_1_2_1 + X_1_2_2 + X_1_3_1 + X_1_3_2 + X_1_4_1 + X_1_4_2
+ X_1_5_1 + X_1_5_2 + X_1_6_1 + X_1_6_2 = 1;
X_2_0_1 + X_2_0_2 + X_2_1_1 + X_2_1_2 + X_2_3_1 + X_2_3_2 + X_2_4_1 + X_2_4_2
+ X_2_5_1 + X_2_5_2 + X_2_6_1 + X_2_6_2 = 1;
X_3_0_1 + X_3_0_2 + X_3_1_1 + X_3_1_2 + X_3_2_1 + X_3_2_2 + X_3_4_1 + X_3_4_2
+ X_3_5_1 + X_3_5_2 + X_3_6_1 + X_3_6_2 = 1;
X_4_0_1 + X_4_0_2 + X_4_1_1 + X_4_1_2 + X_4_2_1 + X_4_2_2 + X_4_3_1 + X_4_3_2
+ X_4_5_1 + X_4_5_2 + X_4_6_1 + X_4_6_2 = 1;
X_5_0_1 + X_5_0_2 + X_5_1_1 + X_5_1_2 + X_5_2_1 + X_5_2_2 + X_5_3_1 + X_5_3_2
+ X_5_4_1 + X_5_4_2 + X_5_6_1 + X_5_6_2 = 1;
```

```
X_6_0_1 + X_6_0_2 + X_6_1_1 + X_6_1_2 + X_6_2_1 + X_6_2_2 + X_6_3_1 + X_6_3_2
+ X_6_4_1 + X_6_4_2 + X_6_5_1 + X_6_5_2 = 1;


X_0_1_1 + X_0_1_2 + X_2_1_1 + X_2_1_2 + X_3_1_1 + X_3_1_2 + X_4_1_1 + X_4_1_2
+ X_5_1_1 + X_5_1_2 + X_6_1_1 + X_6_1_2 = 1;
X_0_2_1 + X_0_2_2 + X_1_2_1 + X_1_2_2 + X_3_2_1 + X_3_2_2 + X_4_2_1 + X_4_2_2
+ X_5_2_1 + X_5_2_2 + X_6_2_1 + X_6_2_2 = 1;
X_0_3_1 + X_0_3_2 + X_1_3_1 + X_1_3_2 + X_2_3_1 + X_2_3_2 + X_4_3_1 + X_4_3_2
+ X_5_3_1 + X_5_3_2 + X_6_3_1 + X_6_3_2 = 1;
X_0_4_1 + X_0_4_2 + X_1_4_1 + X_1_4_2 + X_2_4_1 + X_2_4_2 + X_3_4_1 + X_3_4_2
+ X_5_4_1 + X_5_4_2 + X_6_4_1 + X_6_4_2 = 1;
X_0_5_1 + X_0_5_2 + X_1_5_1 + X_1_5_2 + X_2_5_1 + X_2_5_2 + X_3_5_1 + X_3_5_2
+ X_4_5_1 + X_4_5_2 + X_6_5_1 + X_6_5_2 = 1;
X_0_6_1 + X_0_6_2 + X_1_6_1 + X_1_6_2 + X_2_6_1 + X_2_6_2 + X_3_6_1 + X_3_6_2
+ X_4_6_1 + X_4_6_2 + X_5_6_1 + X_5_6_2 = 1;


X_1_0_1 - X_0_1_1 + X_1_2_1 - X_2_1_1 + X_1_3_1 - X_3_1_1 + X_1_4_1 - X_4_1_1
+ X_1_5_1 - X_5_1_1 + X_1_6_1 - X_6_1_1 = 0;
X_2_0_1 - X_0_2_1 + X_2_1_1 - X_1_2_1 + X_2_3_1 - X_3_2_1 + X_2_4_1 - X_4_2_1
+ X_2_5_1 - X_5_2_1 + X_2_6_1 - X_6_2_1 = 0;
X_3_0_1 - X_0_3_1 + X_3_1_1 - X_1_3_1 + X_3_2_1 - X_2_3_1 + X_3_4_1 - X_4_3_1
+ X_3_5_1 - X_5_3_1 + X_3_6_1 - X_6_3_1 = 0;
X_4_0_1 - X_0_4_1 + X_4_1_1 - X_1_4_1 + X_4_2_1 - X_2_4_1 + X_4_3_1 - X_3_4_1
+ X_4_5_1 - X_5_4_1 + X_4_6_1 - X_6_4_1 = 0;
X_5_0_1 - X_0_5_1 + X_5_1_1 - X_1_5_1 + X_5_2_1 - X_2_5_1 + X_5_3_1 - X_3_5_1
+ X_5_4_1 - X_4_5_1 + X_5_6_1 - X_6_5_1 = 0;
X_6_0_1 - X_0_6_1 + X_6_1_1 - X_1_6_1 + X_6_2_1 - X_2_6_1 + X_6_3_1 - X_3_6_1
+ X_6_4_1 - X_4_6_1 + X_6_5_1 - X_5_6_1 = 0;
X_1_0_2 - X_0_1_2 + X_1_2_2 - X_2_1_2 + X_1_3_2 - X_3_1_2 + X_1_4_2 - X_4_1_2
+ X_1_5_2 - X_5_1_2 + X_1_6_2 - X_6_1_2 = 0;
X_2_0_2 - X_0_2_2 + X_2_1_2 - X_1_2_2 + X_2_3_2 - X_3_2_2 + X_2_4_2 - X_4_2_2
+ X_2_5_2 - X_5_2_2 + X_2_6_2 - X_6_2_2 = 0;
X_3_0_2 - X_0_3_2 + X_3_1_2 - X_1_3_2 + X_3_2_2 - X_2_3_2 + X_3_4_2 - X_4_3_2
+ X_3_5_2 - X_5_3_2 + X_3_6_2 - X_6_3_2 = 0;
X_4_0_2 - X_0_4_2 + X_4_1_2 - X_1_4_2 + X_4_2_2 - X_2_4_2 + X_4_3_2 - X_3_4_2
+ X_4_5_2 - X_5_4_2 + X_4_6_2 - X_6_4_2 = 0;
X_5_0_2 - X_0_5_2 + X_5_1_2 - X_1_5_2 + X_5_2_2 - X_2_5_2 + X_5_3_2 - X_3_5_2
+ X_5_4_2 - X_4_5_2 + X_5_6_2 - X_6_5_2 = 0;
X_6_0_2 - X_0_6_2 + X_6_1_2 - X_1_6_2 + X_6_2_2 - X_2_6_2 + X_6_3_2 - X_3_6_2
+ X_6_4_2 - X_4_6_2 + X_6_5_2 - X_5_6_2 = 0;


Y_1_0_1 + Y_1_2_1 + Y_1_3_1 = 1;
Y_2_0_1 + Y_2_1_1 + Y_2_3_1 = 1;
Y_3_0_1 + Y_3_1_1 + Y_3_2_1 = 1;


Y_0_1_1 + Y_2_1_1 + Y_3_1_1 = 1;
Y_0_2_1 + Y_1_2_1 + Y_3_2_1 = 1;
Y_0_3_1 + Y_1_3_1 + Y_2_3_1 = 1;


Y_1_0_1 - Y_0_1_1 + Y_1_2_1 - Y_2_1_1 + Y_1_3_1 - Y_3_1_1 = 0;
Y_2_0_1 - Y_0_2_1 + Y_2_1_1 - Y_1_2_1 + Y_2_3_1 - Y_3_2_1 = 0;
Y_3_0_1 - Y_0_3_1 + Y_3_1_1 - Y_1_3_1 + Y_3_2_1 - Y_2_3_1 = 0;


5* X_0_1_1 + 4* X_0_2_1 + 8* X_0_3_1 + 4* X_0_4_1 + 7* X_0_5_1 + 6* X_0_6_1 +
4* X_1_2_1 + 8* X_1_3_1 + 4* X_1_4_1 + 7* X_1_5_1 + 6* X_1_6_1 + 5* X_2_1_1 +
8* X_2_3_1 + 4* X_2_4_1 + 7* X_2_5_1 + 6* X_2_6_1 + 5* X_3_1_1 + 4* X_3_2_1 +
4* X_3_4_1 + 7* X_3_5_1 + 6* X_3_6_1 + 5* X_4_1_1 + 4* X_4_2_1 + 8* X_4_3_1 +
```

```
7* X_4_5_1 + 6* X_4_6_1 + 5* X_5_1_1 + 4* X_5_2_1 + 8* X_5_3_1 + 4* X_5_4_1 +
6* X_5_6_1 + 5* X_6_1_1 + 4* X_6_2_1 + 8* X_6_3_1 + 4* X_6_4_1 + 7* X_6_5_1
<= 20;
5* X_0_1_2 + 4* X_0_2_2 + 8* X_0_3_2 + 4* X_0_4_2 + 7* X_0_5_2 + 6* X_0_6_2 +
4* X_1_2_2 + 8* X_1_3_2 + 4* X_1_4_2 + 7* X_1_5_2 + 6* X_1_6_2 + 5* X_2_1_2 +
8* X_2_3_2 + 4* X_2_4_2 + 7* X_2_5_2 + 6* X_2_6_2 + 5* X_3_1_2 + 4* X_3_2_2 +
4* X_3_4_2 + 7* X_3_5_2 + 6* X_3_6_2 + 5* X_4_1_2 + 4* X_4_2_2 + 8* X_4_3_2 +
7* X_4_5_2 + 6* X_4_6_2 + 5* X_5_1_2 + 4* X_5_2_2 + 8* X_5_3_2 + 4* X_5_4_2 +
6* X_5_6_2 + 5* X_6_1_2 + 4* X_6_2_2 + 8* X_6_3_2 + 4* X_6_4_2 + 7* X_6_5_2
<= 20;


19.21 * X_0_1_1 + 9.49 * X_0_2_1 + 25.63 * X_0_3_1 + 20.52 * X_0_4_1 + 12.04
* X_0_5_1 + 22.67 * X_0_6_1 + 19.21 * X_1_0_1 + 27.66 * X_1_2_1 + 26.83 *
X_1_3_1 + 27.02 * X_1_4_1 + 23.19 * X_1_5_1 + 41.87 * X_1_6_1 + 9.49 *
X_2_0_1 + 27.66 * X_2_1_1 + 33.54 * X_2_3_1 + 18.03 * X_2_4_1 + 18.68 *
X_2_5_1 + 15.23 * X_2_6_1 + 25.63 * X_3_0_1 + 26.83 * X_3_1_1 + 33.54 *
X_3_2_1 + 45.28 * X_3_4_1 + 15.03 * X_3_5_1 + 39.81 * X_3_6_1 + 20.52 *
X_4_0_1 + 27.02 * X_4_1_1 + 18.03 * X_4_2_1 + 45.28 * X_4_3_1 + 32.56 *
X_4_5_1 + 31 * X_4_6_1 + 12.04 * X_5_0_1 + 23.19 * X_5_1_1 + 18.68 * X_5_2_1
+ 15.03 * X_5_3_1 + 32.56 * X_5_4_1 + 25.63 * X_5_6_1 + 22.67 * X_6_0_1 +
41.87 * X_6_1_1 + 15.23 * X_6_2_1 + 39.81 * X_6_3_1 + 31 * X_6_4_1 + 25.63 *
X_6_5_1 <= 480;
19.21 * X_0_1_2 + 9.49 * X_0_2_2 + 25.63 * X_0_3_2 + 20.52 * X_0_4_2 + 12.04
* X_0_5_2 + 22.67 * X_0_6_2 + 19.21 * X_1_0_2 + 27.66 * X_1_2_2 + 26.83 *
X_1_3_2 + 27.02 * X_1_4_2 + 23.19 * X_1_5_2 + 41.87 * X_1_6_2 + 9.49 *
X_2_0_2 + 27.66 * X_2_1_2 + 33.54 * X_2_3_2 + 18.03 * X_2_4_2 + 18.68 *
X_2_5_2 + 15.23 * X_2_6_2 + 25.63 * X_3_0_2 + 26.83 * X_3_1_2 + 33.54 *
X_3_2_2 + 45.28 * X_3_4_2 + 15.03 * X_3_5_2 + 39.81 * X_3_6_2 + 20.52 *
X_4_0_2 + 27.02 * X_4_1_2 + 18.03 * X_4_2_2 + 45.28 * X_4_3_2 + 32.56 *
X_4_5_2 + 31 * X_4_6_2 + 12.04 * X_5_0_2 + 23.19 * X_5_1_2 + 18.68 * X_5_2_2
+ 15.03 * X_5_3_2 + 32.56 * X_5_4_2 + 25.63 * X_5_6_2 + 22.67 * X_6_0_2 +
41.87 * X_6_1_2 + 15.23 * X_6_2_2 + 39.81 * X_6_3_2 + 31 * X_6_4_2 + 25.63 *
X_6_5_2 <= 480;

19.21 * Y_0_1_1 + W_0 *  Y_0_1_1 + 9.49 * Y_0_2_1 + W_0 *  Y_0_2_1 + 25.63 *
Y_0_3_1 + W_0 *  Y_0_3_1 + 29.21 * Y_1_0_1 + W_1 *  Y_1_0_1 + 37.66 * Y_1_2_1
+ W_1 *  Y_1_2_1 + 36.83 * Y_1_3_1 + W_1 *  Y_1_3_1 + 19.49 * Y_2_0_1 + W_2 *
Y_2_0_1 + 37.66 * Y_2_1_1 + W_2 *  Y_2_1_1 + 43.54 * Y_2_3_1 + W_2 *  Y_2_3_1
+ 35.63 * Y_3_0_1 + W_3 *  Y_3_0_1 + 36.83 * Y_3_1_1 + W_3 *  Y_3_1_1 + 43.54
* Y_3_2_1 + W_3 *  Y_3_2_1 <= 480;

U_1_1 - U_2_1 + 7* X_1_2_1 <= 6;
U_1_2 - U_2_2 + 7* X_1_2_2 <= 6;
U_1_1 - U_3_1 + 7* X_1_3_1 <= 6;
U_1_2 - U_3_2 + 7* X_1_3_2 <= 6;
U_1_1 - U_4_1 + 7* X_1_4_1 <= 6;
U_1_2 - U_4_2 + 7* X_1_4_2 <= 6;
U_1_1 - U_5_1 + 7* X_1_5_1 <= 6;
U_1_2 - U_5_2 + 7* X_1_5_2 <= 6;
U_1_1 - U_6_1 + 7* X_1_6_1 <= 6;
U_1_2 - U_6_2 + 7* X_1_6_2 <= 6;
U_2_1 - U_1_1 + 7* X_2_1_1 <= 6;
U_2_2 - U_1_2 + 7* X_2_1_2 <= 6;
U_2_1 - U_3_1 + 7* X_2_3_1 <= 6;
U_2_2 - U_3_2 + 7* X_2_3_2 <= 6;
U_2_1 - U_4_1 + 7* X_2_4_1 <= 6;
U_2_2 - U_4_2 + 7* X_2_4_2 <= 6;
U_2_1 - U_5_1 + 7* X_2_5_1 <= 6;
U_2_2 - U_5_2 + 7* X_2_5_2 <= 6;
U_2_1 - U_6_1 + 7* X_2_6_1 <= 6;
U_2_2 - U_6_2 + 7* X_2_6_2 <= 6;
U_3_1 - U_1_1 + 7* X_3_1_1 <= 6;
```

```
U_3_2 - U_1_2 + 7* X_3_1_2 <= 6;
U_3_1 - U_2_1 + 7* X_3_2_1 <= 6;
U_3_2 - U_2_2 + 7* X_3_2_2 <= 6;
U_3_1 - U_4_1 + 7* X_3_4_1 <= 6;
U_3_2 - U_4_2 + 7* X_3_4_2 <= 6;
U_3_1 - U_5_1 + 7* X_3_5_1 <= 6;
U_3_2 - U_5_2 + 7* X_3_5_2 <= 6;
U_3_1 - U_6_1 + 7* X_3_6_1 <= 6;
U_3_2 - U_6_2 + 7* X_3_6_2 <= 6;
U_4_1 - U_1_1 + 7* X_4_1_1 <= 6;
U_4_2 - U_1_2 + 7* X_4_1_2 <= 6;
U_4_1 - U_2_1 + 7* X_4_2_1 <= 6;
U_4_2 - U_2_2 + 7* X_4_2_2 <= 6;
U_4_1 - U_3_1 + 7* X_4_3_1 <= 6;
U_4_2 - U_3_2 + 7* X_4_3_2 <= 6;
U_4_1 - U_5_1 + 7* X_4_5_1 <= 6;
U_4_2 - U_5_2 + 7* X_4_5_2 <= 6;
U_4_1 - U_6_1 + 7* X_4_6_1 <= 6;
U_4_2 - U_6_2 + 7* X_4_6_2 <= 6;
U_5_1 - U_1_1 + 7* X_5_1_1 <= 6;
U_5_2 - U_1_2 + 7* X_5_1_2 <= 6;
U_5_1 - U_2_1 + 7* X_5_2_1 <= 6;
U_5_2 - U_2_2 + 7* X_5_2_2 <= 6;
U_5_1 - U_3_1 + 7* X_5_3_1 <= 6;
U_5_2 - U_3_2 + 7* X_5_3_2 <= 6;
U_5_1 - U_4_1 + 7* X_5_4_1 <= 6;
U_5_2 - U_4_2 + 7* X_5_4_2 <= 6;
U_5_1 - U_6_1 + 7* X_5_6_1 <= 6;
U_5_2 - U_6_2 + 7* X_5_6_2 <= 6;
U_6_1 - U_1_1 + 7* X_6_1_1 <= 6;
U_6_2 - U_1_2 + 7* X_6_1_2 <= 6;
U_6_1 - U_2_1 + 7* X_6_2_1 <= 6;
U_6_2 - U_2_2 + 7* X_6_2_2 <= 6;
U_6_1 - U_3_1 + 7* X_6_3_1 <= 6;
U_6_2 - U_3_2 + 7* X_6_3_2 <= 6;
U_6_1 - U_4_1 + 7* X_6_4_1 <= 6;
U_6_2 - U_4_2 + 7* X_6_4_2 <= 6;
U_6_1 - U_5_1 + 7* X_6_5_1 <= 6;
U_6_2 - U_5_2 + 7* X_6_5_2 <= 6;
V_1_1 - V_2_1 + 4* Y_1_2_1 <= 3;
V_1_1 - V_3_1 + 4* Y_1_3_1 <= 3;
V_2_1 - V_1_1 + 4* Y_2_1_1 <= 3;
V_2_1 - V_3_1 + 4* Y_2_3_1 <= 3;
V_3_1 - V_1_1 + 4* Y_3_1_1 <= 3;
V_3_1 - V_2_1 + 4* Y_3_2_1 <= 3;

e_0 = 0;
f_0 = 0;
w_0 = 0;

19.21* X_0_1_1 + e_0 * X_0_1_1 + 19.21* X_0_1_2 + e_0 * X_0_1_2 + 27.66*
X_2_1_1 + e_2 * X_2_1_1 + 27.66* X_2_1_2 + e_2 * X_2_1_2 + 26.83* X_3_1_1 +
e_3 * X_3_1_1 + 26.83* X_3_1_2 + e_3 * X_3_1_2 + 27.02* X_4_1_1 + e_4 *
X_4_1_1 + 27.02* X_4_1_2 + e_4 * X_4_1_2 + 23.19* X_5_1_1 + e_5 * X_5_1_1 +
23.19* X_5_1_2 + e_5 * X_5_1_2 + 41.87* X_6_1_1 + e_6 * X_6_1_1 + 41.87*
X_6_1_2 + e_6 * X_6_1_2 - e_1 = 0;
9.49* X_0_2_1 + e_0 * X_0_2_1 + 9.49* X_0_2_2 + e_0 * X_0_2_2 + 27.66*
X_1_2_1 + e_1 * X_1_2_1 + 27.66* X_1_2_2 + e_1 * X_1_2_2 + 33.54* X_3_2_1 +
e_3 * X_3_2_1 + 33.54* X_3_2_2 + e_3 * X_3_2_2 + 18.03* X_4_2_1 + e_4 *
X_4_2_1 + 18.03* X_4_2_2 + e_4 * X_4_2_2 + 18.68* X_5_2_1 + e_5 * X_5_2_1 +
18.68* X_5_2_2 + e_5 * X_5_2_2 + 15.23* X_6_2_1 + e_6 * X_6_2_1 + 15.23*
X_6_2_2 + e_6 * X_6_2_2 - e_2 = 0;
25.63* X_0_3_1 + e_0 * X_0_3_1 + 25.63* X_0_3_2 + e_0 * X_0_3_2 + 26.83*
```

```
X_1_3_1 + e_1 * X_1_3_1 + 26.83* X_1_3_2 + e_1 * X_1_3_2 + 33.54* X_2_3_1 +
e_2 * X_2_3_1 + 33.54* X_2_3_2 + e_2 * X_2_3_2 + 45.28* X_4_3_1 + e_4 *
X_4_3_1 + 45.28* X_4_3_2 + e_4 * X_4_3_2 + 15.03* X_5_3_1 + e_5 * X_5_3_1 +
15.03* X_5_3_2 + e_5 * X_5_3_2 + 39.81* X_6_3_1 + e_6 * X_6_3_1 + 39.81*
X_6_3_2 + e_6 * X_6_3_2 - e_3 = 0;
20.52* X_0_4_1 + e_0 * X_0_4_1 + 20.52* X_0_4_2 + e_0 * X_0_4_2 + 27.02*
X_1_4_1 + e_1 * X_1_4_1 + 27.02* X_1_4_2 + e_1 * X_1_4_2 + 18.03* X_2_4_1 +
e_2 * X_2_4_1 + 18.03* X_2_4_2 + e_2 * X_2_4_2 + 45.28* X_3_4_1 + e_3 *
X_3_4_1 + 45.28* X_3_4_2 + e_3 * X_3_4_2 + 32.56* X_5_4_1 + e_5 * X_5_4_1 +
32.56* X_5_4_2 + e_5 * X_5_4_2 + 31* X_6_4_1 + e_6 * X_6_4_1 + 31* X_6_4_2 +
e_6 * X_6_4_2 - e_4 = 0;
12.04* X_0_5_1 + e_0 * X_0_5_1 + 12.04* X_0_5_2 + e_0 * X_0_5_2 + 23.19*
X_1_5_1 + e_1 * X_1_5_1 + 23.19* X_1_5_2 + e_1 * X_1_5_2 + 18.68* X_2_5_1 +
e_2 * X_2_5_1 + 18.68* X_2_5_2 + e_2 * X_2_5_2 + 15.03* X_3_5_1 + e_3 *
X_3_5_1 + 15.03* X_3_5_2 + e_3 * X_3_5_2 + 32.56* X_4_5_1 + e_4 * X_4_5_1 +
32.56* X_4_5_2 + e_4 * X_4_5_2 + 25.63* X_6_5_1 + e_6 * X_6_5_1 + 25.63*
X_6_5_2 + e_6 * X_6_5_2 - e_5 = 0;
22.67* X_0_6_1 + e_0 * X_0_6_1 + 22.67* X_0_6_2 + e_0 * X_0_6_2 + 41.87*
X_1_6_1 + e_1 * X_1_6_1 + 41.87* X_1_6_2 + e_1 * X_1_6_2 + 15.23* X_2_6_1 +
e_2 * X_2_6_1 + 15.23* X_2_6_2 + e_2 * X_2_6_2 + 39.81* X_3_6_1 + e_3 *
X_3_6_1 + 39.81* X_3_6_2 + e_3 * X_3_6_2 + 31* X_4_6_1 + e_4 * X_4_6_1 + 31*
X_4_6_2 + e_4 * X_4_6_2 + 25.63* X_5_6_1 + e_5 * X_5_6_1 + 25.63* X_5_6_2 +
e_5 * X_5_6_2 - e_6 = 0;

19.21 * Y_0_1_1 + f_0 * Y_0_1_1 + w_0 * Y_0_1_1 + 37.66 * Y_2_1_1 + f_2 *
Y_2_1_1 + w_2 * Y_2_1_1 + 36.83 * Y_3_1_1 + f_3 * Y_3_1_1 + w_3 * Y_3_1_1 -
f_1 = 0;
9.49 * Y_0_2_1 + f_0 * Y_0_2_1 + w_0 * Y_0_2_1 + 37.66 * Y_1_2_1 + f_1 *
Y_1_2_1 + w_1 * Y_1_2_1 + 43.54 * Y_3_2_1 + f_3 * Y_3_2_1 + w_3 * Y_3_2_1 -
f_2 = 0;
25.63 * Y_0_3_1 + f_0 * Y_0_3_1 + w_0 * Y_0_3_1 + 36.83 * Y_1_3_1 + f_1 *
Y_1_3_1 + w_1 * Y_1_3_1 + 43.54 * Y_2_3_1 + f_2 * Y_2_3_1 + w_2 * Y_2_3_1 -
f_3 = 0;

f_1 - e_1 <= 60;
f_2 - e_2 <= 60;
f_3 - e_3 <= 60;

w_1 >= 0;
e_1 - f_1 - w_1 <= 0;
w_2 >= 0;
e_2 - f_2 - w_2 <= 0;
w_3 >= 0;
e_3 - f_3 - w_3 <= 0;



@BIN ( X_0_1_1 );
@BIN ( X_0_1_2 );
@BIN ( X_0_2_1 );
@BIN ( X_0_2_2 );
@BIN ( X_0_3_1 );
@BIN ( X_0_3_2 );
@BIN ( X_0_4_1 );
@BIN ( X_0_4_2 );
@BIN ( X_0_5_1 );
@BIN ( X_0_5_2 );
@BIN ( X_0_6_1 );
@BIN ( X_0_6_2 );
@BIN ( X_1_0_1 );
@BIN ( X_1_0_2 );
@BIN ( X_1_2_1 );
@BIN ( X_1_2_2 );
```

```
@BIN ( X_1_3_1 );
@BIN ( X_1_3_2 );
@BIN ( X_1_4_1 );
@BIN ( X_1_4_2 );
@BIN ( X_1_5_1 );
@BIN ( X_1_5_2 );
@BIN ( X_1_6_1 );
@BIN ( X_1_6_2 );
@BIN ( X_2_0_1 );
@BIN ( X_2_0_2 );
@BIN ( X_2_1_1 );
@BIN ( X_2_1_2 );
@BIN ( X_2_3_1 );
@BIN ( X_2_3_2 );
@BIN ( X_2_4_1 );
@BIN ( X_2_4_2 );
@BIN ( X_2_5_1 );
@BIN ( X_2_5_2 );
@BIN ( X_2_6_1 );
@BIN ( X_2_6_2 );
@BIN ( X_3_0_1 );
@BIN ( X_3_0_2 );
@BIN ( X_3_1_1 );
@BIN ( X_3_1_2 );
@BIN ( X_3_2_1 );
@BIN ( X_3_2_2 );
@BIN ( X_3_4_1 );
@BIN ( X_3_4_2 );
@BIN ( X_3_5_1 );
@BIN ( X_3_5_2 );
@BIN ( X_3_6_1 );
@BIN ( X_3_6_2 );
@BIN ( X_4_0_1 );
@BIN ( X_4_0_2 );
@BIN ( X_4_1_1 );
@BIN ( X_4_1_2 );
@BIN ( X_4_2_1 );
@BIN ( X_4_2_2 );
@BIN ( X_4_3_1 );
@BIN ( X_4_3_2 );
@BIN ( X_4_5_1 );
@BIN ( X_4_5_2 );
@BIN ( X_4_6_1 );
@BIN ( X_4_6_2 );
@BIN ( X_5_0_1 );
@BIN ( X_5_0_2 );
@BIN ( X_5_1_1 );
@BIN ( X_5_1_2 );
@BIN ( X_5_2_1 );
@BIN ( X_5_2_2 );
@BIN ( X_5_3_1 );
@BIN ( X_5_3_2 );
@BIN ( X_5_4_1 );
@BIN ( X_5_4_2 );
@BIN ( X_5_6_1 );
@BIN ( X_5_6_2 );
@BIN ( X_6_0_1 );
@BIN ( X_6_0_2 );
@BIN ( X_6_1_1 );
@BIN ( X_6_1_2 );
@BIN ( X_6_2_1 );
@BIN ( X_6_2_2 );
@BIN ( X_6_3_1 );
```

```
@BIN ( X_6_3_2 );
@BIN ( X_6_4_1 );
@BIN ( X_6_4_2 );
@BIN ( X_6_5_1 );
@BIN ( X_6_5_2 );

@BIN ( Y_0_1_1 );
@BIN ( Y_0_2_1 );
@BIN ( Y_0_3_1 );
@BIN ( Y_1_0_1 );
@BIN ( Y_1_2_1 );
@BIN ( Y_1_3_1 );
@BIN ( Y_2_0_1 );
@BIN ( Y_2_1_1 );
@BIN ( Y_2_3_1 );
@BIN ( Y_3_0_1 );
@BIN ( Y_3_1_1 );
@BIN ( Y_3_2_1 );

@GIN ( w_1 );
@GIN ( w_2 );
@GIN ( w_3 );

@GIN ( U_1_1 );
@GIN ( U_1_2 );
@GIN ( U_2_1 );
@GIN ( U_2_2 );
@GIN ( U_3_1 );
@GIN ( U_3_2 );
@GIN ( U_4_1 );
@GIN ( U_4_2 );
@GIN ( U_5_1 );
@GIN ( U_5_2 );
@GIN ( U_6_1 );
@GIN ( U_6_2 );
@GIN ( V_1_1 );
@GIN ( V_2_1 );
@GIN ( V_3_1 );

End
```

# Appendix B: Taguchi method for the hierarchical approach using genetic algorithm

Table B.1 The results of experiment run no. 1 in the Taguchi method.

| | V-d100-i10-a | | V-d100-i10-b | | V-d100-i10-c | | V-d100-i10-d | | V-d100-i10-e | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Test no.** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** |
| 1 | | 689.50 | | 746.65 | | 734.93 | | 733.74 | | 656.88 |
| 2 | | 689.50 | | 746.65 | | 734.93 | | 733.74 | | 656.88 |
| 3 | 4086.90 | 689.50 | 4150.07 | 746.65 | 4191.73 | 734.93 | 4078.15 | 733.74 | 3980.64 | 656.88 |
| 4 | | 689.50 | | 746.65 | | 734.93 | | 733.74 | | 656.88 |
| 5 | | 689.50 | | 746.65 | | 734.93 | | 733.74 | | 656.88 |
| 6 | | 733.31 | | 847.34 | | 873.28 | | 763.78 | | 694.69 |
| 7 | | 733.31 | | 847.34 | | 873.28 | | 763.78 | | 694.69 |
| 8 | 4115.12 | 733.31 | 4167.86 | 847.34 | 4217.86 | 873.28 | 4043.51 | 763.78 | 4085.41 | 694.69 |
| 9 | | 733.31 | | 847.34 | | 873.28 | | 763.78 | | 694.69 |
| 10 | | 733.31 | | 847.34 | | 873.28 | | 763.78 | | 694.69 |
| 11 | | 711.43 | | 751.99 | | 892.56 | | 719.96 | | 774.31 |
| 12 | | 711.43 | | 751.99 | | 892.56 | | 719.96 | | 774.31 |
| 13 | 3985.26 | 711.43 | 4206.11 | 751.99 | 4255.20 | 892.56 | 4192.38 | 719.96 | 4020.37 | 774.31 |
| 14 | | 711.43 | | 751.99 | | 892.56 | | 719.96 | | 774.31 |
| 15 | | 711.43 | | 751.99 | | 892.56 | | 719.96 | | 774.31 |
| 16 | | 798.64 | | 787.11 | | 753.66 | | 577.88 | | 781.51 |
| 17 | | 793.89 | | 787.11 | | 753.66 | | 577.88 | | 781.51 |
| 18 | 4055.78 | 793.89 | 4194.87 | 787.11 | 4217.80 | 753.66 | 4214.14 | 577.88 | 4067.32 | 781.51 |
| 19 | | 793.89 | | 787.11 | | 753.66 | | 577.88 | | 781.51 |
| 20 | | 793.89 | | 787.11 | | 753.66 | | 577.88 | | 781.51 |
| 21 | | 721.54 | | 772.85 | | 793.68 | | 701.21 | | 826.31 |
| 22 | | 721.54 | | 772.85 | | 793.68 | | 701.21 | | 826.31 |
| 23 | 4014.40 | 721.54 | 4208.71 | 772.85 | 4119.93 | 793.68 | 4078.83 | 701.21 | 4037.72 | 826.31 |
| 24 | | 721.54 | | 772.85 | | 793.68 | | 701.21 | | 826.31 |
| 25 | | 721.54 | | 772.85 | | 793.68 | | 701.21 | | 826.31 |
| **Average** | **4051.492** | **730.124** | **4185.524** | **781.188** | **4200.504** | **809.622** | **4121.402** | **699.314** | **4038.292** | **746.740** |
| **Stdev** | **52.638** | **36.158** | **25.582** | **36.873** | **50.401** | **64.381** | **76.467** | **65.379** | **40.951** | **63.055** |

(Number of installation customers (A) = 10, Service Level (B) =60 min, Installation time (C) = 10 min, Fixed cost per installation vehicle (D) = 50)

Table B.2 The results of experiment run no. 2 in the Taguchi method.

| | V-d100-i10-a | | V-d100-i10-b | | V-d100-i10-c | | V-d100-i10-d | | V-d100-i10-e | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Test no.** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** |
| 1 | | 1048.77 | | 1118.65 | | 1120.28 | | 1115.26 | | 1064.94 |
| 2 | | 1048.01 | | 1118.65 | | 1120.28 | | 1115.26 | | 1064.94 |
| 3 | 4086.90 | 1048.01 | 4150.07 | 1118.65 | 4191.73 | 1147.22 | 4078.15 | 1115.26 | 3980.64 | 1064.94 |
| 4 | | 1048.01 | | 1118.65 | | 1120.28 | | 1115.26 | | 1064.94 |
| 5 | | 1048.01 | | 1118.65 | | 1120.28 | | 1115.26 | | 1064.94 |
| 6 | | 1093.18 | | 1254.2 | | 1128.91 | | 1131.6 | | 1091.81 |
| 7 | | 1093.18 | | 1176.43 | | 1128.91 | | 1131.6 | | 1091.81 |
| 8 | 4115.12 | 1093.18 | 4167.86 | 1176.43 | 4217.86 | 1128.91 | 4043.51 | 1131.6 | 4085.41 | 1091.81 |
| 9 | | 1093.18 | | 1176.43 | | 1128.91 | | 1131.6 | | 1091.81 |
| 10 | | 1094.04 | | 1254.2 | | 1128.91 | | 1131.6 | | 1091.81 |
| 11 | | 1072.31 | | 1139.78 | | 1139.39 | | 1090.99 | | 1091.07 |
| 12 | | 1074.91 | | 1139.78 | | 1139.39 | | 1090.99 | | 1091.07 |
| 13 | 3985.26 | 1072.31 | 4206.11 | 1139.78 | 4255.20 | 1139.39 | 4192.38 | 1090.99 | 4020.37 | 1091.07 |
| 14 | | 1072.31 | | 1152.34 | | 1139.39 | | 1090.99 | | 1091.07 |
| 15 | | 1072.31 | | 1139.78 | | 1139.39 | | 1090.99 | | 1091.07 |
| 16 | | 1136.36 | | 1343.15 | | 1127.63 | | 927.44 | | 1104.42 |
| 17 | | 1136.36 | | 1343.15 | | 1127.63 | | 927.44 | | 1109.54 |
| 18 | 4055.78 | 1136.36 | 4194.87 | 1343.15 | 4217.80 | 1127.63 | 4214.14 | 927.44 | 4067.32 | 1104.42 |
| 19 | | 1136.36 | | 1343.15 | | 1127.63 | | 927.44 | | 1104.42 |
| 20 | | 1136.36 | | 1343.15 | | 1127.63 | | 927.44 | | 1104.42 |
| 21 | | 1082.75 | | 1157.37 | | 1170.05 | | 1079.76 | | 1155.06 |
| 22 | | 1082.75 | | 1148.16 | | 1170.05 | | 1079.76 | | 1155.06 |
| 23 | 4014.40 | 1082.75 | 4208.71 | 1148.16 | 4119.93 | 1170.05 | 4078.83 | 1079.76 | 4037.72 | 1155.06 |
| 24 | | 1082.75 | | 1148.16 | | 1170.05 | | 1079.76 | | 1155.06 |
| 25 | | 1082.75 | | 1148.16 | | 1170.05 | | 1079.76 | | 1155.06 |
| **Average** | **4051.492** | **1086.691** | **4185.524** | **1192.326** | **4200.504** | **1138.330** | **4121.402** | **1069.010** | **4038.292** | **1101.665** |
| **Stdev** | **52.638** | **29.597** | **25.582** | **84.400** | **50.401** | **17.601** | **76.467** | **74.578** | **40.951** | **30.371** |

(Number of installation customers (A) = 10, Service Level (B) =120 min, Installation time (C) = 35 min, Fixed cost per installation vehicle (D) = 100)

Table B.3 The results of experiment run no. 3 in the Taguchi method.

| | V-d100-i10-a | | V-d100-i10-b | | V-d100-i10-c | | V-d100-i10-d | | V-d100-i10-e | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Test no.** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** |
| 1 | | 1435.74 | | 1518.65 | | 1524.98 | | 1509.15 | | 1475.31 |
| 2 | | 1435.74 | | 1518.65 | | 1524.98 | | 1509.15 | | 1475.31 |
| 3 | 4086.90 | 1435.74 | 4150.07 | 1518.65 | 4191.73 | 1524.98 | 4078.15 | 1509.15 | 3980.64 | 1475.31 |
| 4 | | 1435.74 | | 1518.65 | | 1524.98 | | 1509.15 | | 1475.31 |
| 5 | | 1435.74 | | 1518.65 | | 1524.98 | | 1509.15 | | 1483.82 |
| 6 | | 1468.18 | | 1533.26 | | 1528.91 | | 1530.89 | | 1511.42 |
| 7 | | 1468.18 | | 1533.26 | | 1528.91 | | 1530.89 | | 1511.42 |
| 8 | 4115.12 | 1468.18 | 4167.86 | 1533.26 | 4217.86 | 1528.91 | 4043.51 | 1530.89 | 4085.41 | 1511.42 |
| 9 | | 1468.18 | | 1533.26 | | 1528.91 | | 1530.89 | | 1511.42 |
| 10 | | 1468.18 | | 1533.26 | | 1528.91 | | 1530.89 | | 1511.42 |
| 11 | | 1449.91 | | 1539.78 | | 1539.39 | | 1527.18 | | 1472.67 |
| 12 | | 1449.91 | | 1539.78 | | 1539.39 | | 1527.18 | | 1472.67 |
| 13 | 3985.26 | 1449.91 | 4206.11 | 1539.78 | 4255.20 | 1539.39 | 4192.38 | 1527.18 | 4020.37 | 1472.67 |
| 14 | | 1449.91 | | 1539.78 | | 1539.39 | | 1527.18 | | 1472.67 |
| 15 | | 1449.91 | | 1539.78 | | 1539.39 | | 1527.18 | | 1472.67 |
| 16 | | 1505.7 | | 1769.86 | | 1527.63 | | 1532.69 | | 1469.14 |
| 17 | | 1505.7 | | 1769.86 | | 1527.63 | | 1532.69 | | 1469.14 |
| 18 | 4055.78 | 1505.7 | 4194.87 | 1769.86 | 4217.80 | 1527.63 | 4214.14 | 1532.69 | 4067.32 | 1469.14 |
| 19 | | 1505.7 | | 1769.86 | | 1527.63 | | 1532.69 | | 1469.14 |
| 20 | | 1505.7 | | 1769.86 | | 1527.63 | | 1532.69 | | 1469.14 |
| 21 | | 1482.75 | | 1548.16 | | 1575.23 | | 1499.83 | | 1551.59 |
| 22 | | 1482.75 | | 1557.37 | | 1575.23 | | 1499.83 | | 1538.07 |
| 23 | 4014.40 | 1482.75 | 4208.71 | 1548.16 | 4119.93 | 1575.23 | 4078.83 | 1499.83 | 4037.72 | 1551.59 |
| 24 | | 1482.75 | | 1548.16 | | 1575.23 | | 1499.83 | | 1538.07 |
| 25 | | 1482.75 | | 1548.16 | | 1575.23 | | 1499.83 | | 1538.07 |
| **Average** | **4051.492** | **1468.456** | **4185.524** | **1582.310** | **4200.504** | **1539.228** | **4121.402** | **1519.948** | **4038.292** | **1494.744** |
| **Stdev** | **52.638** | **25.024** | **25.582** | **96.285** | **50.401** | **19.040** | **76.467** | **13.352** | **40.951** | **29.460** |

(Number of installation customers (A) = 10, Service Level (B) =180 min, Installation time (C) = 60 min, Fixed cost per installation vehicle (D) = 150)

Table B.4 The results of experiment run no. 4 in the Taguchi method.

| | V-d100-i30-a | | V-d100-i30-b | | V-d100-i30-c | | V-d100-i30-d | | V-d100-i30-e | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Test no.** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** |
| 1 | | 3400.61 | | 3422.95 | | 3480.10 | | 3530.77 | | 3211.63 |
| 2 | | 3370.10 | | 3464.49 | | 3719.63 | | 3511.09 | | 3224.82 |
| 3 | 4145.00 | 3380.45 | 4325.80 | 3671.18 | 4042.11 | 3714.71 | 4207.26 | 3511.09 | 3873.12 | 3236.17 |
| 4 | | 3173.01 | | 3422.95 | | 3749.83 | | 3379.02 | | 3229.97 |
| 5 | | 3344.49 | | 3439.89 | | 3486.72 | | 3387.13 | | 3211.63 |
| 6 | | 3120.51 | | 3739.45 | | 3444.82 | | 3462.45 | | 3207.14 |
| 7 | | 3124.36 | | 3907.84 | | 3383.54 | | 3512.94 | | 3386.84 |
| 8 | 4076.46 | 3118.45 | 4242.68 | 3727.49 | 4018.72 | 3429.04 | 4121.96 | 3461.72 | 4039.66 | 3438.26 |
| 9 | | 3087.99 | | 3740.04 | | 3194.48 | | 3462.45 | | 3421.39 |
| 10 | | 3093.10 | | 3727.49 | | 3475.78 | | 3469.93 | | 3395.67 |
| 11 | | 3114.58 | | 3466.84 | | 3480.92 | | 3423.51 | | 3564.05 |
| 12 | | 3105.76 | | 3472.01 | | 3485.00 | | 3296.40 | | 3357.46 |
| 13 | 4018.01 | 3098.90 | 4258.49 | 3663.70 | 4100.15 | 3491.10 | 4099.00 | 3504.81 | 4070.10 | 3563.50 |
| 14 | | 3091.16 | | 3729.94 | | 3482.60 | | 3322.04 | | 3557.42 |
| 15 | | 3099.91 | | 3649.52 | | 3545.88 | | 3488.07 | | 3358.14 |
| 16 | | 3362.61 | | 3800.37 | | 3501.90 | | 3271.55 | | 3230.48 |
| 17 | | 3122.86 | | 3536.69 | | 3549.09 | | 3271.55 | | 3254.70 |
| 18 | 4004.77 | 3338.78 | 4284.57 | 3536.69 | 4106.16 | 3280.73 | 4149.02 | 3527.19 | 3967.35 | 3235.93 |
| 19 | | 3099.56 | | 3546.67 | | 3510.07 | | 3271.55 | | 3230.48 |
| 20 | | 3311.44 | | 3628.63 | | 3283.45 | | 3271.55 | | 3254.70 |
| 21 | | 3064.63 | | 3561.08 | | 3491.25 | | 3374.94 | | 3194.55 |
| 22 | | 3087.46 | | 3561.08 | | 3465.51 | | 3620.12 | | 3189.42 |
| 23 | 4056.58 | 3096.04 | 4267.90 | 3789.19 | 4065.32 | 3675.64 | 4065.18 | 3374.43 | 3985.68 | 3216.71 |
| 24 | | 3143.21 | | 3560.43 | | 3504.18 | | 3629.53 | | 3204.03 |
| 25 | | 3051.73 | | 3561.08 | | 3493.89 | | 3374.94 | | 3011.45 |
| **Average** | **4060.164** | **3176.068** | **4275.888** | **3613.107** | **4066.491** | **3492.794** | **4128.484** | **3428.430** | **3987.182** | **3295.461** |
| **Stdev** | **55.500** | **119.220** | **31.764** | **133.078** | **37.364** | **129.797** | **53.728** | **106.607** | **75.886** | **135.652** |

(Number of installation customers (A) = 30, Service Level (B) =60 min, Installation time (C) = 35 min, Fixed cost per installation vehicle (D) = 150)

Table B.5 The results of experiment run no. 5 in the Taguchi method.

| | V-d100-i30-a | | V-d100-i30-b | | V-d100-i30-c | | V-d100-i30-d | | V-d100-i30-e | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Test no.** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** |
| 1 | | 3112.68 | | 3432.01 | | 3486.28 | | 3380.07 | | 3277.95 |
| 2 | | 3146.69 | | 3358.61 | | 3470.38 | | 3399.76 | | 3264.18 |
| 3 | 4145.00 | 3137.82 | 4325.80 | 3429.80 | 4042.11 | 3460.72 | 4207.26 | 3333.32 | 3873.12 | 3264.18 |
| 4 | | 3143.61 | | 3442.72 | | 3459.20 | | 3294.20 | | 3264.18 |
| 5 | | 3135.26 | | 3459.78 | | 3467.43 | | 3354.04 | | 3264.18 |
| 6 | | 3022.46 | | 3415.16 | | 3395.79 | | 3307.88 | | 3233.35 |
| 7 | | 3026.36 | | 3411.20 | | 3412.17 | | 3348.18 | | 3247.51 |
| 8 | 4076.46 | 3097.94 | 4242.68 | 3443.63 | 4018.72 | 3271.67 | 4121.96 | 3317.45 | 4039.66 | 3249.16 |
| 9 | | 3019.46 | | 3473.36 | | 3403.81 | | 3310.59 | | 3242.80 |
| 10 | | 3066.21 | | 3447.91 | | 3247.46 | | 3332.54 | | 3237.46 |
| 11 | | 3123.00 | | 3372.71 | | 3434.00 | | 3292.70 | | 3376.61 |
| 12 | | 3150.94 | | 3414.98 | | 3445.59 | | 3289.57 | | 3393.19 |
| 13 | 4018.01 | 3078.27 | 4258.49 | 3425.51 | 4100.15 | 3356.90 | 4099.00 | 3290.47 | 4070.10 | 3423.38 |
| 14 | | 3069.92 | | 3405.49 | | 3335.33 | | 3303.97 | | 3368.43 |
| 15 | | 3147.34 | | 3407.73 | | 3445.10 | | 3329.91 | | 3385.01 |
| 16 | | 3161.24 | | 3350.85 | | 3347.93 | | 3324.66 | | 3175.87 |
| 17 | | 3148.79 | | 3491.17 | | 3336.98 | | 3470.12 | | 3294.38 |
| 18 | 4004.77 | 3188.00 | 4284.57 | 3468.48 | 4106.16 | 3367.16 | 4149.02 | 3361.50 | 3967.35 | 3299.27 |
| 19 | | 3161.27 | | 3466.33 | | 3360.77 | | 3335.32 | | 3269.34 |
| 20 | | 3158.10 | | 3534.03 | | 3412.11 | | 3322.78 | | 3190.52 |
| 21 | | 3169.00 | | 3511.28 | | 3387.62 | | 3478.01 | | 3272.26 |
| 22 | | 3110.39 | | 3408.80 | | 3453.32 | | 3469.05 | | 3298.72 |
| 23 | 4056.58 | 3120.51 | 4267.90 | 3536.00 | 4065.32 | 3327.63 | 4065.18 | 3486.05 | 3985.68 | 3273.17 |
| 24 | | 3155.86 | | 3559.30 | | 3310.28 | | 3477.90 | | 3285.73 |
| 25 | | 3115.09 | | 3544.16 | | 3321.09 | | 3474.59 | | 3214.73 |
| **Average** | **4060.164** | **3118.648** | **4275.888** | **3448.439** | **4066.491** | **3388.668** | **4128.484** | **3363.384** | **3987.182** | **3282.621** |
| **Stdev** | **55.500** | **47.291** | **31.764** | **56.607** | **37.364** | **65.862** | **53.728** | **69.856** | **75.886** | **62.600** |

(Number of installation customers (A) = 30, Service Level (B) =120 min, Installation time (C) = 60 min, Fixed cost per installation vehicle (D) = 50)

Table B.6 The results of experiment run no. 6 in the Taguchi method.

| | V-d100-i30-a | | V-d100-i30-b | | V-d100-i30-c | | V-d100-i30-d | | V-d100-i30-e | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Test no.** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** |
| 1 | | 1198.74 | | 1287.67 | | 1202.16 | | 1419.63 | | 1189.93 |
| 2 | | 1195.71 | | 1184.86 | | 1162.43 | | 1462.39 | | 1247.69 |
| 3 | 4145.00 | 1228.11 | 4325.80 | 1216.15 | 4042.11 | 1165.59 | 4207.26 | 1285.75 | 3873.12 | 1251.96 |
| 4 | | 1194.26 | | 1266.28 | | 1162.83 | | 1251.94 | | 1157.50 |
| 5 | | 1181.57 | | 1236.57 | | 1229.27 | | 1248.54 | | 1170.68 |
| 6 | | 1220.22 | | 1276.17 | | 1354.21 | | 1401.83 | | 1150.72 |
| 7 | | 1207.69 | | 1263.32 | | 1299.51 | | 1420.68 | | 1197.53 |
| 8 | 4076.46 | 1164.25 | 4242.68 | 1283.01 | 4018.72 | 1192.35 | 4121.96 | 1276.82 | 4039.66 | 1156.70 |
| 9 | | 1254.70 | | 1256.30 | | 1231.91 | | 1428.92 | | 1166.42 |
| 10 | | 1193.14 | | 1388.79 | | 1180.47 | | 1434.71 | | 1165.86 |
| 11 | | 1151.88 | | 1231.95 | | 1181.46 | | 1463.15 | | 1286.07 |
| 12 | | 1143.59 | | 1351.42 | | 1199.76 | | 1309.25 | | 1289.80 |
| 13 | 4018.01 | 1166.82 | 4258.49 | 1190.92 | 4100.15 | 1202.81 | 4099.00 | 1253.47 | 4070.10 | 1252.34 |
| 14 | | 1165.85 | | 1193.95 | | 1424.14 | | 1261.93 | | 1256.54 |
| 15 | | 1172.58 | | 1361.46 | | 1197.65 | | 1219.14 | | 1225.26 |
| 16 | | 1154.30 | | 1363.12 | | 1178.14 | | 1360.81 | | 1250.98 |
| 17 | | 1205.39 | | 1269.22 | | 1202.60 | | 1428.65 | | 1169.85 |
| 18 | 4004.77 | 1131.96 | 4284.57 | 1363.12 | 4106.16 | 1180.42 | 4149.02 | 1423.26 | 3967.35 | 1177.65 |
| 19 | | 1208.94 | | 1258.90 | | 1180.42 | | 1447.02 | | 1177.74 |
| 20 | | 1190.32 | | 1270.70 | | 1250.00 | | 1320.98 | | 1183.97 |
| 21 | | 1161.39 | | 1298.77 | | 1365.85 | | 1503.11 | | 1147.07 |
| 22 | | 1170.05 | | 1210.59 | | 1191.14 | | 1492.44 | | 1182.13 |
| 23 | 4056.58 | 1164.16 | 4267.90 | 1230.14 | 4065.32 | 1353.98 | 4065.18 | 1498.61 | 3985.68 | 1197.54 |
| 24 | | 1237.14 | | 1230.21 | | 1197.17 | | 1463.88 | | 1236.51 |
| 25 | | 1207.88 | | 1237.75 | | 1344.79 | | 1268.76 | | 1178.14 |
| **Average** | **4060.164** | **1186.826** | **4275.888** | **1268.854** | **4066.491** | **1233.242** | **4128.484** | **1373.827** | **3987.182** | **1202.663** |
| **Stdev** | **55.500** | **30.400** | **31.764** | **57.960** | **37.364** | **76.094** | **53.728** | **93.483** | **75.886** | **43.737** |

(Number of installation customers (A) = 30, Service Level (B) =180 min, Installation time (C) = 10 min, Fixed cost per installation vehicle (D) = 100)

Table B.7 The results of experiment run no. 7 in the Taguchi method.

| | V-d100-i50-a | | V-d100-i50-b | | V-d100-i50-c | | V-d100-i50-d | | V-d100-i50-e | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Test no.** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** |
| 1 | | 7304.427 | | 6817.388 | | 7284.807 | | 6969.718 | | 6956.068 |
| 2 | | 7591.708 | | 6969.478 | | 7481.187 | | 6948.618 | | 6943.587 |
| 3 | 4140.03 | 7334.917 | 4235.63 | 6819.688 | 4056.469 | 7446.787 | 3896.27 | 6845.928 | 3827.25 | 7053.427 |
| 4 | | 7505.577 | | 6991.667 | | 7452.618 | | 7029.297 | | 6954.758 |
| 5 | | 7496.757 | | 7002.157 | | 7259.737 | | 7013.528 | | 6979.908 |
| 6 | | 7355.618 | | 7184.917 | | 6869.947 | | 6922.817 | | 6833.198 |
| 7 | | 7509.447 | | 7282.518 | | 6846.057 | | 6881.768 | | 6784.127 |
| 8 | 4029.14 | 7307.797 | 4280.86 | 7117.018 | 4196.24 | 7053.458 | 3891.96 | 6978.377 | 3909.96 | 6805.868 |
| 9 | | 7323.057 | | 7031.908 | | 6854.447 | | 6909.388 | | 6797.517 |
| 10 | | 7308.488 | | 7189.107 | | 6971.337 | | 6910.277 | | 6956.298 |
| 11 | | 7313.227 | | 7126.768 | | 6897.567 | | 6695.857 | | 7054.478 |
| 12 | | 7276.098 | | 7095.687 | | 7161.817 | | 6907.388 | | 7059.317 |
| 13 | 3971.28 | 7270.797 | 4242.44 | 7140.397 | 4159.13 | 7054.618 | 3861.45 | 6887.488 | 3911.76 | 7254.888 |
| 14 | | 7272.428 | | 7180.447 | | 7111.397 | | 6954.868 | | 6831.687 |
| 15 | | 7288.157 | | 6930.777 | | 6933.068 | | 6907.027 | | 7039.638 |
| 16 | | 7206.317 | | 7360.367 | | 7287.238 | | 7038.397 | | 7066.118 |
| 17 | | 7215.197 | | 7384.808 | | 7238.697 | | 6825.887 | | 6892.257 |
| 18 | 4030.77 | 7218.957 | 4233.169 | 7336.627 | 4095.81 | 7232.328 | 3902 | 6846.147 | 3883.7 | 6827.137 |
| 19 | | 7433.157 | | 7183.267 | | 7219.388 | | 6880.807 | | 6828.657 |
| 20 | | 7370.347 | | 7292.587 | | 7206.637 | | 6713.288 | | 7048.698 |
| 21 | | 7245.857 | | 7208.098 | | 7291.768 | | 7149.847 | | 6634.228 |
| 22 | | 7457.316 | | 7173.007 | | 7320.297 | | 7133.017 | | 6628.108 |
| 23 | 3976.87 | 7237.728 | 4327.42 | 7173.897 | 4101.58 | 7116.048 | 3838.47 | 7122.107 | 3976.49 | 6657.528 |
| 24 | | 7433.147 | | 7135.027 | | 7125.067 | | 7182.627 | | 6611.968 |
| 25 | | 7204.647 | | 7135.027 | | 7286.627 | | 7094.458 | | 6660.998 |
| **Average** | **4029.618** | **7339.247** | **4263.904** | **7130.505** | **4121.846** | **7160.118** | **3878.030** | **6949.957** | **3901.832** | **6886.418** |
| **Stdev** | **67.783** | **108.497** | **40.394** | **149.756** | **55.424** | **188.078** | **27.116** | **125.169** | **53.921** | **167.921** |

(Number of installation customers (A) = 50, Service Level (B) =60 min, Installation time (C) = 60 min, Fixed cost per installation vehicle (D) = 100)

Table B.8 The results of experiment run no. 8 in the Taguchi method.

| | V-d100-i50-a | | V-d100-i50-b | | V-d100-i50-c | | V-d100-i50-d | | V-d100-i50-e | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Test no.** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** |
| 1 | | 2500.41 | | 2394.97 | | 2434.02 | | 2110.41 | | 2635.94 |
| 2 | | 2490.12 | | 2681.43 | | 2402.62 | | 2340.96 | | 2550.88 |
| 3 | 4140.03 | 2502.46 | 4235.63 | 2403.31 | 4056.47 | 2393.19 | 3896.27 | 2352.88 | 3827.25 | 2330.23 |
| 4 | | 2460.45 | | 2449.37 | | 2424.41 | | 2132.98 | | 2364.87 |
| 5 | | 2474.58 | | 2367.51 | | 2425.28 | | 2363.57 | | 2388.89 |
| 6 | | 2365.71 | | 2614.37 | | 2423.67 | | 2314.53 | | 2372.77 |
| 7 | | 2334.70 | | 2427.48 | | 2413.73 | | 2367.67 | | 2367.47 |
| 8 | 4029.14 | 2439.59 | 4280.86 | 2418.63 | 4196.24 | 2411.05 | 3891.96 | 2316.38 | 3909.96 | 2676.02 |
| 9 | | 2366.92 | | 2374.72 | | 2421.80 | | 2331.55 | | 2213.97 |
| 10 | | 2385.87 | | 2458.18 | | 2386.02 | | 2404.27 | | 2358.89 |
| 11 | | 2377.08 | | 2653.34 | | 2389.65 | | 2170.48 | | 2396.99 |
| 12 | | 2390.90 | | 2432.64 | | 2393.90 | | 2389.80 | | 2381.31 |
| 13 | 3971.28 | 2428.74 | 4242.44 | 2368.03 | 4159.13 | 2467.88 | 3861.45 | 2164.57 | 3911.76 | 2421.21 |
| 14 | | 2169.94 | | 2448.09 | | 2450.40 | | 2153.62 | | 2408.72 |
| 15 | | 2385.37 | | 2402.40 | | 2432.95 | | 2346.63 | | 2435.37 |
| 16 | | 2404.06 | | 2386.30 | | 2370.59 | | 2416.36 | | 2395.43 |
| 17 | | 2170.04 | | 2633.97 | | 2351.35 | | 2417.18 | | 2395.06 |
| 18 | 4030.77 | 2443.39 | 4233.17 | 2454.31 | 4095.81 | 2402.08 | 3902.00 | 2385.00 | 3883.70 | 2406.76 |
| 19 | | 2411.99 | | 2336.77 | | 2383.39 | | 2378.81 | | 2412.79 |
| 20 | | 2444.06 | | 2446.38 | | 2381.03 | | 2433.83 | | 2656.30 |
| 21 | | 2748.08 | | 2717.18 | | 2350.07 | | 2437.13 | | 2196.79 |
| 22 | | 2708.04 | | 2471.03 | | 2572.68 | | 2364.85 | | 2485.36 |
| 23 | 3976.87 | 2671.23 | 4327.42 | 2473.48 | 4101.58 | 2330.45 | 3838.47 | 2165.13 | 3976.49 | 2169.16 |
| 24 | | 2552.34 | | 2688.37 | | 2615.68 | | 2446.15 | | 2134.02 |
| 25 | | 2437.38 | | 2700.86 | | 2387.41 | | 2647.27 | | 2217.36 |
| **Average** | **4029.618** | **2442.538** | **4263.904** | **2488.125** | **4121.846** | **2416.612** | **3878.030** | **2334.080** | **3901.832** | **2390.902** |
| **Stdev** | **67.783** | **133.330** | **40.394** | **121.963** | **55.424** | **62.304** | **27.116** | **123.962** | **53.921** | **140.078** |

(Number of installation customers (A) = 50, Service Level (B) =120 min, Installation time (C) = 10 min, Fixed cost per installation vehicle (D) = 150)

Table B.9 The results of experiment run no. 9 in the Taguchi method.

| | V-d100-i50-a | | V-d100-i50-b | | V-d100-i50-c | | V-d100-i50-d | | V-d100-i50-e | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Test no.** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** | **Stage 1** | **Stage 2** |
| 1 | | 3480.01 | | 3270.319 | | 3411.619 | | 3311.309 | | 3367.539 |
| 2 | | 3336.24 | | 3350.389 | | 3256.629 | | 3280.639 | | 3329.049 |
| 3 | 4140.03 | 3408.08 | 4235.63 | 3327.329 | 4056.469 | 3213.999 | 3896.27 | 3238.529 | 3827.25 | 3347.879 |
| 4 | | 3508.929 | | 3290.479 | | 3361.49 | | 3382.639 | | 3300.289 |
| 5 | | 3450.599 | | 3256.019 | | 3361.229 | | 3344.579 | | 3346.909 |
| 6 | | 3457.879 | | 3355.989 | | 3395.179 | | 3197.589 | | 3234.679 |
| 7 | | 3341.879 | | 3291.979 | | 3253.03 | | 3331.279 | | 3219.049 |
| 8 | 4029.14 | 3431.589 | 4280.86 | 3378.219 | 4196.24 | 3217.669 | 3891.96 | 3295.269 | 3909.96 | 3407.169 |
| 9 | | 3353.329 | | 3351.239 | | 3248.639 | | 3296.889 | | 3226.679 |
| 10 | | 3226.609 | | 3330.309 | | 3250.669 | | 3291.239 | | 3226.309 |
| 11 | | 3365.709 | | 3347.28 | | 3170.739 | | 3203.619 | | 3273.429 |
| 12 | | 3357.299 | | 3274.739 | | 3215.789 | | 3210.339 | | 3236.789 |
| 13 | 3971.28 | 3373.82 | 4242.44 | 3245.199 | 4159.13 | 3262.429 | 3861.45 | 3279.999 | 3911.76 | 3231.959 |
| 14 | | 3272.129 | | 3201.329 | | 3228.189 | | 3266.469 | | 3300.979 |
| 15 | | 3335.399 | | 3357.519 | | 3192.869 | | 3286.659 | | 3234.529 |
| 16 | | 3315.589 | | 3406.699 | | 3325.539 | | 3330.789 | | 3313.939 |
| 17 | | 3287.759 | | 3303.599 | | 3260.919 | | 3229.979 | | 3394.489 |
| 18 | 4030.77 | 3296.949 | 4233.169 | 3296.529 | 4095.81 | 3297.469 | 3902 | 3385.349 | 3883.7 | 3269.229 |
| 19 | | 3359.59 | | 3411.869 | | 3295.909 | | 3311.729 | | 3447.229 |
| 20 | | 3283.959 | | 3356.99 | | 3339.839 | | 3192.549 | | 3208.139 |
| 21 | | 3416.729 | | 3416.24 | | 3289.019 | | 3273.979 | | 3414.569 |
| 22 | | 3363.229 | | 3324.529 | | 3244.479 | | 3362.289 | | 3378.719 |
| 23 | 3976.87 | 3322.879 | 4327.42 | 3348.359 | 4101.58 | 3183.309 | 3838.47 | 3219.759 | 3976.49 | 3301.649 |
| 24 | | 3305.399 | | 3453.899 | | 3239.01 | | 3509.069 | | 3239.619 |
| 25 | | 3331.909 | | 3333.709 | | 3268.499 | | 3413.199 | | 3218.459 |
| **Average** | **4029.618** | **3359.340** | **4263.904** | **3331.230** | **4121.846** | **3271.366** | **3878.030** | **3297.829** | **3901.832** | **3298.771** |
| **Stdev** | **67.783** | **69.044** | **40.394** | **58.465** | **55.424** | **64.243** | **27.116** | **75.256** | **53.921** | **72.570** |

(Number of installation customers (A) = 50, Service Level (B) =180 min, Installation time (C) = 35 min, Fixed cost per installation vehicle (D) = 50)

# Appendix C: Program code of the Endosymbiotic evolutionary algorithm

The program of the EEA for the VRP under consideration has been programmed in Visual Basic language with the Microsoft Visual Studio.NET Framework 1.1 version. The functions for the EEA have been programmed as modules. The program consists of 11 files whose program codes are as follows.

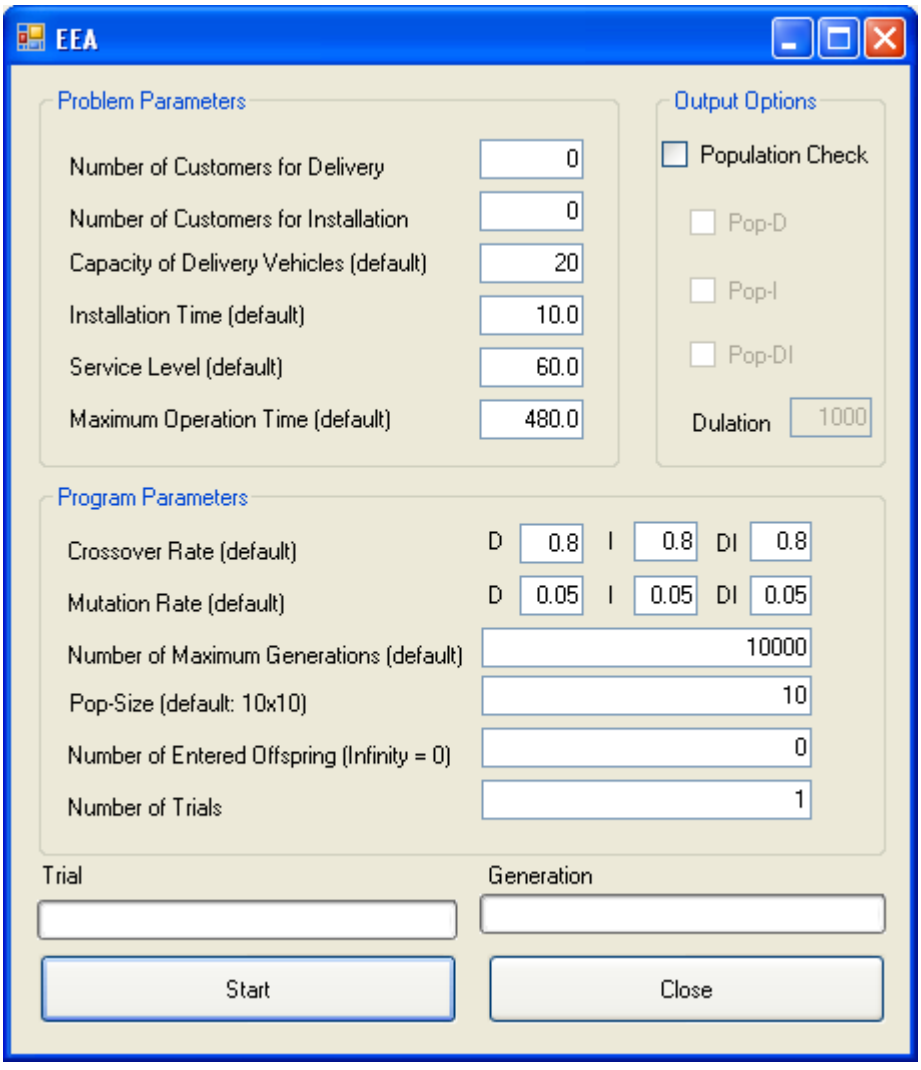



Figure C.1 The program interface for the EEA (EEA.vb).

Table C.1 The program code in EEA.vb.

```
Public Class EEA

    'Initialize interface
    Private Sub EEA_Load(ByVal sender As System.Object, ByVal e As System.Eve
ntArgs) Handles MyBase.Load
        Randomize(CDec(Now.Millisecond))
        chk_POPI.Enabled = False
        chk_POPD.Enabled = False
        chk_POPDI.Enabled = False
        txt_POP_DU.Enabled = False
    End Sub


    'Program start
    Private Sub Btn_Start_EEA_Click(ByVal sender As System.Object, ByVal e As
 System.EventArgs) Handles Btn_Start_EEA.Click

        btn_close.Enabled = False
        Btn_Start_EEA.Enabled = False

        txt_NCD.Enabled = False
        txt_NCI.Enabled = False
        txt_XR_D.Enabled = False
        txt_XR_I.Enabled = False
        txt_XR_DI.Enabled = False
        txt_MR_D.Enabled = False
        txt_MR_I.Enabled = False
        txt_MR_DI.Enabled = False
        txt_NMG.Enabled = False
        txt_NT.Enabled = False
        txt_NEO.Enabled = False

        'Start EEA program
        start_EEA()         'Main.vb

        Me.Close()

    End Sub


    Private Sub btn_close_Click(ByVal sender As System.Object, ByVal e As Sys
tem.EventArgs) Handles btn_close.Click
        Me.Close()
    End Sub


    'Option for printing populations
    Private Sub chk_popchk_CheckedChanged(ByVal sender As System.Object, ByVa
l e As System.EventArgs) Handles chk_popchk.CheckedChanged
        If chk_popchk.Checked = True Then
            chk_POPI.Enabled = True
            chk_POPD.Enabled = True
            chk_POPDI.Enabled = True
            txt_POP_DU.Enabled = True
        Else
            chk_POPI.Checked = False
            chk_POPD.Checked = False
            chk_POPDI.Checked = False

            chk_POPI.Enabled = False
            chk_POPD.Enabled = False
```

```
            chk_POPDI.Enabled = False
            txt_POP_DU.Enabled = False
        End If
    End Sub

End Class
```

Table C.2 The program code in Main.vb.

```
Module Main


    'Definition of Symbiont and Endosymbiont
    Public Structure Symbiotic_D
        Dim gene_D() As Integer
        Dim vehicle_D() As Integer
        Dim arrival_time_D() As Double
        Dim travel_time_D As Double
        Dim fitness_D As Double
    End Structure

    Public Structure Symbiotic_I
        Dim gene_I() As Integer
        Dim vehicle_I() As Integer
        Dim arrival_time_I() As Double
        Dim wait_time_I() As Double
        Dim travel_time_I As Double
        Dim fitness_I As Double
    End Structure

    Public Structure Endosymbiotic
        Dim gene_D() As Integer
        Dim gene_I() As Integer
        Dim vehicle_D() As Integer
        Dim vehicle_I() As Integer
        Dim arrival_time_D() As Double
        Dim arrival_time_I() As Double
        Dim wait_time_I() As Double
        Dim travel_time_D As Double
        Dim travel_time_I As Double
        Dim fitness_DI As Double
    End Structure

    Structure Location_Index
        Dim loc_X As Integer
        Dim loc_Y As Integer
    End Structure

    Structure Pair
        Dim p1 As Location_Index
        Dim p2 As Location_Index
    End Structure


    Public Const BIGM As Double = 999999.999
    Public sum_entr_sol As Integer = 0
    Public stop_gen As Integer = 0



    Sub start_EEA()
        '=======================================
        ' Problem variables
        '=======================================

        'number of customers requiring delivery
        Dim Num_Customer_D As Integer
        'number of customers requiring delivery and installation
        Dim Num_Customer_I As Integer
```

```
        'capacity of delivery vehicle
        Dim Capacity_Vehicle_D As Integer
        'installation time at a customer (service time)
        Dim Installation_Time_I As Double
        'service level
        Dim Service_Level As Double
        'maximum operation time
        Dim Max_Operation_Time As Double

        'Distance and Customer Demand
        Dim Distance(Num_Customer_D, Num_Customer_D) As Double
        Dim Customer_Demand(Num_Customer_D - 1) As Integer


        '=========================================
        'Algorithm variables
        '=========================================

        Dim Pop_Size As Integer         'population size
        Dim Num_Generation As Integer   'maximum # of generation

        'Crossover rate for populations
        Dim Crossover_Rate_D As Single
        Dim Crossover_Rate_I As Single
        Dim Crossover_Rate_DI As Single

        'Mutation rate for populations
        Dim Mutation_Rate_D As Single
        Dim Mutation_Rate_I As Single
        Dim Mutation_Rate_DI As Single

        '3X3 matrix for the neighborhood
        Dim Nb_D(2, 2) As Symbiotic_D
        Dim Nb_I(2, 2) As Symbiotic_I
        Dim Nb_DI(2, 2) As Endosymbiotic


        '=========================================
        'Program variables
        '=========================================

        Dim best_solution As Endosymbiotic
        Dim best_comb As Endosymbiotic
        Dim tNb_D(2, 2) As Symbiotic_D

        Dim max_trial As Integer
        Dim current_trial As Integer
        Dim current_generation As Integer = 0
        Dim loc_Nb As Location_Index
        Dim Nb_Index(2, 2) As Location_Index

        Dim temp_best_travel As Double
        Dim temp_fit_DI As Double

        Dim input_file As String = ""
        Dim temp_file_name As String = ""

        Dim start_time As Date
        Dim end_time As Date
        Dim elapsed_time As TimeSpan

        Dim num_offspring As Integer
        Dim check_pop As Integer
```

```
        '=======================================
        'Program functions
        '=======================================

        'Obtain program parameters from interface(Main.vb)
        program_setup(Num_Customer_D, Num_Customer_I, Capacity_Vehicle_D, Ins
tallation_Time_I, Service_Level, Max_Operation_Time, Crossover_Rate_D, Crosso
ver_Rate_I, Crossover_Rate_DI, Mutation_Rate_D, Mutation_Rate_I, Mutation_Rat
e_DI, Num_Generation, max_trial, num_offspring, input_file, check_pop, Pop_Si
ze)

        'Define populations
        Dim POP_D(Pop_Size - 1, Pop_Size - 1) As Symbiotic_D
        Dim POP_I(Pop_Size - 1, Pop_Size - 1) As Symbiotic_I
        Dim POP_DI(Pop_Size - 1, Pop_Size - 1) As Endosymbiotic

        'redefine distances and customers' demands
        ReDim Distance(Num_Customer_D, Num_Customer_D)
        ReDim Customer_Demand(Num_Customer_D - 1)

        'Data loading from input file (Dt_Input.vb)
        If read_data(input_file, Customer_Demand, Distance, Num_Customer_D, N
um_Customer_I) = False Then
            GoTo PROGRAM_END
        End If


        For current_trial = 1 To max_trial

            'Show progress of trial
            EEA.pgb_trial.Value = current_trial

            current_generation = 0
            sum_entr_sol = 0
            temp_best_travel = BIGM


            'Initialization of output files (Dt_Output.vb)
            temp_file_name = Outfile_setup(current_trial, Num_Customer_D, Num
_Customer_I, Pop_Size)

            'Define structure of Populations, Neighborhood, and Best solution
 (Initialization.vb)
            define_population(POP_DI, POP_D, POP_I, Pop_Size, Num_Customer_D,
 Num_Customer_I)
            best_solution = define_individual_DI(Num_Customer_D, Num_Customer
_I)

            'Record starting time
            start_time = Now

            'Generate initial population (Initialization.vb)
            population_generation_D(POP_D, Pop_Size)
            population_generation_I(POP_I, Pop_Size)
            population_generation_DI(POP_DI, Pop_Size)

            'prefix initial solutions (Initialization.vb)
            fix_solution(POP_D, POP_I, POP_DI, Pop_Size, Distance, Customer_D
emand, Capacity_Vehicle_D)
```

```
            'Evaluate fitness value of initial population(Fitness.vb)
            cal_fitness_sub(Pop_Size, POP_D, POP_I, Capacity_Vehicle_D, Insta
llation_Time_I, Service_Level, Max_Operation_Time, Distance, Customer_Demand)
            cal_fitness_etr(Pop_Size, POP_DI, Capacity_Vehicle_D, Installatio
n_Time_I, Service_Level, Max_Operation_Time, Distance, Customer_Demand)

            'print all fitness values of the initial population (Utility.vb)
            file_population_check(POP_DI, POP_D, POP_I, Pop_Size, current_gen
eration, loc_Nb, check_pop, temp_file_name)


            For current_generation = 1 To Num_Generation

                'Selection of a random location to generate neighborhoods (Co
operation.vb)
                loc_Nb = random_location(Pop_Size)
                Nb_Index = set_loc_neighbor(loc_Nb, Pop_Size)

                'Creation of set of neighborhoods (Cooperation.vb)
                Nb_D = create_neighborhood_D(POP_D, Nb_Index)
                Nb_I = create_neighborhood_I(POP_I, Nb_Index)
                Nb_DI = create_neighborhood_DI(POP_DI, Nb_Index)

                'Cooperation between Nb_D and Nb_I (Cooperation.vb)
                best_comb = cooperation_sub(Nb_D, Nb_I, Installation_Time_I,
Service_Level, Max_Operation_Time, Distance, Capacity_Vehicle_D, Customer_Dem
and)

                'update best solution with best combination from Nb_D and Nb_
I (Cooperation.vb)
                best_solution = update_best(best_comb, best_solution)

                'update best solution with Nb_DI (Cooperation.vb)
                cal_fitness_etr(3, Nb_DI, Capacity_Vehicle_D, Installation_Ti
me_I, Service_Level, Max_Operation_Time, Distance, Customer_Demand)
                best_solution = update_best_DI(Nb_DI, best_solution)

                'Competition between Nb_DI and best solution (Cooperation.vb)
                competition(Nb_DI, best_comb)


                'Duplicate Nb_D for evolution of Nb_I (Initialization.vb)
                tNb_D = copy_Nb_D(tNb_D, Nb_D)

                'evolution of Nb_D(Evolution_D.vb)
                evolve_D(Nb_D, Nb_I, Crossover_Rate_D, Mutation_Rate_D, Capac
ity_Vehicle_D, Max_Operation_Time, Distance, Customer_Demand, _
                Installation_Time_I, Service_Level)

                'evolution of Nb_I(Evolution_I.vb)
                evolve_I(Nb_I, tNb_D, Crossover_Rate_I, Mutation_Rate_I, Max_
Operation_Time, Distance, Customer_Demand, _
                Installation_Time_I, Service_Level)

                'evolution of Nb_DI(Evolution_DI.vb)
                evolve_DI(Nb_DI, Crossover_Rate_DI, Mutation_Rate_DI, Capacit
y_Vehicle_D, Max_Operation_Time, Distance, Customer_Demand, _
                Installation_Time_I, Service_Level)

                'Release neighborhoods (Cooperation.vb)
                release_neighborhood_D(Nb_D, POP_D, Nb_Index)
```

```
                release_neighborhood_I(Nb_I, POP_I, Nb_Index)
                release_neighborhood_DI(Nb_DI, POP_DI, Nb_Index)


                'Show progress of generation
                If CInt(current_generation Mod (Num_Generation / 20)) = 0 The
n
                    EEA.pgb_generation.Value = current_generation
                End If


                'Optional report per generation (Dt_Output.vb)
                temp_best_travel = generation_record(current_generation, best
_solution, temp_best_travel, temp_file_name, start_time, end_time)

                'print all fitness values of the population (Utility.vb)
                file_population_check(POP_DI, POP_D, POP_I, Pop_Size, current
_generation, loc_Nb, check_pop, temp_file_name)


                'Checking termination conditions (optional)
                If sum_entr_sol > num_offspring And num_offspring > 0 Then

                    stop_gen = current_generation
                    current_generation = Num_Generation

                End If

            Next

            'Check running time
            end_time = Now
            elapsed_time = end_time.Subtract(start_time)

            'Check final solution
            temp_fit_DI = best_solution.fitness_DI
            best_solution = fitness_etr_D(best_solution, Capacity_Vehicle_D,
Max_Operation_Time, Customer_Demand, Distance)
            best_solution = fitness_etr_I(best_solution, Installation_Time_I,
 Service_Level, Max_Operation_Time, Distance)

            If temp_fit_DI <> best_solution.fitness_DI Then

                best_solution.fitness_DI = 0.0

            End If

            'Generate final report on output file (Dt_Output.vb)
            final_report(best_solution, elapsed_time, temp_file_name)


        Next


PROGRAM_END:
    End Sub


    'Load parameters from program interface
    Sub program_setup(ByRef Num_Customer_D As Integer, ByRef Num_Customer_I A
s Integer, ByRef Capacity_Vehicle_D As Integer, ByRef Installation_Time_I As
Double, ByRef Service_Level As Double, ByRef Max_Operation_Time As Double, By
Ref Crossover_rate_D As Double, ByRef Crossover_rate_I As Double, ByRef Cross
```

```
over_rate_DI As Double, ByRef Mutation_rate_D As Double, ByRef Mutation_rate_
I As Double, ByRef Mutation_rate_DI As Double, ByRef Num_Generation As Intege
r, ByRef trial As Integer, ByRef Num_offspring As Integer, ByRef input_file A
s String, ByRef check_pop As Integer, ByRef Pop_size As Integer)

        Try
            Num_Customer_D = CInt(EEA.txt_NCD.Text)
            Num_Customer_I = CInt(EEA.txt_NCI.Text)
            Crossover_rate_D = CDec(EEA.txt_XR_D.Text)
            Crossover_rate_I = CDec(EEA.txt_XR_I.Text)
            Crossover_rate_DI = CDec(EEA.txt_XR_DI.Text)
            Mutation_rate_D = CDec(EEA.txt_MR_D.Text)
            Mutation_rate_I = CDec(EEA.txt_MR_I.Text)
            Mutation_rate_DI = CDec(EEA.txt_MR_DI.Text)
            Num_Generation = CInt(EEA.txt_NMG.Text)
            trial = CInt(EEA.txt_NT.Text)
            Num_offspring = CInt(EEA.txt_NEO.Text)
            Capacity_Vehicle_D = CInt(EEA.txt_CDV.Text)
            Installation_Time_I = CDec(EEA.txt_IT.Text)
            Service_Level = CDec(EEA.txt_SL.Text)
            Max_Operation_Time = CDec(EEA.txt_MOT.Text)
            check_pop = CInt(EEA.txt_POP_DU.Text)
            Pop_size = CInt(EEA.txt_popsize.Text)

        Catch ex As Exception
            MessageBox.Show("Input data error")
            EEA.Close()
        End Try

        If Num_Customer_D <= 0 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If Num_Customer_I <= 0 Or Num_Customer_I > Num_Customer_D Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If Crossover_rate_D <= 0 Or Crossover_rate_D > 1 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If Crossover_rate_I <= 0 Or Crossover_rate_I > 1 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If Crossover_rate_DI <= 0 Or Crossover_rate_DI > 1 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If Mutation_rate_D <= 0 Or Mutation_rate_D > 1 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If Mutation_rate_I <= 0 Or Mutation_rate_I > 1 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If
```

```
        If Mutation_rate_DI <= 0 Or Mutation_rate_DI > 1 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If Num_Generation <= 0 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If trial <= 0 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If Num_offspring < 0 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If Capacity_Vehicle_D <= 0 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If Installation_Time_I < 0 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If Service_Level < 0 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If Max_Operation_Time < 0 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        If Pop_size <= 3 Then
            MessageBox.Show("Input data error")
            EEA.Close()
        End If

        input_file = CStr(Num_Customer_D) + "-" + CStr(Num_Customer_I) + "-1
0.txt"

        EEA.pgb_generation.Maximum = Num_Generation
        EEA.pgb_trial.Maximum = trial

    End Sub


End Module
```

Table C.3 The program code in Cooperation.vb.

```
Module Cooperation

    'Select random location in population
    Function random_location(ByVal Pop_size As Integer)
        Dim temp_loc As Location_Index

        temp_loc.loc_X = CInt(Int((Rnd() * Pop_size)))
        temp_loc.loc_Y = CInt(Int((Rnd() * Pop_size)))

        Return temp_loc

    End Function


    'Generate neighborhood in POP-D
    Function create_neighborhood_D(ByVal POP_D(,) As Symbiotic_D, ByRef Nb_In
dex(,) As Location_Index)

        Dim temp_NB(2, 2) As Symbiotic_D
        Dim i, j As Integer

        For i = 0 To 2
            For j = 0 To 2
                temp_NB(i, j) = POP_D(Nb_Index(i, j).loc_X, Nb_Index(i, j).lo
c_Y)
            Next
        Next

        Return temp_NB

    End Function


    'Generate neighborhood in POP-I
    Function create_neighborhood_I(ByVal POP_I(,) As Symbiotic_I, ByRef Nb_In
dex(,) As Location_Index)

        Dim temp_NB(2, 2) As Symbiotic_I
        Dim i, j As Integer

        For i = 0 To 2
            For j = 0 To 2
                temp_NB(i, j) = POP_I(Nb_Index(i, j).loc_X, Nb_Index(i, j).lo
c_Y)
            Next
        Next

        Return temp_NB

    End Function


    'Generate neighborhood in POP-DI
    Function create_neighborhood_DI(ByVal POP_DI(,) As Endosymbiotic, ByRef N
b_Index(,) As Location_Index)

        Dim temp_NB(2, 2) As Endosymbiotic
        Dim i, j As Integer

        For i = 0 To 2
```

```
            For j = 0 To 2
                temp_NB(i, j) = POP_DI(Nb_Index(i, j).loc_X, Nb_Index(i, j).l
oc_Y)
            Next
        Next

        Return temp_NB

    End Function


    'Locations of neighborhood in Toroidal grid
    Function set_loc_neighbor(ByVal loc_Nb As Location_Index, ByVal Pop_size
As Integer)

        Dim temp_nb_set(2, 2) As Location_Index
        Dim i, j As Integer

        For i = 0 To 2
            For j = 0 To 2
                If i = 0 Then
                    temp_nb_set(i, j).loc_X = (loc_Nb.loc_X + (Pop_size - 1))
 Mod Pop_size
                ElseIf i = 1 Then
                    temp_nb_set(i, j).loc_X = loc_Nb.loc_X
                Else
                    temp_nb_set(i, j).loc_X = (loc_Nb.loc_X + 1) Mod Pop_size
                End If

                If j = 0 Then
                    temp_nb_set(i, j).loc_Y = (loc_Nb.loc_Y + (Pop_size - 1))
 Mod Pop_size
                ElseIf j = 1 Then
                    temp_nb_set(i, j).loc_Y = loc_Nb.loc_Y
                Else
                    temp_nb_set(i, j).loc_Y = (loc_Nb.loc_Y + 1) Mod Pop_size
                End If
            Next
        Next

        Return temp_nb_set

    End Function


    'Coorperation between two subproblems (Delivery and Installation)
    Function cooperation_sub(ByRef Nb_D(,) As Symbiotic_D, ByRef Nb_I(,) As S
ymbiotic_I, ByVal Installation_Time_I As Double, _
    ByVal Service_Level As Double, ByVal Max_Operation_Time As Double, ByVal
Distance(,) As Double, ByVal Capacity_Vehicle_D As Integer, _
    ByVal Customer_Demand() As Integer)

        'Dim temp_symbiotic_D As Symbiotic_D
        Dim temp_Indv_I As Symbiotic_I
        Dim best_comb As Endosymbiotic
        Dim i, j, u, v As Integer
        Dim s, t, p, q As Integer

        best_comb = define_individual_DI(Nb_D(0, 0).gene_D.Length, Nb_I(0,
0).gene_I.Length)
        best_comb.fitness_DI = 0.0

        For i = 0 To 2
```

```
            For j = 0 To 2

                'Select individual from Neighborhood D
                For u = 0 To 2
                    For v = 0 To 2
                        'Select individual from Neighborhood I
                        temp_Indv_I = Nb_I(u, v)
                        temp_Indv_I = fitness_sub_I(Nb_D(i, j), temp_Indv_I,
Installation_Time_I, Service_Level, Max_Operation_Time, Distance)

                        If temp_Indv_I.fitness_I > best_comb.fitness_DI Then
                            best_comb = combine_sub(Nb_D(i, j), temp_Indv_I)

                            'store last index of best combination
                            s = i
                            t = j
                            p = u
                            q = v
                        End If
                    Next
                Next
            Next
        Next

        best_comb = fitness_etr_D(best_comb, Capacity_Vehicle_D, Max_Operatio
n_Time, Customer_Demand, Distance)
        best_comb = fitness_etr_I(best_comb, Installation_Time_I, Service_Lev
el, Max_Operation_Time, Distance)

        Return best_comb

    End Function


    'Make a complet solution from partial solutions in two subproblems
    Function combine_sub(ByVal Indv_D As Symbiotic_D, ByVal Indv_I As Symbiot
ic_I)

        Dim temp_comb As Endosymbiotic
        Dim i As Integer

        temp_comb = define_individual_DI(Indv_D.gene_D.Length, Indv_I.gene_I.
Length)

        For i = 0 To Indv_D.gene_D.Length - 1
            temp_comb.arrival_time_D(i) = Indv_D.arrival_time_D(i)
            temp_comb.gene_D(i) = Indv_D.gene_D(i)
            temp_comb.vehicle_D(i) = Indv_D.vehicle_D(i)
        Next

        temp_comb.travel_time_D = Indv_D.travel_time_D

        For i = 0 To Indv_I.gene_I.Length - 1
            temp_comb.arrival_time_I(i) = Indv_I.arrival_time_I(i)
            temp_comb.gene_I(i) = Indv_I.gene_I(i)
            temp_comb.vehicle_I(i) = Indv_I.vehicle_I(i)
            temp_comb.wait_time_I(i) = Indv_I.wait_time_I(i)
        Next

        temp_comb.travel_time_I = Indv_I.travel_time_I

        temp_comb.fitness_DI = Indv_I.fitness_I
```

```
        Return temp_comb

    End Function


    'Divide a complete soltuion into partial solutions for two subproblems
    Sub separate_sub(ByRef Comb As Endosymbiotic, ByRef Indv_D As Symbiotic_
D, ByRef Indv_I As Symbiotic_I)

        Dim i As Integer

        For i = 0 To Indv_D.gene_D.Length - 1
            Indv_D.arrival_time_D(i) = Comb.arrival_time_D(i)
            Indv_D.gene_D(i) = Comb.gene_D(i)
            Indv_D.vehicle_D(i) = Comb.vehicle_D(i)
        Next

        Indv_D.travel_time_D = Comb.travel_time_D

        For i = 0 To Indv_I.gene_I.Length - 1
            Indv_I.arrival_time_I(i) = Comb.arrival_time_I(i)
            Indv_I.gene_I(i) = Comb.gene_I(i)
            Indv_I.vehicle_I(i) = Comb.vehicle_I(i)
            Indv_I.wait_time_I(i) = Comb.wait_time_I(i)
        Next

        Indv_I.travel_time_I = Comb.travel_time_I

        Indv_I.fitness_I = Comb.fitness_DI

    End Sub


    'Update best solution (compare current best solution and solution from PO
P-DI
    Function update_best(ByRef best_comb As Endosymbiotic, ByRef best_solutio
n As Endosymbiotic)

        If best_comb.fitness_DI > best_solution.fitness_DI Then
            Return best_comb
        Else
            Return best_solution
        End If

    End Function


    'Update best solution (compare current best solution and solutiosn in nei
ghborhood from POP-DI
    Function update_best_DI(ByRef Nb_DI(,) As Endosymbiotic, ByRef best_solut
ion As Endosymbiotic)

        Dim i, j As Integer

        For i = 0 To 2
            For j = 0 To 2
                best_solution = update_best(Nb_DI(i, j), best_solution)
            Next
        Next

        Return best_solution

    End Function
```

```
    'Competition between best combination from the cooperation and individual
s of neighborhood in POP-DI
    Sub competition(ByRef Nb_DI(,) As Endosymbiotic, ByRef best_comb As Endos
ymbiotic)

        Dim loc_worst_comb As Location_Index
        Dim temp_fitness As Double
        Dim i, j As Integer

        temp_fitness = BIGM

        For i = 0 To 2
            For j = 0 To 2
                If Nb_DI(i, j).fitness_DI < temp_fitness Then
                    temp_fitness = Nb_DI(i, j).fitness_DI
                    loc_worst_comb.loc_X = i
                    loc_worst_comb.loc_Y = j
                End If
            Next
        Next

        If best_comb.fitness_DI > Nb_DI(loc_worst_comb.loc_X, loc_worst_comb.
loc_Y).fitness_DI Then

            Nb_DI(loc_worst_comb.loc_X, loc_worst_comb.loc_Y) = copy_DI2DI(Nb
_DI(loc_worst_comb.loc_X, loc_worst_comb.loc_Y), best_comb)

        End If

    End Sub


    'Release current neighborhoods to populations (POP-D)
    Sub release_neighborhood_D(ByRef Nb_D(,) As Symbiotic_D, ByRef POP_D(,) A
s Symbiotic_D, ByVal Nb_Index(,) As Location_Index)

        Dim i, j As Integer

        For i = 0 To 2
            For j = 0 To 2
                POP_D(Nb_Index(i, j).loc_X, Nb_Index(i, j).loc_Y) = copy_D2D
(POP_D(Nb_Index(i, j).loc_X, Nb_Index(i, j).loc_Y), Nb_D(i, j))
            Next
        Next

    End Sub


    'Release current neighborhoods to populations (POP-I)
    Sub release_neighborhood_I(ByRef Nb_I(,) As Symbiotic_I, ByRef POP_I(,) A
s Symbiotic_I, ByVal Nb_Index(,) As Location_Index)

        Dim i, j As Integer

        For i = 0 To 2
            For j = 0 To 2
                POP_I(Nb_Index(i, j).loc_X, Nb_Index(i, j).loc_Y) = copy_I2I
(POP_I(Nb_Index(i, j).loc_X, Nb_Index(i, j).loc_Y), Nb_I(i, j))
            Next
        Next
```

```
    End Sub


    'Release current neighborhoods to populations (POP-DI)
    Sub release_neighborhood_DI(ByRef Nb_DI(,) As Endosymbiotic, ByRef POP_DI
(,) As Endosymbiotic, ByVal Nb_Index(,) As Location_Index)

        Dim i, j As Integer

        For i = 0 To 2
            For j = 0 To 2
                POP_DI(Nb_Index(i, j).loc_X, Nb_Index(i, j).loc_Y) = copy_DI2
DI(POP_DI(Nb_Index(i, j).loc_X, Nb_Index(i, j).loc_Y), Nb_DI(i, j))
            Next
        Next

    End Sub

End Module
```

Table C.4 The program code in Initialization.vb.

```
Module Initialization

    'Define Individual Structures (POP-D, POP-I, and POP-DI)
    Sub define_population(ByRef POP_DI(,) As Endosymbiotic, ByRef POP_D(,) As
 Symbiotic_D, ByRef POP_I(,) As Symbiotic_I, _
    ByVal Pop_size As Integer, ByVal Num_Customer_D As Integer, ByVal Num_Cus
tomer_I As Integer)

        define_pop_D(POP_D, Pop_size, Num_Customer_D)
        define_pop_I(POP_I, Pop_size, Num_Customer_I)
        define_pop_DI(POP_DI, Pop_size, Num_Customer_D, Num_Customer_I)

    End Sub

    'Define POP-D
    Sub define_pop_D(ByRef POP_D(,) As Symbiotic_D, ByVal Pop_size As Intege
r, ByVal Num_Customer_D As Integer)

        Dim i, j As Integer

        For i = 0 To Pop_size - 1
            For j = 0 To Pop_size - 1
                POP_D(i, j) = define_individual_D(Num_Customer_D)
            Next
        Next

    End Sub


    'Define individual D
    Function define_individual_D(ByVal Num_Customer_D As Integer)

        Dim init_ind_D As Symbiotic_D
        Dim i As Integer

        ReDim init_ind_D.gene_D(Num_Customer_D - 1)
        ReDim init_ind_D.vehicle_D(Num_Customer_D - 1)
        ReDim init_ind_D.arrival_time_D(Num_Customer_D - 1)

        For i = 0 To Num_Customer_D - 1
            init_ind_D.gene_D(i) = 0
            init_ind_D.vehicle_D(i) = 0
            init_ind_D.arrival_time_D(i) = 0.0
        Next

        init_ind_D.travel_time_D = 0.0
        init_ind_D.fitness_D = 0.0

        Return init_ind_D

    End Function


    'Define POP-I
    Sub define_pop_I(ByRef POP_I(,) As Symbiotic_I, ByVal Pop_size As Intege
r, ByVal Num_Customer_I As Integer)

        Dim i, j As Integer

        For i = 0 To Pop_size - 1
            For j = 0 To Pop_size - 1
```

```
                POP_I(i, j) = define_individual_I(Num_Customer_I)
            Next
        Next

    End Sub


    'Define individual I
    Function define_individual_I(ByVal Num_Customer_I As Integer)

        Dim init_ind_I As Symbiotic_I
        Dim i As Integer

        ReDim init_ind_I.gene_I(Num_Customer_I - 1)
        ReDim init_ind_I.vehicle_I(Num_Customer_I - 1)
        ReDim init_ind_I.arrival_time_I(Num_Customer_I - 1)
        ReDim init_ind_I.wait_time_I(Num_Customer_I - 1)

        For i = 0 To Num_Customer_I - 1
            init_ind_I.gene_I(i) = 0
            init_ind_I.vehicle_I(i) = 0
            init_ind_I.arrival_time_I(i) = 0.0
            init_ind_I.wait_time_I(i) = 0.0
        Next

        init_ind_I.travel_time_I = 0.0
        init_ind_I.fitness_I = 0.0

        Return init_ind_I

    End Function


    'Define POP-DI
    Sub define_pop_DI(ByRef POP_DI(,) As Endosymbiotic, ByVal Pop_size As Int
eger, ByVal Num_Customer_D As Integer, _
    ByVal Num_Customer_I As Integer)
        Dim i, j As Integer

        For i = 0 To Pop_size - 1
            For j = 0 To Pop_size - 1
                POP_DI(i, j) = define_individual_DI(Num_Customer_D, Num_Custo
mer_I)
            Next
        Next

    End Sub


    'Define individual DI
    Function define_individual_DI(ByVal Num_Customer_D As Integer, ByVal Num_
Customer_I As Integer)

        Dim init_ind_DI As Endosymbiotic
        Dim i As Integer

        ReDim init_ind_DI.gene_D(Num_Customer_D - 1)
        ReDim init_ind_DI.gene_I(Num_Customer_I - 1)
        ReDim init_ind_DI.vehicle_D(Num_Customer_D - 1)
        ReDim init_ind_DI.vehicle_I(Num_Customer_I - 1)
        ReDim init_ind_DI.arrival_time_D(Num_Customer_D - 1)
        ReDim init_ind_DI.arrival_time_I(Num_Customer_I - 1)
        ReDim init_ind_DI.wait_time_I(Num_Customer_I - 1)
```

```
        For i = 0 To Num_Customer_D - 1
            init_ind_DI.gene_D(i) = 0
            init_ind_DI.vehicle_D(i) = 0
            init_ind_DI.arrival_time_D(i) = 0.0
        Next
        init_ind_DI.travel_time_D = 0.0

        For i = 0 To Num_Customer_I - 1
            init_ind_DI.gene_I(i) = 0
            init_ind_DI.vehicle_I(i) = 0
            init_ind_DI.arrival_time_I(i) = 0.0
            init_ind_DI.wait_time_I(i) = 0.0
        Next

        init_ind_DI.travel_time_I = 0.0
        init_ind_DI.fitness_DI = 0.0

        Return init_ind_DI

    End Function


    'Generate initial POP-D
    Sub population_generation_D(ByRef POP_D(,) As Symbiotic_D, ByVal Pop_size
 As Integer)

        Dim i, j, k, rand_gene, temp As Integer

        For i = 0 To Pop_size - 1
            For j = 0 To Pop_size - 1
                'put sequential customer index
                For k = 0 To POP_D(i, j).gene_D.Length - 1
                    POP_D(i, j).gene_D(k) = k
                Next

                'suffle the customer index
                For k = 0 To POP_D(i, j).gene_D.Length - 1
                    rand_gene = POP_D(i, j).gene_D.Length

                    While (rand_gene = POP_D(i, j).gene_D.Length)
                        rand_gene = CInt(Int((Rnd() * POP_D(i, j).gene_D.Leng
th)))
                    End While

                    temp = POP_D(i, j).gene_D(k)
                    POP_D(i, j).gene_D(k) = POP_D(i, j).gene_D(rand_gene)
                    POP_D(i, j).gene_D(rand_gene) = temp
                Next
            Next
        Next

    End Sub


    'Generate initial POP-I
    Sub population_generation_I(ByRef POP_I(,) As Symbiotic_I, ByVal Pop_size
 As Integer)

        Dim i, j, k, rand_gene, temp As Integer

        For i = 0 To Pop_size - 1
            For j = 0 To Pop_size - 1
```

```
                For k = 0 To POP_I(i, j).gene_I.Length - 1
                    POP_I(i, j).gene_I(k) = k
                Next

                For k = 0 To POP_I(i, j).gene_I.Length - 1
                    rand_gene = POP_I(i, j).gene_I.Length

                    While (rand_gene = POP_I(i, j).gene_I.Length)
                        rand_gene = CInt(Int((Rnd() * POP_I(i, j).gene_I.Leng
th)))
                    End While

                    temp = POP_I(i, j).gene_I(k)
                    POP_I(i, j).gene_I(k) = POP_I(i, j).gene_I(rand_gene)
                    POP_I(i, j).gene_I(rand_gene) = temp
                Next
            Next
        Next

    End Sub


    'Generate initial POP-DI
    Sub population_generation_DI(ByRef POP_DI(,) As Endosymbiotic, ByVal Pop_
size As Integer)

        Dim i, j, k, rand_gene, temp As Integer

        For i = 0 To Pop_size - 1
            For j = 0 To Pop_size - 1
                'Delivery part
                For k = 0 To POP_DI(i, j).gene_D.Length - 1
                    POP_DI(i, j).gene_D(k) = k
                Next
                For k = 0 To POP_DI(i, j).gene_D.Length - 1

                    rand_gene = POP_DI(i, j).gene_D.Length

                    While (rand_gene = POP_DI(i, j).gene_D.Length)
                        rand_gene = CInt(Int((Rnd() * POP_DI(i, j).gene_D.Len
gth)))
                    End While

                    temp = POP_DI(i, j).gene_D(k)
                    POP_DI(i, j).gene_D(k) = POP_DI(i, j).gene_D(rand_gene)
                    POP_DI(i, j).gene_D(rand_gene) = temp
                Next

                'Installation part
                For k = 0 To POP_DI(i, j).gene_I.Length - 1
                    POP_DI(i, j).gene_I(k) = k
                Next
                For k = 0 To POP_DI(i, j).gene_I.Length - 1

                    rand_gene = POP_DI(i, j).gene_I.Length

                    While (rand_gene = POP_DI(i, j).gene_I.Length)
                        rand_gene = CInt(Int((Rnd() * POP_DI(i, j).gene_I.Len
gth)))
                    End While

                    temp = POP_DI(i, j).gene_I(k)
                    POP_DI(i, j).gene_I(k) = POP_DI(i, j).gene_I(rand_gene)
```

```
                    POP_DI(i, j).gene_I(rand_gene) = temp
                Next
            Next
        Next
    End Sub


    'Copy functions
    Function copy_D2D(ByRef Indv_Dt As Symbiotic_D, ByVal Indv_Ds As Symbioti
c_D)
        ' Indv_Dt : target individual for sub problem D
        ' Indv_Ds : source individual for sub problem D

        Dim i As Integer

        For i = 0 To Indv_Ds.gene_D.Length - 1
            Indv_Dt.gene_D(i) = Indv_Ds.gene_D(i)
            Indv_Dt.arrival_time_D(i) = Indv_Ds.arrival_time_D(i)
            Indv_Dt.vehicle_D(i) = Indv_Ds.vehicle_D(i)
        Next
        Indv_Dt.travel_time_D = Indv_Ds.travel_time_D
        Indv_Dt.fitness_D = Indv_Ds.fitness_D

        Return Indv_Dt

    End Function

    Function copy_I2I(ByRef Indv_It As Symbiotic_I, ByVal Indv_Is As Symbioti
c_I)
        ' Indv_It : target individual for sub problem I
        ' Indv_Is : source individual for sub problem I

        Dim i As Integer

        For i = 0 To Indv_It.gene_I.Length - 1
            Indv_It.gene_I(i) = Indv_Is.gene_I(i)
            Indv_It.arrival_time_I(i) = Indv_Is.arrival_time_I(i)
            Indv_It.vehicle_I(i) = Indv_Is.vehicle_I(i)
            Indv_It.wait_time_I(i) = Indv_Is.wait_time_I(i)
        Next
        Indv_It.travel_time_I = Indv_Is.travel_time_I
        Indv_It.fitness_I = Indv_Is.fitness_I

        Return Indv_It

    End Function

    Function copy_DI2DI(ByRef Indv_DIt As Endosymbiotic, ByVal Indv_DIs As En
dosymbiotic)
        ' Indv_DIt : target individual for entire problem DI
        ' Indv_DIs : source individual for entire problem DI

        Dim i As Integer

        For i = 0 To Indv_DIt.gene_D.Length - 1
            Indv_DIt.gene_D(i) = Indv_DIs.gene_D(i)
            Indv_DIt.arrival_time_D(i) = Indv_DIs.arrival_time_D(i)
            Indv_DIt.vehicle_D(i) = Indv_DIs.vehicle_D(i)
        Next

        For i = 0 To Indv_DIt.gene_I.Length - 1
            Indv_DIt.gene_I(i) = Indv_DIs.gene_I(i)
            Indv_DIt.arrival_time_I(i) = Indv_DIs.arrival_time_I(i)
```

```
            Indv_DIt.vehicle_I(i) = Indv_DIs.vehicle_I(i)
            Indv_DIt.wait_time_I(i) = Indv_DIs.wait_time_I(i)
        Next

        Indv_DIt.travel_time_I = Indv_DIs.travel_time_I
        Indv_DIt.travel_time_D = Indv_DIs.travel_time_D
        Indv_DIt.fitness_DI = Indv_DIs.fitness_DI

        Return Indv_DIt

    End Function

    Function copy_Nb_D(ByRef tNb_D(,) As Symbiotic_D, ByVal Nb_D(,) As Symbio
tic_D)

        Dim i, j As Integer

        For i = 0 To 2
            For j = 0 To 2
                tNb_D(i, j) = Nb_D(i, j)
            Next
        Next

        Return tNb_D
    End Function

End Module
```

Table C.5 The program code in Fitness.vb.

```
Module Fitness

    'Fitness test individuals in subproblems having same location index
    Sub cal_fitness_sub(ByVal Pop_Size As Integer, ByRef POP_D(,) As Symbioti
c_D, ByRef POP_I(,) As Symbiotic_I, _
    ByVal Capacity_Vehicle_D As Integer, ByVal Installation_Time_I As Double,
 ByVal Service_Level As Double, _
    ByVal Max_Operation_Time As Double, ByVal Distance(,) As Double, ByVal Cu
stomer_Demand() As Integer)

        Dim i, j As Integer

        For i = 0 To Pop_Size - 1
            For j = 0 To Pop_Size - 1
                cal_fitness_sub_indv(POP_D(i, j), POP_I(i, j), Capacity_Vehic
le_D, Installation_Time_I, Service_Level, Max_Operation_Time, Distance, Custo
mer_Demand)
            Next
        Next

    End Sub

    'Fitness test a pair of partial solutions from subproblems D & I
    Sub cal_fitness_sub_indv(ByRef Indv_D As Symbiotic_D, ByRef Indv_I As Sym
biotic_I, ByVal Capacity_Vehicle_D As Integer, _
    ByVal Installation_Time_I As Double, ByVal Service_Level As Double, ByVal
 Max_Operation_Time As Double, _
    ByVal Distance(,) As Double, ByVal Customer_Demand() As Integer)

        Indv_D = fitness_sub_D(Indv_D, Capacity_Vehicle_D, Max_Operation_Tim
e, Customer_Demand, Distance)
        Indv_I = fitness_sub_I(Indv_D, Indv_I, Installation_Time_I, Service_L
evel, Max_Operation_Time, Distance)

        Indv_D.fitness_D = Indv_I.fitness_I

        Return

    End Sub


    'Part of Fitness function for subproblem D
    Function fitness_sub_D(ByRef Indv_D As Symbiotic_D, ByVal Capacity_Vehicl
e_D As Integer, ByVal Max_Operation_Time As Double, _
    ByVal Customer_Demand() As Integer, ByVal Distance(,) As Double)

        Dim customer_index As Integer
        Dim vehicle_num As Integer
        Dim vehicle_load As Integer

        Dim temp_travel As Double
        Dim total_travel As Double

        customer_index = 0
        vehicle_num = 1

        vehicle_load = Customer_Demand(Indv_D.gene_D(customer_index))

        temp_travel = Distance(Indv_D.gene_D.Length, Indv_D.gene_D(customer_i
ndex))
        total_travel = Distance(Indv_D.gene_D.Length, Indv_D.gene_D(customer_
```

```
index))
        Indv_D.vehicle_D(customer_index) = vehicle_num
        Indv_D.arrival_time_D(customer_index) = temp_travel

        For customer_index = 0 To Indv_D.gene_D.Length - 2

            If temp_travel + Distance(Indv_D.gene_D(customer_index), Indv_D.g
ene_D(customer_index + 1)) + _
            Distance(Indv_D.gene_D(customer_index + 1), Indv_D.gene_D.Length)
 <= Max_Operation_Time And _
            vehicle_load + Customer_Demand(Indv_D.gene_D(customer_index + 1))
 <= Capacity_Vehicle_D Then

                vehicle_load = vehicle_load + Customer_Demand(Indv_D.gene_D(c
ustomer_index + 1))
                temp_travel = temp_travel + Distance(Indv_D.gene_D(customer_i
ndex), Indv_D.gene_D(customer_index + 1))
                total_travel = total_travel + Distance(Indv_D.gene_D(customer
_index), Indv_D.gene_D(customer_index + 1))
                Indv_D.vehicle_D(customer_index + 1) = vehicle_num
                Indv_D.arrival_time_D(customer_index + 1) = temp_travel
            Else
                vehicle_load = Customer_Demand(Indv_D.gene_D(customer_index +
 1))
                temp_travel = Distance(Indv_D.gene_D.Length, Indv_D.gene_D(cu
stomer_index + 1))
                total_travel = total_travel + Distance(Indv_D.gene_D(customer
_index), Indv_D.gene_D.Length) + _
                Distance(Indv_D.gene_D.Length, Indv_D.gene_D(customer_index +
 1))

                vehicle_num = vehicle_num + 1

                Indv_D.vehicle_D(customer_index + 1) = vehicle_num
                Indv_D.arrival_time_D(customer_index + 1) = temp_travel
            End If
        Next

        total_travel = total_travel + Distance(Indv_D.gene_D(Indv_D.gene_D.Le
ngth - 1), Indv_D.gene_D.Length)
        Indv_D.travel_time_D = total_travel

        Return Indv_D

    End Function


    'Part of Fitness function for subproblem I
    Function fitness_sub_I(ByRef Indv_D As Symbiotic_D, ByRef Indv_I As Symbi
otic_I, ByVal Installation_Time_I As Double, _
    ByVal Service_Level As Double, ByVal Max_Operation_Time As Double, ByVal
Distance(,) As Double)

        Dim arr_time_D(Indv_I.gene_I.Length - 1) As Double

        Dim i As Integer
        Dim customer_index As Integer
        Dim vehicle_num As Integer
        Dim temp_wait As Double
        Dim temp_travel As Double
        Dim total_travel As Double

        'Make a time table of delivery vehicles for the installation customer
```

```
s
        For i = 0 To Indv_D.gene_D.Length - 1
            If Indv_D.gene_D(i) < Indv_I.gene_I.Length Then
                arr_time_D(Indv_D.gene_D(i)) = Indv_D.arrival_time_D(i)
            End If
        Next

        customer_index = 0
        vehicle_num = 1

        If Distance(Indv_D.gene_D.Length, Indv_I.gene_I(customer_index)) >= a
rr_time_D(Indv_I.gene_I(customer_index)) Then
            temp_travel = Distance(Indv_D.gene_D.Length, Indv_I.gene_I(custom
er_index))
            total_travel = Distance(Indv_D.gene_D.Length, Indv_I.gene_I(custo
mer_index))
            Indv_I.vehicle_I(customer_index) = vehicle_num
            Indv_I.arrival_time_I(customer_index) = temp_travel
            Indv_I.wait_time_I(customer_index) = 0.0

            temp_travel += Installation_Time_I
            total_travel += Installation_Time_I
        Else
            temp_wait = arr_time_D(Indv_I.gene_I(customer_index)) - Distance
(Indv_D.gene_D.Length, Indv_I.gene_I(customer_index))
            temp_travel = arr_time_D(Indv_I.gene_I(customer_index))
            total_travel = arr_time_D(Indv_I.gene_I(customer_index))
            Indv_I.vehicle_I(customer_index) = vehicle_num
            Indv_I.arrival_time_I(customer_index) = temp_travel - temp_wait
            Indv_I.wait_time_I(customer_index) = temp_wait

            temp_travel += Installation_Time_I
            total_travel += Installation_Time_I
        End If

        For customer_index = 0 To Indv_I.gene_I.Length - 2
            If temp_travel + Distance(Indv_I.gene_I(customer_index), Indv_I.g
ene_I(customer_index + 1)) >= arr_time_D(Indv_I.gene_I(customer_index + 1)) A
nd _
            temp_travel + Distance(Indv_I.gene_I(customer_index), Indv_I.gene
_I(customer_index + 1)) <= arr_time_D(Indv_I.gene_I(customer_index + 1)) + Se
rvice_Level And _
            temp_travel + Distance(Indv_I.gene_I(customer_index), Indv_I.gene
_I(customer_index + 1)) + Installation_Time_I + Distance(Indv_I.gene_I(custom
er_index + 1), Indv_D.gene_D.Length) <= Max_Operation_Time Then

                temp_travel = temp_travel + Distance(Indv_I.gene_I(customer_i
ndex), Indv_I.gene_I(customer_index + 1))
                total_travel = total_travel + Distance(Indv_I.gene_I(customer
_index), Indv_I.gene_I(customer_index + 1))

                Indv_I.vehicle_I(customer_index + 1) = vehicle_num
                Indv_I.arrival_time_I(customer_index + 1) = temp_travel
                Indv_I.wait_time_I(customer_index + 1) = 0.0

                temp_travel += Installation_Time_I
                total_travel += Installation_Time_I

            ElseIf temp_travel + Distance(Indv_I.gene_I(customer_index), Indv
_I.gene_I(customer_index + 1)) < arr_time_D(Indv_I.gene_I(customer_index +
1)) And _
            arr_time_D(Indv_I.gene_I(customer_index + 1)) + Installation_Time
_I + Distance(Indv_I.gene_I(customer_index + 1), Indv_D.gene_D.Length) <= Max
```

```
_Operation_Time Then

                temp_wait = arr_time_D(Indv_I.gene_I(customer_index + 1)) - t
emp_travel - Distance(Indv_I.gene_I(customer_index), Indv_I.gene_I(customer_i
ndex + 1))
                temp_travel = arr_time_D(Indv_I.gene_I(customer_index + 1))
                total_travel = total_travel + Distance(Indv_I.gene_I(customer
_index), Indv_I.gene_I(customer_index + 1)) + temp_wait

                Indv_I.vehicle_I(customer_index + 1) = vehicle_num
                Indv_I.arrival_time_I(customer_index + 1) = temp_travel
                Indv_I.wait_time_I(customer_index + 1) = temp_wait

                temp_travel += Installation_Time_I
                total_travel += Installation_Time_I

            ElseIf Distance(Indv_D.gene_D.Length, Indv_I.gene_I(customer_inde
x + 1)) >= arr_time_D(Indv_I.gene_I(customer_index + 1)) Then

                vehicle_num = vehicle_num + 1

                temp_travel = Distance(Indv_D.gene_D.Length, Indv_I.gene_I(cu
stomer_index + 1))
                total_travel = total_travel + Distance(Indv_I.gene_I(customer
_index), Indv_D.gene_D.Length) + Distance(Indv_D.gene_D.Length, Indv_I.gene_I
(customer_index))

                Indv_I.vehicle_I(customer_index + 1) = vehicle_num
                Indv_I.arrival_time_I(customer_index + 1) = temp_travel
                Indv_I.wait_time_I(customer_index + 1) = 0.0

                temp_travel += Installation_Time_I
                total_travel += Installation_Time_I
            Else
                vehicle_num = vehicle_num + 1

                temp_wait = arr_time_D(Indv_I.gene_I(customer_index + 1)) - D
istance(Indv_D.gene_D.Length, Indv_I.gene_I(customer_index + 1))
                temp_travel = arr_time_D(Indv_I.gene_I(customer_index + 1))
                total_travel = total_travel + Distance(Indv_I.gene_I(customer
_index), Indv_D.gene_D.Length) + Distance(Indv_D.gene_D.Length, Indv_I.gene_I
(customer_index + 1)) + _
                temp_wait

                Indv_I.vehicle_I(customer_index + 1) = vehicle_num
                Indv_I.arrival_time_I(customer_index + 1) = temp_travel
                Indv_I.wait_time_I(customer_index + 1) = temp_wait

                temp_travel += Installation_Time_I
                total_travel += Installation_Time_I
            End If
        Next

        total_travel = total_travel + Distance(Indv_I.gene_I(Indv_I.gene_I.Le
ngth - 1), Indv_D.gene_D.Length)
        Indv_I.travel_time_I = total_travel
        Indv_I.fitness_I = 1 / (Indv_I.travel_time_I + Indv_D.travel_time_D +
 (Indv_D.vehicle_D(Indv_D.gene_D.Length - 1) * 100.0) + (Indv_I.vehicle_I(Ind
v_I.gene_I.Length - 1) * 100.0))

        Return Indv_I

    End Function
```

```
    'Fitness test of individuals in POP-DI
    Sub cal_fitness_etr(ByRef Pop_Size As Integer, ByRef POP_DI(,) As Endosym
biotic, ByVal Capacity_Vehicle_D As Integer, _
    ByVal Installation_Time_I As Double, ByVal Service_Level As Double, ByVal
 Max_Operation_Time As Double, ByVal Distance(,) As Double, _
    ByVal Customer_Demand() As Integer)

        Dim i, j As Integer

        For i = 0 To Pop_Size - 1
            For j = 0 To Pop_Size - 1
                POP_DI(i, j) = fitness_etr_D(POP_DI(i, j), Capacity_Vehicle_
D, Max_Operation_Time, Customer_Demand, Distance)
                POP_DI(i, j) = fitness_etr_I(POP_DI(i, j), Installation_Time_
I, Service_Level, Max_Operation_Time, Distance)
            Next
        Next

    End Sub


    'Part of fitness function for delivery portion in individual for POP-DI
    Function fitness_etr_D(ByRef Indv_DI As Endosymbiotic, ByVal Capacity_Veh
icle_D As Integer, ByVal Max_Operation_Time As Double, ByRef customer_demand
() As Integer, _
    ByVal Distance(,) As Double)

        Dim customer_index_D As Integer     'current customer index
        Dim vehicle_num_D As Integer
        Dim vehicle_load As Integer
        Dim temp_travel_D As Double
        Dim total_travel_D As Double

        customer_index_D = 0
        vehicle_num_D = 1

        vehicle_load = customer_demand(Indv_DI.gene_D(customer_index_D))
        temp_travel_D = Distance(Indv_DI.gene_D.Length, Indv_DI.gene_D(custom
er_index_D))
        total_travel_D = Distance(Indv_DI.gene_D.Length, Indv_DI.gene_D(custo
mer_index_D))
        Indv_DI.vehicle_D(customer_index_D) = vehicle_num_D
        Indv_DI.arrival_time_D(customer_index_D) = temp_travel_D

        For customer_index_D = 0 To Indv_DI.gene_D.Length - 2

            If temp_travel_D + Distance(Indv_DI.gene_D(customer_index_D), Ind
v_DI.gene_D(customer_index_D + 1)) + _
            Distance(Indv_DI.gene_D(customer_index_D + 1), Indv_DI.gene_D.Len
gth) <= Max_Operation_Time And _
            vehicle_load + customer_demand(Indv_DI.gene_D(customer_index_D +
1)) <= Capacity_Vehicle_D Then

                vehicle_load = vehicle_load + customer_demand(Indv_DI.gene_D
(customer_index_D + 1))
                temp_travel_D = temp_travel_D + Distance(Indv_DI.gene_D(custo
mer_index_D), Indv_DI.gene_D(customer_index_D + 1))
                total_travel_D = total_travel_D + Distance(Indv_DI.gene_D(cus
tomer_index_D), Indv_DI.gene_D(customer_index_D + 1))
                Indv_DI.vehicle_D(customer_index_D + 1) = vehicle_num_D
                Indv_DI.arrival_time_D(customer_index_D + 1) = temp_travel_D
```

```
            Else

                vehicle_load = customer_demand(Indv_DI.gene_D(customer_index_
D + 1))
                temp_travel_D = Distance(Indv_DI.gene_D.Length, Indv_DI.gene_
D(customer_index_D + 1))
                total_travel_D = total_travel_D + Distance(Indv_DI.gene_D(cus
tomer_index_D), Indv_DI.gene_D.Length) + _
                Distance(Indv_DI.gene_D.Length, Indv_DI.gene_D(customer_index
_D + 1))

                vehicle_num_D = vehicle_num_D + 1

                Indv_DI.vehicle_D(customer_index_D + 1) = vehicle_num_D
                Indv_DI.arrival_time_D(customer_index_D + 1) = temp_travel_D

            End If
        Next

        total_travel_D = total_travel_D + Distance(Indv_DI.gene_D(Indv_DI.gen
e_D.Length - 1), Indv_DI.gene_D.Length)
        Indv_DI.travel_time_D = total_travel_D

        Return Indv_DI

    End Function


    'Part of fitness function for installation portion in individual for POP-
DI
    Function fitness_etr_I(ByRef Indv_DI As Endosymbiotic, ByVal Installation
_Time_I As Double, ByVal Service_Level As Double, _
    ByVal Max_Operation_Time As Double, ByVal Distance(,) As Double)

        Dim arr_time_D(Indv_DI.gene_I.Length - 1) As Double

        Dim i As Integer
        Dim temp_wait As Double
        Dim customer_index_I As Integer
        Dim vehicle_num_I As Integer
        Dim temp_travel_I As Double
        Dim total_travel_I As Double

        'Make a time table of delivery vehicles for the installation customer
s
        For i = 0 To Indv_DI.gene_D.Length - 1
            If Indv_DI.gene_D(i) < Indv_DI.gene_I.Length Then
                arr_time_D(Indv_DI.gene_D(i)) = Indv_DI.arrival_time_D(i)
            End If
        Next

        customer_index_I = 0
        vehicle_num_I = 1

        If Distance(Indv_DI.gene_D.Length, Indv_DI.gene_I(customer_index_I))
>= arr_time_D(Indv_DI.gene_I(customer_index_I)) Then

            temp_travel_I = Distance(Indv_DI.gene_D.Length, Indv_DI.gene_I(cu
stomer_index_I))
            total_travel_I = Distance(Indv_DI.gene_D.Length, Indv_DI.gene_I(c
ustomer_index_I))
            Indv_DI.vehicle_I(customer_index_I) = vehicle_num_I
```

```
            Indv_DI.arrival_time_I(customer_index_I) = temp_travel_I
            Indv_DI.wait_time_I(customer_index_I) = 0.0

            temp_travel_I += Installation_Time_I
            total_travel_I += Installation_Time_I

        Else

            temp_wait = arr_time_D(Indv_DI.gene_I(customer_index_I)) - Distan
ce(Indv_DI.gene_D.Length, Indv_DI.gene_I(customer_index_I))
            temp_travel_I = arr_time_D(Indv_DI.gene_I(customer_index_I))
            total_travel_I = arr_time_D(Indv_DI.gene_I(customer_index_I))
            Indv_DI.vehicle_I(customer_index_I) = vehicle_num_I
            Indv_DI.arrival_time_I(customer_index_I) = temp_travel_I - temp_w
ait
            Indv_DI.wait_time_I(customer_index_I) = temp_wait

            temp_travel_I += Installation_Time_I
            total_travel_I += Installation_Time_I

        End If

        For customer_index_I = 0 To Indv_DI.gene_I.Length - 2

            If temp_travel_I + Distance(Indv_DI.gene_I(customer_index_I), Ind
v_DI.gene_I(customer_index_I + 1)) >= arr_time_D(Indv_DI.gene_I(customer_inde
x_I + 1)) And _
            temp_travel_I + Distance(Indv_DI.gene_I(customer_index_I), Indv_D
I.gene_I(customer_index_I + 1)) <= arr_time_D(Indv_DI.gene_I(customer_index_I
 + 1)) + Service_Level And _
            temp_travel_I + Distance(Indv_DI.gene_I(customer_index_I), Indv_D
I.gene_I(customer_index_I + 1)) + Installation_Time_I + Distance(Indv_DI.gene
_I(customer_index_I + 1), Indv_DI.gene_D.Length) <= Max_Operation_Time Then

                temp_travel_I = temp_travel_I + Distance(Indv_DI.gene_I(custo
mer_index_I), Indv_DI.gene_I(customer_index_I + 1))
                total_travel_I = total_travel_I + Distance(Indv_DI.gene_I(cus
tomer_index_I), Indv_DI.gene_I(customer_index_I + 1))

                Indv_DI.vehicle_I(customer_index_I + 1) = vehicle_num_I
                Indv_DI.arrival_time_I(customer_index_I + 1) = temp_travel_I
                Indv_DI.wait_time_I(customer_index_I + 1) = 0.0

                temp_travel_I += Installation_Time_I
                total_travel_I += Installation_Time_I

            ElseIf temp_travel_I + Distance(Indv_DI.gene_I(customer_index_I),
 Indv_DI.gene_I(customer_index_I + 1)) < arr_time_D(Indv_DI.gene_I(customer_i
ndex_I + 1)) And _
            arr_time_D(Indv_DI.gene_I(customer_index_I + 1)) + Installation_T
ime_I + Distance(Indv_DI.gene_I(customer_index_I + 1), Indv_DI.gene_D.Length)
 <= Max_Operation_Time Then

                temp_wait = arr_time_D(Indv_DI.gene_I(customer_index_I + 1))
- temp_travel_I - Distance(Indv_DI.gene_I(customer_index_I), Indv_DI.gene_I(c
ustomer_index_I + 1))
                temp_travel_I = arr_time_D(Indv_DI.gene_I(customer_index_I +
1))
                total_travel_I = total_travel_I + Distance(Indv_DI.gene_I(cus
tomer_index_I), Indv_DI.gene_I(customer_index_I + 1)) + temp_wait

                Indv_DI.vehicle_I(customer_index_I + 1) = vehicle_num_I
                Indv_DI.arrival_time_I(customer_index_I + 1) = temp_travel_I
```

```
                Indv_DI.wait_time_I(customer_index_I + 1) = temp_wait

                temp_travel_I += Installation_Time_I
                total_travel_I += Installation_Time_I

            ElseIf Distance(Indv_DI.gene_D.Length, Indv_DI.gene_I(customer_in
dex_I + 1)) >= arr_time_D(Indv_DI.gene_I(customer_index_I + 1)) Then

                vehicle_num_I = vehicle_num_I + 1

                temp_travel_I = Distance(Indv_DI.gene_D.Length, Indv_DI.gene_
I(customer_index_I + 1))
                total_travel_I = total_travel_I + Distance(Indv_DI.gene_I(cus
tomer_index_I), Indv_DI.gene_D.Length) + Distance(Indv_DI.gene_D.Length, Indv
_DI.gene_I(customer_index_I))

                Indv_DI.vehicle_I(customer_index_I + 1) = vehicle_num_I
                Indv_DI.arrival_time_I(customer_index_I + 1) = temp_travel_I
                Indv_DI.wait_time_I(customer_index_I) = 0.0

                temp_travel_I += Installation_Time_I
                total_travel_I += Installation_Time_I

            Else

                vehicle_num_I = vehicle_num_I + 1

                temp_wait = arr_time_D(Indv_DI.gene_I(customer_index_I + 1))
- Distance(Indv_DI.gene_D.Length, Indv_DI.gene_I(customer_index_I + 1))
                temp_travel_I = arr_time_D(Indv_DI.gene_I(customer_index_I +
1))
                total_travel_I = total_travel_I + Distance(Indv_DI.gene_I(cus
tomer_index_I), Indv_DI.gene_D.Length) + Distance(Indv_DI.gene_D.Length, Indv
_DI.gene_I(customer_index_I + 1)) + _
                temp_wait

                Indv_DI.vehicle_I(customer_index_I + 1) = vehicle_num_I
                Indv_DI.arrival_time_I(customer_index_I + 1) = temp_travel_I
                Indv_DI.wait_time_I(customer_index_I + 1) = temp_wait

                temp_travel_I += Installation_Time_I
                total_travel_I += Installation_Time_I

            End If
        Next

        total_travel_I = total_travel_I + Distance(Indv_DI.gene_I(Indv_DI.gen
e_I.Length - 1), Indv_DI.gene_D.Length)
        Indv_DI.travel_time_I = total_travel_I
        Indv_DI.fitness_DI = 1 / (Indv_DI.travel_time_I + Indv_DI.travel_time
_D + (Indv_DI.vehicle_D(Indv_DI.gene_D.Length - 1) * 100.0) + (Indv_DI.vehicl
e_I(Indv_DI.gene_I.Length - 1) * 100.0))

        Return Indv_DI

    End Function


End Module
```

Table C.6 The program code in Improvement.vb.

```
Module Improvement

    'Fix initial solutions (main)
    Sub fix_solution(ByRef POP_D(,) As Symbiotic_D, ByRef POP_I(,) As Symbiot
ic_I, ByRef POP_DI(,) As Endosymbiotic, ByVal Pop_Size As Integer, ByVal Dist
ance(,) As Double, ByVal Customer_Demand() As Integer, ByVal Capacity_Vehicle
_D As Integer)

        fix_solution_D(POP_D, Pop_Size, Distance, Customer_Demand, Capacity_V
ehicle_D)
        fix_solution_DI(POP_DI, Pop_Size, Distance, Customer_Demand, Capacity
_Vehicle_D)

    End Sub


    'Fix initial solutions in subproblem D
    Sub fix_solution_D(ByRef POP_D(,) As Symbiotic_D, ByVal Pop_Size As Integ
er, ByVal Distance(,) As Double, ByVal Customer_Demand() As Integer, ByVal Ca
pacity_Vehicle_D As Integer)

        Dim i, j, k, l As Integer
        Dim temp_gene As Integer
        Dim temp_dist As Double
        Dim temp_load As Integer
        Dim short_gene As Integer

        For i = 0 To Pop_Size - 1
            For j = 0 To Pop_Size - 1
                temp_load = 0
                For k = 0 To POP_D(0, 0).gene_D.Length - 2
                    temp_dist = 99999999
                    short_gene = 99999999
                    temp_load = temp_load + Customer_Demand(POP_D(i, j).gene_
D(k))
                    For l = k + 1 To POP_D(0, 0).gene_D.Length - 1
                        If temp_dist > Distance(POP_D(i, j).gene_D(k), POP_D
(i, j).gene_D(l)) Then
                            short_gene = l
                            temp_dist = Distance(POP_D(i, j).gene_D(k), POP_D
(i, j).gene_D(l))
                        End If
                    Next

                    If temp_load + Customer_Demand(POP_D(i, j).gene_D(short_g
ene)) <= Capacity_Vehicle_D Then
                        temp_gene = POP_D(i, j).gene_D(k + 1)
                        POP_D(i, j).gene_D(k + 1) = POP_D(i, j).gene_D(short_
gene)
                        POP_D(i, j).gene_D(short_gene) = temp_gene
                    Else
                        temp_load = 0
                    End If
                Next
            Next
        Next

    End Sub
```

```
    'Fix initial solutions in POP-DI
    Sub fix_solution_DI(ByRef POP_DI(,) As Endosymbiotic, ByVal Pop_Size As I
nteger, ByVal Distance(,) As Double, ByVal Customer_Demand() As Integer, ByVa
l Capacity_Vehicle_D As Integer)

        Dim i, j, k, l As Integer
        Dim temp_gene As Integer
        Dim temp_dist As Double
        Dim temp_load As Integer
        Dim short_gene As Integer


        For i = 0 To Pop_Size - 1
            For j = 0 To Pop_Size - 1
                temp_load = 0
                For k = 0 To POP_DI(0, 0).gene_D.Length - 2
                    temp_dist = 99999999
                    short_gene = 99999999
                    temp_load = temp_load + Customer_Demand(POP_DI(i, j).gene
_D(k))
                    For l = k + 1 To POP_DI(0, 0).gene_D.Length - 1
                        If temp_dist > Distance(POP_DI(i, j).gene_D(k), POP_D
I(i, j).gene_D(l)) Then
                            short_gene = l
                            temp_dist = Distance(POP_DI(i, j).gene_D(k), POP_
DI(i, j).gene_D(l))
                        End If
                    Next

                    If temp_load + Customer_Demand(POP_DI(i, j).gene_D(short_
gene)) <= Capacity_Vehicle_D Then
                        temp_gene = POP_DI(i, j).gene_D(k + 1)
                        POP_DI(i, j).gene_D(k + 1) = POP_DI(i, j).gene_D(shor
t_gene)
                        POP_DI(i, j).gene_D(short_gene) = temp_gene
                    Else
                        temp_load = 0
                    End If
                Next
            Next
        Next

    End Sub

End Module
```

Table C.7 The program code in Evolution_D.vb.

```
Module Evolution_D

    'Evolution of Nb_D (main function)
    Sub evolve_D(ByRef Nb_D(,) As Symbiotic_D, ByRef Nb_I(,) As Symbiotic_I,
ByVal Crossover_Rate As Single, _
    ByVal Mutation_Rate As Single, ByVal Capacity_Vehicle_D As Integer, ByVal
 Max_Operation_Time As Double, _
    ByVal Distance(,) As Double, ByVal Customer_Demand() As Integer, ByVal In
stallation_Time_I As Double, _
    ByVal Service_Level As Double)

        Dim temp_Nb_D(2, 2) As Symbiotic_D
        Dim temp_ind_D As Symbiotic_D
        Dim temp_ind_I As Symbiotic_I

        Dim selection_set As Pair

        Dim offspring As Symbiotic_D
        Dim compete_indv As Location_Index
        Dim rand_value As Double
        Dim i, j, k As Integer

        Dim num_xover As Integer = 18


        'Duplicate Neighborhood D
        For i = 0 To 2
            For j = 0 To 2
                temp_Nb_D(i, j) = define_individual_D(Nb_D(0, 0).gene_D.Lengt
h)
                temp_Nb_D(i, j) = Nb_D(i, j)
            Next
        Next

        'Crossover
        For i = 0 To num_xover
            rand_value = Rnd()
            If rand_value <= Crossover_Rate Then
                'Create parent set for crossover
                selection_set = select_pair_D(temp_Nb_D)

                offspring = crossover_D(temp_Nb_D(selection_set.p1.loc_X, sel
ection_set.p1.loc_Y), temp_Nb_D(selection_set.p2.loc_X, _
                selection_set.p2.loc_Y))

                'Select a neighhor to compete with offspring randomly
                compete_indv = roulette_inverse_D(Nb_D)

                'replace with comparing fitness
                Nb_D(compete_indv.loc_X, compete_indv.loc_Y) = compare_D(Nb_D
(compete_indv.loc_X, compete_indv.loc_Y), offspring, _
                Nb_I(compete_indv.loc_X, compete_indv.loc_Y), Capacity_Vehicl
e_D, Max_Operation_Time, Distance, Customer_Demand, Installation_Time_I, Serv
ice_Level)
            End If
        Next

        'Mutation
        For i = 0 To 2
            For j = 0 To 2
                rand_value = Rnd()
```

```
                If rand_value <= Mutation_Rate Then
                    k = 0
                    For k = 0 To 3
                        temp_ind_D = Nb_D(i, j)
                        temp_ind_I = Nb_I(i, j)

                        temp_ind_D = mutation_D(temp_ind_D)

                        Nb_D(i, j) = fitness_sub_D(Nb_D(i, j), Capacity_Vehic
le_D, Max_Operation_Time, Customer_Demand, Distance)
                        Nb_I(i, j) = fitness_sub_I(Nb_D(i, j), Nb_I(i, j), In
stallation_Time_I, Service_Level, Max_Operation_Time, Distance)
                        Nb_D(i, j).fitness_D = Nb_I(i, j).fitness_I

                        temp_ind_D = fitness_sub_D(temp_ind_D, Capacity_Vehic
le_D, Max_Operation_Time, Customer_Demand, Distance)
                        temp_ind_I = fitness_sub_I(temp_ind_D, temp_ind_I, In
stallation_Time_I, Service_Level, Max_Operation_Time, Distance)
                        temp_ind_D.fitness_D = temp_ind_I.fitness_I

                        If temp_ind_D.fitness_D > Nb_D(i, j).fitness_D Then
                            Nb_D(i, j) = temp_ind_D
                            Nb_I(i, j) = temp_ind_I
                            k = 3
                        Else
                            k = k + 1
                        End If
                    Next
                End If
            Next
        Next

    End Sub


    'Create set of parents for crossover
    Function select_pair_D(ByVal Nb_D(,) As Symbiotic_D)

        Dim selection_set As Pair

        selection_set.p1 = roulette_D(Nb_D)
        selection_set.p2 = roulette_D(Nb_D)

        While selection_set.p1.loc_X = selection_set.p2.loc_X And selection_s
et.p1.loc_Y = selection_set.p2.loc_Y
            selection_set.p2 = roulette_D(Nb_D)
        End While

        Return selection_set

    End Function


    'Select good individual with probability
    Function roulette_D(ByVal Nb_D(,) As Symbiotic_D)

        Dim selected_loc As Location_Index
        Dim i, j As Integer
        Dim rand_value As Double
        Dim temp_cur_fitness As Double
        Dim temp_sum_fitness As Double

        temp_sum_fitness = 0.0
```

```
        For i = 0 To 2
            For j = 0 To 2
                temp_sum_fitness += Nb_D(i, j).fitness_D
            Next
        Next

        rand_value = Rnd()
        temp_cur_fitness = 0.0

        For i = 0 To 2
            For j = 0 To 2
                If rand_value > temp_cur_fitness And rand_value <= temp_cur_f
itness + (Nb_D(i, j).fitness_D / temp_sum_fitness) Then
                    selected_loc.loc_X = i
                    selected_loc.loc_Y = j
                End If
                temp_cur_fitness += (Nb_D(i, j).fitness_D / temp_sum_fitness)
            Next
        Next

        Return selected_loc

    End Function


    'Select bad individual with probability
    Function roulette_inverse_D(ByVal Nb_D(,) As Symbiotic_D)

        Dim selected_loc As Location_Index

        Dim i, j As Integer
        Dim rand_value As Double
        Dim temp_cur_fitness As Double
        Dim temp_sum_fitness As Double

        temp_sum_fitness = 0.0

        For i = 0 To 2
            For j = 0 To 2
                temp_sum_fitness += 1 / Nb_D(i, j).fitness_D
            Next
        Next

        rand_value = Rnd()
        temp_cur_fitness = 0.0

        For i = 0 To 2
            For j = 0 To 2
                If rand_value > temp_cur_fitness And rand_value <= temp_cur_f
itness + ((1 / Nb_D(i, j).fitness_D) / temp_sum_fitness) Then
                    selected_loc.loc_X = i
                    selected_loc.loc_Y = j
                End If
                temp_cur_fitness += ((1 / Nb_D(i, j).fitness_D) / temp_sum_fi
tness)
            Next
        Next

        Return selected_loc

    End Function
```

```
    'Select worst individual in neighborhood
    Function select_worst_D(ByVal Nb_D(,) As Symbiotic_D)

        Dim selected_loc As Location_Index

        Dim i, j As Integer
        Dim temp_fitness As Double


        temp_fitness = 0.0

        selected_loc.loc_X = 0
        selected_loc.loc_Y = 0
        For i = 0 To 2
            For j = 0 To 2
                If Nb_D(i, j).fitness_D > temp_fitness Then
                    temp_fitness = Nb_D(i, j).fitness_D
                    selected_loc.loc_X = i
                    selected_loc.loc_Y = j
                End If
            Next
        Next

        Return selected_loc

    End Function


    'Crossover (2 steps)
    Function crossover_D(ByVal Nb_D1 As Symbiotic_D, ByVal Nb_D2 As Symbiotic
_D)

        Dim temp_child As Symbiotic_D

        temp_child = xover_hybrid_p1_D(Nb_D1)
        temp_child = xover_hybrid_p2_D(Nb_D2, temp_child)

        'Add different crossover functions below with "if ~ endif"

        Return temp_child

    End Function

    'Crossover with the parent 1 (step 1)
    Function xover_hybrid_p1_D(ByVal Nb_D1 As Symbiotic_D)

        Dim temp_child As Symbiotic_D

        Dim temp_gene(Nb_D1.gene_D.Length - 1) As Integer
        Dim num_vehicle_used As Integer
        Dim selected_vehicle As Integer
        Dim temp_count4child As Integer
        Dim temp_count4remain As Integer
        Dim one_cut_point As Integer

        Dim i As Integer

        'temp_child = define_individual_D(temp_child, Nb_D1.gene_D.Length)
        temp_child = define_individual_D(Nb_D1.gene_D.Length)

        'Vehicle selection based crossover
        num_vehicle_used = Nb_D1.vehicle_D(Nb_D1.gene_D.Length - 1)
```

```
        selected_vehicle = CInt(Int((Rnd() * (num_vehicle_used - 1))) + 1)


        temp_count4child = 0
        temp_count4remain = 0

        For i = 0 To Nb_D1.gene_D.Length - 1
            If Nb_D1.vehicle_D(i) = selected_vehicle Then
                temp_child.gene_D(temp_count4child) = Nb_D1.gene_D(i)
                temp_count4child += 1
            Else
                temp_gene(temp_count4remain) = Nb_D1.gene_D(i)
                temp_count4remain += 1
            End If
        Next

        'One cut point crossover
         one_cut_point = CInt(Int((Rnd() * (temp_count4remain - 2))) + 1)

        For i = 0 To one_cut_point
            temp_child.gene_D(temp_count4child) = temp_gene(i)
            temp_count4child += 1
        Next

        For i = temp_count4child To Nb_D1.gene_D.Length - 1
            temp_child.gene_D(i) = -1
        Next

        Return temp_child

    End Function


    'Crossover with the parent 2 (step 2)
    Function xover_hybrid_p2_D(ByVal Nb_D2 As Symbiotic_D, ByRef temp_child A
s Symbiotic_D)

        Dim i, j, k As Integer
        Dim plag As Boolean

        For i = 0 To Nb_D2.gene_D.Length - 1
            plag = False

            If temp_child.gene_D(i) = -1 Then
                k = 0
                While temp_child.gene_D(i) = -1
                    For j = 0 To i - 1
                        If Nb_D2.gene_D(k) = temp_child.gene_D(j) Then
                            plag = True
                        End If
                    Next

                    If plag Then
                        plag = False
                        k += 1
                    Else
                        temp_child.gene_D(i) = Nb_D2.gene_D(k)
                    End If
                End While

            End If
```

```
        Next

        Return temp_child
    End Function


    'Decide whether new offspring can enter neighborhood or not
    Function compare_D(ByRef Indv_D As Symbiotic_D, ByRef offspring As Symbio
tic_D, ByRef Indv_I As Symbiotic_I, ByVal Capacity_Vehicle_D As Integer, _
    ByVal Max_Operation_Time As Double, ByVal Distance(,) As Double, ByVal Cu
stomer_Demand() As Integer, ByVal Installation_Time_I As Double, _
    ByVal Service_Level As Double)

        Dim temp_Indv_I As Symbiotic_I

        temp_Indv_I = define_individual_I(Indv_I.gene_I.Length)

        temp_Indv_I = copy_I2I(temp_Indv_I, Indv_I)
        offspring = fitness_sub_D(offspring, Capacity_Vehicle_D, Max_Operatio
n_Time, Customer_Demand, Distance)
        temp_Indv_I = fitness_sub_I(offspring, temp_Indv_I, Installation_Time
_I, Service_Level, Max_Operation_Time, Distance)
        offspring.fitness_D = temp_Indv_I.fitness_I

        If offspring.travel_time_D < Indv_D.travel_time_D Then
            Indv_D = offspring
            Main.sum_entr_sol = Main.sum_entr_sol + 1
        End If

        Return Indv_D

    End Function


    'Mutation
    Function mutation_D(ByRef Indv_D As Symbiotic_D)

        Indv_D = mutation_exchange_D(Indv_D)

        'Add different mutation functions below with "if ~ endif"

        Return Indv_D

    End Function


    'Mutation (exchange mutation)
    Function mutation_exchange_D(ByRef Indv_D As Symbiotic_D)

        Dim rand_select_1 As Integer
        Dim rand_select_2 As Integer
        Dim temp_gene As Integer

        rand_select_1 = CInt(Int(Rnd() * (Indv_D.gene_D.Length - 1)))
        rand_select_2 = rand_select_1

        While (rand_select_2 = rand_select_1)
            rand_select_2 = CInt(Int(Rnd() * (Indv_D.gene_D.Length - 1)))
        End While

        temp_gene = Indv_D.gene_D(rand_select_1)
        Indv_D.gene_D(rand_select_1) = Indv_D.gene_D(rand_select_2)
```

```
        Indv_D.gene_D(rand_select_2) = temp_gene

        Return Indv_D
    End Function

End Module
```

Table C.8 The program code in Evolution_I.vb.

```
Module Evolution_I

    'Evolution of Nb_I (main function)
    Sub evolve_I(ByRef Nb_I(,) As Symbiotic_I, ByRef Nb_D(,) As Symbiotic_D,
ByVal Crossover_Rate As Single, _
    ByVal Mutation_Rate As Single, ByVal Max_Operation_Time As Double, ByVal
Distance(,) As Double, _
    ByVal Customer_Demand() As Integer, ByVal Installation_Time_I As Double,
ByVal Service_Level As Double)

        Dim temp_Nb_I(2, 2) As Symbiotic_I

        Dim temp_ind_I As Symbiotic_I

        Dim selection_set As Pair
        Dim offspring As Symbiotic_I
        Dim compete_indv As Location_Index
        Dim i, j, k As Integer

        Dim num_xover As Integer = 18


        'Duplicate Neighborhood I
        For i = 0 To 2
            For j = 0 To 2
                temp_Nb_I(i, j) = define_individual_I(Nb_I(0, 0).gene_I.Lengt
h)
                temp_Nb_I(i, j) = copy_I2I(temp_Nb_I(i, j), Nb_I(i, j))
            Next
        Next

        'Crossover
        For i = 0 To num_xover
            If Rnd() <= Crossover_Rate Then
                'Create set of parent set for crossover
                selection_set = select_pair_I(temp_Nb_I)

                offspring = crossover_I(temp_Nb_I(selection_set.p1.loc_X, sel
ection_set.p1.loc_Y), temp_Nb_I(selection_set.p2.loc_X, _
                selection_set.p2.loc_Y))

                'Select a neighhor to compete with offspring randomly
                compete_indv = roulette_inverse_I(Nb_I)

                compare_I(Nb_I(compete_indv.loc_X, compete_indv.loc_Y), offsp
ring, Nb_D(compete_indv.loc_X, compete_indv.loc_Y), _
                Max_Operation_Time, Distance, Installation_Time_I, Service_Le
vel)
            End If
        Next

        'Mutation
        For i = 0 To 2
            For j = 0 To 2
                If Rnd() <= Mutation_Rate Then
                    k = 0
                    For k = 0 To 3
                        temp_ind_I = Nb_I(i, j)
                        temp_ind_I = mutation_I(temp_ind_I)

                        Nb_I(i, j) = fitness_sub_I(Nb_D(i, j), Nb_I(i, j), In
```

```
stallation_Time_I, Service_Level, Max_Operation_Time, Distance)

                        temp_ind_I = fitness_sub_I(Nb_D(i, j), temp_ind_I, In
stallation_Time_I, Service_Level, Max_Operation_Time, Distance)

                        If temp_ind_I.fitness_I > Nb_I(i, j).fitness_I Then
                            Nb_I(i, j) = temp_ind_I
                            Nb_D(i, j).fitness_D = temp_ind_I.fitness_I
                            k = 3
                        Else
                            k = k + 1
                        End If
                    Next
                End If
            Next
        Next

    End Sub


    'Create set of parents for crossover
    Function select_pair_I(ByVal Nb_I(,) As Symbiotic_I)

        Dim selection_set As Pair

        selection_set.p1 = roulette_I(Nb_I)
        selection_set.p2 = roulette_I(Nb_I)

        While selection_set.p1.loc_X = selection_set.p2.loc_X And selection_s
et.p1.loc_Y = selection_set.p2.loc_Y
            selection_set.p2 = roulette_I(Nb_I)
        End While

        Return selection_set

    End Function


    'Select good individual with probability
    Function roulette_I(ByVal Nb_I(,) As Symbiotic_I)

        Dim selected_loc As Location_Index
        Dim i, j As Integer
        Dim rand_value As Double
        Dim temp_cur_fitness As Double
        Dim temp_sum_fitness As Double

        temp_sum_fitness = 0.0

        For i = 0 To 2
            For j = 0 To 2
                temp_sum_fitness += Nb_I(i, j).fitness_I
            Next
        Next

        rand_value = Rnd()
        temp_cur_fitness = 0.0

        For i = 0 To 2
            For j = 0 To 2
                If rand_value > temp_cur_fitness And rand_value <= temp_cur_f
itness + (Nb_I(i, j).fitness_I / temp_sum_fitness) Then
                    selected_loc.loc_X = i
```

```
                    selected_loc.loc_Y = j
                End If
                temp_cur_fitness += (Nb_I(i, j).fitness_I / temp_sum_fitness)
            Next
        Next

        Return selected_loc
    End Function

    ''Select bad individual with probability
    Function roulette_inverse_I(ByVal Nb_I(,) As Symbiotic_I)

        Dim selected_loc As Location_Index

        Dim i, j As Integer
        Dim rand_value As Double
        Dim temp_cur_fitness As Double
        Dim temp_sum_fitness As Double

        temp_sum_fitness = 0.0

        For i = 0 To 2
            For j = 0 To 2
                temp_sum_fitness += 1 / Nb_I(i, j).fitness_I
            Next
        Next

        rand_value = Rnd()

        temp_cur_fitness = 0.0

        For i = 0 To 2
            For j = 0 To 2
                If rand_value > temp_cur_fitness And rand_value <= temp_cur_f
itness + ((1 / Nb_I(i, j).fitness_I) / temp_sum_fitness) Then
                    selected_loc.loc_X = i
                    selected_loc.loc_Y = j
                End If
                temp_cur_fitness += ((1 / Nb_I(i, j).fitness_I) / temp_sum_fi
tness)
            Next
        Next

        Return selected_loc

    End Function


    'Select worst individual in neighborhood
    Function select_worst_I(ByVal Nb_I(,) As Symbiotic_I)

        Dim selected_loc As Location_Index

        Dim i, j As Integer
        Dim temp_fitness As Double


        temp_fitness = 0.0

        selected_loc.loc_X = 0
        selected_loc.loc_Y = 0
        For i = 0 To 2
            For j = 0 To 2
```

```
                If Nb_I(i, j).fitness_I > temp_fitness Then
                    temp_fitness = Nb_I(i, j).fitness_I
                    selected_loc.loc_X = i
                    selected_loc.loc_Y = j
                End If
            Next
        Next

        Return selected_loc

    End Function

    'Crossover (2 steps)
    Function crossover_I(ByVal Nb_I1 As Symbiotic_I, ByVal Nb_I2 As Symbiotic
_I)

        Dim temp_child As Symbiotic_I

        temp_child = xover_hybrid_p1_I(Nb_I1)
        temp_child = xover_hybrid_p2_I(Nb_I2, temp_child)

        'Add different crossover functions below with "if ~ endif"

        Return temp_child

    End Function


    'Crossover with the parent 1 (step 1)
    Function xover_hybrid_p1_I(ByVal Nb_I1 As Symbiotic_I)

        Dim temp_child As Symbiotic_I

        Dim temp_gene(Nb_I1.gene_I.Length - 1) As Integer
        Dim num_vehicle_used As Integer
        Dim selected_vehicle As Integer
        Dim temp_count4child As Integer
        Dim temp_count4remain As Integer
        Dim one_cut_point As Integer

        Dim i As Integer

        temp_child = define_individual_I(Nb_I1.gene_I.Length)

        'Vehicle selection based crossover
        num_vehicle_used = Nb_I1.vehicle_I(Nb_I1.gene_I.Length - 1)

        selected_vehicle = CInt(Int((Rnd() * num_vehicle_used - 1)) + 1)

        temp_count4child = 0
        temp_count4remain = 0

        For i = 0 To Nb_I1.gene_I.Length - 1
            If Nb_I1.vehicle_I(i) = selected_vehicle Then
                temp_child.gene_I(temp_count4child) = Nb_I1.gene_I(i)
                temp_count4child += 1
            Else
                temp_gene(temp_count4remain) = Nb_I1.gene_I(i)
                temp_count4remain += 1
            End If
        Next

        'One cut point crossover
```

```
        one_cut_point = CInt(Int((Rnd() * (temp_count4remain - 2))) + 1)

        For i = 0 To one_cut_point
            temp_child.gene_I(temp_count4child) = temp_gene(i)
            temp_count4child += 1
        Next

        For i = temp_count4child To Nb_I1.gene_I.Length - 1
            temp_child.gene_I(i) = -1
        Next

        Return temp_child

    End Function


    'Crossover with the parent 2 (step 2)
    Function xover_hybrid_p2_I(ByVal Nb_I2 As Symbiotic_I, ByRef temp_child A
s Symbiotic_I)

        Dim i, j, k As Integer
        Dim plag As Boolean

        For i = 0 To Nb_I2.gene_I.Length - 1
            plag = False
            If temp_child.gene_I(i) = -1 Then
                k = 0
                While temp_child.gene_I(i) = -1
                    For j = 0 To i - 1
                        If Nb_I2.gene_I(k) = temp_child.gene_I(j) Then
                            plag = True
                        End If
                    Next

                    If plag Then
                        plag = False
                        k += 1
                    Else
                        temp_child.gene_I(i) = Nb_I2.gene_I(k)
                    End If
                End While
            End If
        Next

        Return temp_child

    End Function


    'Decide whether new offspring can enter neighborhood or not
    Function compare_I(ByRef Indv_I As Symbiotic_I, ByRef offspring As Symbio
tic_I, ByRef Indv_D As Symbiotic_D, ByVal Max_Operation_Time As Double, _
    ByVal Distance(,) As Double, ByVal Installation_Time_I As Double, ByVal S
ervice_Level As Double)

        offspring = fitness_sub_I(Indv_D, offspring, Installation_Time_I, Ser
vice_Level, Max_Operation_Time, Distance)

        If offspring.fitness_I > Indv_I.fitness_I Then
            Indv_I = copy_I2I(Indv_I, offspring)
            Indv_D.fitness_D = offspring.fitness_I

            Main.sum_entr_sol = Main.sum_entr_sol + 1
```

```
        End If

        Return Indv_I

    End Function


    'Mutation
    Function mutation_I(ByRef Indv_I As Symbiotic_I)

        Indv_I = mutation_exchange_I(Indv_I)

        'Add different mutation functions below with "if ~ endif"

        Return Indv_I

    End Function


    'Mutation (exchange mutation)
    Function mutation_exchange_I(ByRef Indv_I As Symbiotic_I)

        Dim rand_select_1 As Integer
        Dim rand_select_2 As Integer
        Dim temp_gene As Integer

        rand_select_1 = CInt(Int(Rnd() * (Indv_I.gene_I.Length - 1)))
        rand_select_2 = rand_select_1

        While (rand_select_2 = rand_select_1)
            rand_select_2 = CInt(Int(Rnd() * (Indv_I.gene_I.Length - 1)))
        End While

        temp_gene = Indv_I.gene_I(rand_select_1)
        Indv_I.gene_I(rand_select_1) = Indv_I.gene_I(rand_select_2)
        Indv_I.gene_I(rand_select_2) = temp_gene

        Return Indv_I

    End Function

End Module
```

Table C.9 The program code in Evoltuion_DI.vb.

```
Module Evolution_DI


    'Evolution of Nb_DI(main function)
    Sub evolve_DI(ByRef Nb_DI(,) As Endosymbiotic, ByVal Crossover_Rate As Si
ngle, _
    ByVal Mutation_Rate As Single, ByVal Capacity_Vehicle_D As Integer, ByVal
 Max_Operation_Time As Double, _
    ByVal Distance(,) As Double, ByVal Customer_Demand() As Integer, ByVal In
stallation_Time_I As Double, _
    ByVal Service_Level As Double)

        Dim temp_Nb_DI(2, 2) As Endosymbiotic
        Dim temp_ind_DI As Endosymbiotic

        Dim selection_set As Pair
        Dim offspring As Endosymbiotic
        Dim compete_indv As Location_Index
        Dim i, j, k As Integer

        Dim num_xover As Integer = 18

        'Duplicate Neighborhood DI
        For i = 0 To 2
            For j = 0 To 2
                temp_Nb_DI(i, j) = define_individual_DI(Nb_DI(0, 0).gene_D.Le
ngth, Nb_DI(0, 0).gene_I.Length)
                temp_Nb_DI(i, j) = copy_DI2DI(temp_Nb_DI(i, j), Nb_DI(i, j))
            Next
        Next

        'Crossover
        For i = 0 To num_xover
            If Rnd() < Crossover_Rate Then
                'Create parent set for crossover
                selection_set = select_pair_DI(Nb_DI)

                offspring = crossover_DI(Nb_DI(selection_set.p1.loc_X, select
ion_set.p1.loc_Y), Nb_DI(selection_set.p2.loc_X, _
                selection_set.p2.loc_Y))

                'Select a neighhor to compete with offspring randomly
                compete_indv = roulette_inverse_DI(Nb_DI)

                compare_DI(Nb_DI(compete_indv.loc_X, compete_indv.loc_Y), off
spring, Capacity_Vehicle_D, Max_Operation_Time, _
                Distance, Customer_Demand, Installation_Time_I, Service_Leve
l)
            End If
        Next

        'Mutation
        For i = 0 To 2
            For j = 0 To 2
                If Rnd() < Mutation_Rate Then
                    k = 0
                    For k = 0 To 3
                        temp_ind_DI = Nb_DI(i, j)
                        temp_ind_DI = mutation_DI(temp_ind_DI)

                        temp_ind_DI = fitness_etr_D(temp_ind_DI, Capacity_Veh
```

```
icle_D, Max_Operation_Time, Customer_Demand, Distance)
                        temp_ind_DI = fitness_etr_I(temp_ind_DI, Installation
_Time_I, Service_Level, Max_Operation_Time, Distance)

                        If temp_ind_DI.fitness_DI > Nb_DI(i, j).fitness_DI Th
en
                            Nb_DI(i, j) = temp_ind_DI
                            k = 3
                        Else
                            k = k + 1
                        End If
                    Next
                End If
            Next
        Next

    End Sub


    'Create set of parent for crossover
    Function select_pair_DI(ByVal Nb_DI(,) As Endosymbiotic)

        Dim selection_set As Pair

        selection_set.p1 = roulette_DI(Nb_DI)
        selection_set.p2 = roulette_DI(Nb_DI)

        While selection_set.p1.loc_X = selection_set.p2.loc_X And selection_s
et.p1.loc_Y = selection_set.p2.loc_Y
            selection_set.p2 = roulette_DI(Nb_DI)
        End While

        Return selection_set

    End Function


    'Select good individual with probability
    Function roulette_DI(ByVal Nb_DI(,) As Endosymbiotic)

        Dim selected_loc As Location_Index
        Dim i, j As Integer
        Dim rand_value As Double
        Dim temp_cur_fitness As Double
        Dim temp_sum_fitness As Double

        temp_sum_fitness = 0.0

        For i = 0 To 2
            For j = 0 To 2
                temp_sum_fitness += Nb_DI(i, j).fitness_DI
            Next
        Next

        rand_value = Rnd()
        temp_cur_fitness = 0.0

        For i = 0 To 2
            For j = 0 To 2
                If rand_value > temp_cur_fitness And rand_value <= temp_cur_f
itness + (Nb_DI(i, j).fitness_DI / temp_sum_fitness) Then
                    selected_loc.loc_X = i
                    selected_loc.loc_Y = j
```

```
                End If
                temp_cur_fitness += (Nb_DI(i, j).fitness_DI / temp_sum_fitnes
s)
            Next
        Next

        Return selected_loc

    End Function


    'Select bad individual with probability
    Function roulette_inverse_DI(ByVal Nb_DI(,) As Endosymbiotic)

        Dim selected_loc As Location_Index

        Dim i, j As Integer
        Dim rand_value As Double
        Dim temp_cur_fitness As Double
        Dim temp_sum_fitness As Double

        temp_sum_fitness = 0.0

        For i = 0 To 2
            For j = 0 To 2
                temp_sum_fitness += 1 / Nb_DI(i, j).fitness_DI
            Next
        Next

        rand_value = Rnd()
        temp_cur_fitness = 0.0

        For i = 0 To 2
            For j = 0 To 2
                If rand_value > temp_cur_fitness And rand_value <= temp_cur_f
itness + ((1 / Nb_DI(i, j).fitness_DI) / temp_sum_fitness) Then
                    selected_loc.loc_X = i
                    selected_loc.loc_Y = j
                End If
                temp_cur_fitness += ((1 / Nb_DI(i, j).fitness_DI) / temp_sum_
fitness)
            Next
        Next

        Return selected_loc

    End Function

    'Select worst individual in neighborhood
    Function select_worst_DI(ByVal Nb_DI(,) As Endosymbiotic)

        Dim selected_loc As Location_Index

        Dim i, j As Integer
        Dim temp_fitness As Double

        temp_fitness = 0.0

        selected_loc.loc_X = 0
        selected_loc.loc_Y = 0
        For i = 0 To 2
            For j = 0 To 2
                If Nb_DI(i, j).fitness_DI > temp_fitness Then
```

```
                    temp_fitness = Nb_DI(i, j).fitness_DI
                    selected_loc.loc_X = i
                    selected_loc.loc_Y = j
                End If
            Next
        Next

        Return selected_loc

    End Function


    'Crossover (2 steps)
    Function crossover_DI(ByVal Nb_DI1 As Endosymbiotic , ByVal Nb_DI2 As End
osymbiotic)

        Dim temp_child As Endosymbiotic

        temp_child = xover_hybrid_p1_DI(Nb_DI1)
        temp_child = xover_hybrid_p2_DI(Nb_DI2, temp_child)

        'Add different crossover functions below with "if ~ endif"

        Return temp_child

    End Function


    'Crossover with the parent 1 (step 1)
    Function xover_hybrid_p1_DI(ByVal Nb_DI1 As Endosymbiotic)

        Dim temp_child As Endosymbiotic

        Dim temp_gene_D(Nb_DI1.gene_D.Length - 1) As Integer
        Dim temp_gene_I(Nb_DI1.gene_I.Length - 1) As Integer

        Dim num_vehicle_used As Integer
        Dim selected_vehicle As Integer

        Dim temp_count4child As Integer
        Dim temp_count4remain As Integer
        Dim one_cut_point As Integer

        Dim i As Integer

        temp_child = define_individual_DI(Nb_DI1.gene_D.Length, Nb_DI1.gene_
I.Length)

        'Vehicle selection based crossover for part D
        num_vehicle_used = Nb_DI1.vehicle_D(Nb_DI1.gene_D.Length - 1)

        selected_vehicle = CInt(Int((Rnd() * num_vehicle_used - 1)) + 1)

        temp_count4child = 0
        temp_count4remain = 0

        For i = 0 To Nb_DI1.gene_D.Length - 1
            If Nb_DI1.vehicle_D(i) = selected_vehicle Then
                temp_child.gene_D(temp_count4child) = Nb_DI1.gene_D(i)
                temp_count4child += 1
            Else
                temp_gene_D(temp_count4remain) = Nb_DI1.gene_D(i)
                temp_count4remain += 1
```

```
            End If
        Next

        'One cut point crossover for part D
        one_cut_point = CInt(Int((Rnd() * (temp_count4remain - 2))) + 1)

        For i = 0 To one_cut_point
            temp_child.gene_D(temp_count4child) = temp_gene_D(i)
            temp_count4child += 1
        Next

        For i = temp_count4child To Nb_DI1.gene_D.Length - 1
            temp_child.gene_D(i) = -1
        Next

        'Vehicle selection based crossover for part I
        num_vehicle_used = Nb_DI1.vehicle_I(Nb_DI1.gene_I.Length - 1)

        selected_vehicle = CInt(Int((Rnd() * num_vehicle_used - 1)) + 1)

        temp_count4child = 0
        temp_count4remain = 0

        For i = 0 To Nb_DI1.gene_I.Length - 1
            If Nb_DI1.vehicle_I(i) = selected_vehicle Then
                temp_child.gene_I(temp_count4child) = Nb_DI1.gene_I(i)
                temp_count4child += 1
            Else
                temp_gene_I(temp_count4remain) = Nb_DI1.gene_I(i)
                temp_count4remain += 1
            End If
        Next

        'One cut point crossover for part I
        one_cut_point = CInt(Int((Rnd() * (temp_count4remain - 2))) + 1)

        For i = 0 To one_cut_point
            temp_child.gene_I(temp_count4child) = temp_gene_I(i)
            temp_count4child += 1
        Next

        For i = temp_count4child To Nb_DI1.gene_I.Length - 1
            temp_child.gene_I(i) = -1
        Next

        Return temp_child

    End Function


    'Crossover with the parent 2 (step 2)
    Function xover_hybrid_p2_DI(ByVal Nb_DI2 As Endosymbiotic, ByRef temp_chi
ld As Endosymbiotic)

        Dim i, j, k As Integer
        Dim plag As Boolean

        'For part D
        For i = 0 To Nb_DI2.gene_D.Length - 1
            plag = False
            If temp_child.gene_D(i) = -1 Then
                k = 0
                While temp_child.gene_D(i) = -1
```

```
                    For j = 0 To i - 1
                        If Nb_DI2.gene_D(k) = temp_child.gene_D(j) Then
                            plag = True
                        End If
                    Next

                    If plag Then
                        plag = False
                        k += 1
                    Else
                        temp_child.gene_D(i) = Nb_DI2.gene_D(k)
                    End If
                End While
            End If
        Next

        'For part I
        For i = 0 To Nb_DI2.gene_I.Length - 1
            plag = False
            If temp_child.gene_I(i) = -1 Then
                k = 0
                While temp_child.gene_I(i) = -1
                    For j = 0 To i - 1
                        If Nb_DI2.gene_I(k) = temp_child.gene_I(j) Then
                            plag = True
                        End If
                    Next

                    If plag Then
                        plag = False
                        k += 1
                    Else
                        temp_child.gene_I(i) = Nb_DI2.gene_I(k)
                    End If
                End While
            End If
        Next


        Return temp_child

    End Function


    'Decide whether new offspring can enter neighborhood or not
    Function compare_DI(ByRef Indv_DI As Endosymbiotic, ByRef offspring As En
dosymbiotic, ByVal Capacity_Vehicle_D As Integer, ByVal Max_Operation_Time As
 Double, _
    ByVal Distance(,) As Double, ByVal Customer_Demand() As Integer, ByVal In
stallation_Time_I As Double, ByVal Service_Level As Double)

        offspring = fitness_etr_D(offspring, Capacity_Vehicle_D, Max_Operatio
n_Time, Customer_Demand, Distance)
        offspring = fitness_etr_I(offspring, Installation_Time_I, Service_Lev
el, Max_Operation_Time, Distance)

        If offspring.fitness_DI > Indv_DI.fitness_DI Then
            Indv_DI = offspring

            Main.sum_entr_sol = Main.sum_entr_sol + 1
        End If

        Return Indv_DI
```

```
    End Function


    'Mutation
    Function mutation_DI(ByRef Indv_DI As Endosymbiotic)

        Indv_DI = mutation_exchange_DI(Indv_DI)

        'Add different mutation functions below with "if ~ endif"

        Return Indv_DI
    End Function


    'Mutation (exchange mutation)
    Function mutation_exchange_DI(ByRef Indv_DI As Endosymbiotic)

        Dim rand_select_1 As Integer
        Dim rand_select_2 As Integer
        Dim temp_gene As Integer

        'Mutaion for part D
        rand_select_1 = CInt(Int(Rnd() * (Indv_DI.gene_D.Length - 1)))
        rand_select_2 = rand_select_1

        While (rand_select_2 = rand_select_1)
            rand_select_2 = CInt(Int(Rnd() * (Indv_DI.gene_D.Length - 1)))
        End While

        temp_gene = Indv_DI.gene_D(rand_select_1)
        Indv_DI.gene_D(rand_select_1) = Indv_DI.gene_D(rand_select_2)
        Indv_DI.gene_D(rand_select_2) = temp_gene

        'Mutaion for part I
        rand_select_1 = CInt(Int(Rnd() * (Indv_DI.gene_I.Length - 1)))
        rand_select_2 = rand_select_1

        While (rand_select_2 = rand_select_1)
            rand_select_2 = CInt(Int(Rnd() * (Indv_DI.gene_I.Length - 1)))
        End While

        temp_gene = Indv_DI.gene_I(rand_select_1)
        Indv_DI.gene_I(rand_select_1) = Indv_DI.gene_I(rand_select_2)
        Indv_DI.gene_I(rand_select_2) = temp_gene

        Return Indv_DI

    End Function

End Module
```

Table C.10 The program code in Dt_Input.vb.

```
Module Dt_Input

    'Load problem variables from input file
    Function read_data(ByVal input_file As String, ByVal Dmd() As Integer, By
Val Dst(,) As Double, ByVal N_DCustomer As Integer, ByVal N_ICustomer As Inte
ger)

        Dim i, j As Integer
        Dim check_file As Boolean = True
        Dim temp_Customer, temp_Check_Installation As Integer
        Dim temp_Installation_time, temp_Timewindow_start, temp_Timewindow_en
d As Integer
        Dim temp_Distance As Double

        Try
            FileOpen(1, input_file, OpenMode.Input)
        Catch ex As Exception
            MsgBox("Wrong file name")
            check_file = False
            FileClose(1)
            GoTo END_FUNC
        End Try

        'read customer demand
        For i = 0 To N_DCustomer - 1
            Input(1, temp_Customer)

            If i <> temp_Customer Then
                MsgBox("Customer Index Incorrect!")
            End If

            Input(1, Dmd(i))
            Input(1, temp_Check_Installation)

            If (i > N_ICustomer - 1 And temp_Check_Installation = 1) Or (i <=
 N_ICustomer - 1 And temp_Check_Installation = 0) Then
                MsgBox("Installation Customer Index Mismatch!")
            End If

            'no need 3 value in EEA (only for HGA)
            Input(1, temp_Installation_time)
            Input(1, temp_Timewindow_start)
            Input(1, temp_Timewindow_end)
        Next


        For i = 0 To N_DCustomer
            Dst(i, i) = 0
            For j = i + 1 To N_DCustomer
                Input(1, temp_Distance)
                Dst(i, j) = temp_Distance
                Dst(j, i) = temp_Distance
            Next
        Next

        FileClose(1)

END_FUNC:
        Return check_file

    End Function
```

```
End Module
```

Table C.11 The program code in Dt_Output.vb.

```
Module Dt_Output

    'Initialize output file
    Function Outfile_setup(ByVal current_trial As Integer, ByVal Num_Customer
_D As Integer, ByVal Num_Customer_I As Integer, ByVal Pop_size As Integer)

        Dim temp_file_name As String

        temp_file_name = "OT-" + CStr(Num_Customer_D) + "-" + CStr(Num_Custom
er_I) + "-" + CStr(Pop_size) + "-" + CStr(current_trial) + "-"

        'Delete old file
        If My.Computer.FileSystem.FileExists(temp_file_name + "Fnl.txt") Then
            My.Computer.FileSystem.DeleteFile(temp_file_name + "Fnl.txt")
        End If
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Fina
l solution of this problem " & vbCrLf & vbCrLf, True)


        If My.Computer.FileSystem.FileExists(temp_file_name + "Upd.txt") Then
            My.Computer.FileSystem.DeleteFile(temp_file_name + "Upd.txt")
        End If
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Upd.txt", "Fitn
ess value at the improved generation " & vbCrLf & vbCrLf, True)


        If My.Computer.FileSystem.FileExists(temp_file_name + "Pop-D.txt") Th
en
            My.Computer.FileSystem.DeleteFile(temp_file_name + "Pop-D.txt")
        End If
        If My.Computer.FileSystem.FileExists(temp_file_name + "Pop-I.txt") Th
en
            My.Computer.FileSystem.DeleteFile(temp_file_name + "Pop-I.txt")
        End If
        If My.Computer.FileSystem.FileExists(temp_file_name + "Pop-DI.txt") T
hen
            My.Computer.FileSystem.DeleteFile(temp_file_name + "Pop-DI.txt")
        End If



        Return temp_file_name

    End Function


    'Record final results on output file
    Sub final_report(ByVal best_solution As Endosymbiotic, ByVal elapsed_time
 As TimeSpan, ByVal temp_file_name As String)

        Dim i As Integer
        Dim temp_result As String
        Dim temp_wait As Double

        temp_result = ""


        'Report for delivery vehicles
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Deli
very Vehicles" & vbCrLf & vbCrLf, True)
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Calc
```

```
uation time : " & CStr(elapsed_time.TotalSeconds.ToString("0.00")) & vbCrLf &
 vbCrLf, True)
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Gene
 for Delivery Vehicles" & vbCrLf, True)

        'Customer visiting order for delivery vehicles
        For i = 0 To best_solution.gene_D.Length - 1
            temp_result = temp_result + CStr(best_solution.gene_D(i)) + " "
        Next

        'Assigned delivery vehicles
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", temp_
result & vbCrLf, True)
        temp_result = ""
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Assi
gned Delivery Vehicles" & vbCrLf, True)

        For i = 0 To best_solution.gene_D.Length - 1
            temp_result = temp_result + CStr(best_solution.vehicle_D(i)) + "
"
        Next

        'Arrival times of delivery vehicles
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", temp_
result & vbCrLf, True)
        temp_result = ""
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Arri
val Delivery Vehicles" & vbCrLf, True)

        For i = 0 To best_solution.gene_D.Length - 1
            temp_result = temp_result + CStr(best_solution.arrival_time_D(i))
 + " "
        Next

        'Traveling times of delivery vehicles
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", temp_
result & vbCrLf, True)
        temp_result = ""
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Trav
eling time of Delivery vehicles" & vbCrLf, True)
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", CStr
(best_solution.travel_time_D) & vbCrLf & vbCrLf & vbCrLf & vbCrLf, True)
        temp_result = ""


        'Report for installation vehicles
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Inst
allation Vehicles" & vbCrLf & vbCrLf, True)
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Gene
 for Installation Vehicles" & vbCrLf, True)

        For i = 0 To best_solution.gene_I.Length - 1
            temp_result = temp_result + CStr(best_solution.gene_I(i)) + " "
        Next

        'Assigned installation vehicles
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", temp_
result & vbCrLf, True)
        temp_result = ""
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Assi
gned Installation Vehicles" & vbCrLf, True)

        For i = 0 To best_solution.gene_I.Length - 1
```

```
            temp_result = temp_result + CStr(best_solution.vehicle_I(i)) + "
"
        Next

        'Arrival times of installation vehicles
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", temp_
result & vbCrLf, True)
        temp_result = ""
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Arri
val Installation Vehicles" & vbCrLf, True)

        For i = 0 To best_solution.gene_I.Length - 1
            temp_result = temp_result + CStr(best_solution.arrival_time_I(i))
 + " "
        Next

        'Waitign times of installation vehicles
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", temp_
result & vbCrLf, True)
        temp_result = ""
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Wait
ing time Installation Vehicles" & vbCrLf, True)
        temp_wait = 0.0

        For i = 0 To best_solution.gene_I.Length - 1
            temp_wait += best_solution.wait_time_I(i)
            temp_result = temp_result + CStr(best_solution.wait_time_I(i)) +
" "
        Next

        'Traveling times of installation vehicles
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", temp_
result & vbCrLf, True)
        temp_result = ""
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Trav
elling time of Installation vehicles" & vbCrLf, True)
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", CStr
(best_solution.travel_time_I - (10 * best_solution.gene_I.Length) - temp_wai
t) & vbCrLf & vbCrLf & vbCrLf, True)
        temp_result = ""

        'Traveling time of all vehicles
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Tota
l travelling time" & vbCrLf, True)
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", CStr
(best_solution.travel_time_D + best_solution.travel_time_I - (10 * best_solut
ion.gene_I.Length) - temp_wait) & vbCrLf & vbCrLf, True)
        'Fitness values of best solutions
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", "Fitn
ess value" & vbCrLf, True)
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", CStr
(best_solution.fitness_DI), True)
        'Number of generated offspring so far
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", vbCrL
f & vbCrLf & "Number of entered offspring : ", True)
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", CStr
(Main.sum_entr_sol), True)
        'Number of last generation
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", vbCrL
f & "Last generation : ", True)
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Fnl.txt", CStr
(Main.stop_gen), True)
```

```
    End Sub


    'Report update record for any improvement
    Function generation_record(ByVal current_generation As Integer, ByVal bes
t_solution As Endosymbiotic, ByRef temp_best_travel As Double, _
    ByRef temp_file_name As String, ByVal start_time As Date, ByRef end_time
As Date)

        Dim sum_wait As Double
        Dim current_travel As Double
        Dim i As Integer
        Dim elapsed_time As TimeSpan
        Dim sum_installation_time As Double

        sum_installation_time = 10 * best_solution.gene_I.Length
        sum_wait = 0.0

        For i = 0 To best_solution.gene_I.Length - 1
            sum_wait += best_solution.wait_time_I(i)
        Next

        current_travel = best_solution.travel_time_D + best_solution.travel_t
ime_I - sum_installation_time - sum_wait

        'Check the improvement of best solution and report
        If current_travel < temp_best_travel Then
            'Update new best solution
            temp_best_travel = current_travel
            'Record ending time
            end_time = Now
            elapsed_time = end_time.Subtract(start_time)
            'Record on File
            update_log(current_generation, temp_best_travel, temp_file_name,
elapsed_time)
        End If

        Return temp_best_travel

    End Function


    'Record traveling times at certain generation
    Sub best_record(ByVal current_generation As Integer, ByVal current_travel
 As Double, ByRef temp_file_name As String)

        My.Computer.FileSystem.WriteAllText(temp_file_name + "Bst.txt", CStr
(current_generation) & vbTab & CStr(current_travel) _
        & vbCrLf, True)

    End Sub


    'Record best traveling times at any improvement
    Sub update_log(ByVal current_generation As Integer, ByVal temp_best_trave
l As Double, ByRef temp_file_name As String, ByVal elapsed_time As TimeSpan)

        My.Computer.FileSystem.WriteAllText(temp_file_name + "Upd.txt", CStr
(current_generation) & vbTab & CStr(Main.sum_entr_sol) & vbTab & CStr(temp_be
st_travel) _
        & vbTab & CStr(elapsed_time.TotalSeconds.ToString("0.00")) & vbCrLf,
True)
```

```
    End Sub


    'Print POP-D on file
    Sub file_population_sub_D(ByVal POP_D(,) As Symbiotic_D, ByVal pop_size A
s Integer, _
    ByVal current_generation As Integer, ByVal loc_Nb As Location_Index, ByRe
f temp_file_name As String)

        Dim i, j As Integer

        My.Computer.FileSystem.WriteAllText(temp_file_name + "Pop-D.txt", "Ge
neration: " & CStr(current_generation) & ": " & _
        CStr(loc_Nb.loc_X) & " , " & CStr(loc_Nb.loc_Y) & vbCrLf, True)

        For j = 0 To pop_size - 1
            For i = 0 To pop_size - 1
                My.Computer.FileSystem.WriteAllText(temp_file_name + "Pop-D.t
xt", Format(POP_D(i, j).fitness_D, "0.00000000") & vbTab, True)
            Next
            My.Computer.FileSystem.WriteAllText(temp_file_name + "Pop-D.txt",
 vbCrLf, True)
        Next

        My.Computer.FileSystem.WriteAllText(temp_file_name + "Pop-D.txt", vbC
rLf & vbCrLf, True)

    End Sub


    'Print POP-I on file
    Sub file_population_sub_I(ByVal POP_I(,) As Symbiotic_I, ByVal pop_size A
s Integer, _
    ByVal current_generation As Integer, ByVal loc_Nb As Location_Index, ByRe
f temp_file_name As String)

        Dim i, j As Integer

        My.Computer.FileSystem.WriteAllText(temp_file_name + "Pop-I.txt", "Ge
neration: " & CStr(current_generation) & ": " & _
        CStr(loc_Nb.loc_X) & " , " & CStr(loc_Nb.loc_Y) & vbCrLf, True)

        For j = 0 To pop_size - 1
            For i = 0 To pop_size - 1
                My.Computer.FileSystem.WriteAllText(temp_file_name + "Pop-I.t
xt", Format(POP_I(i, j).fitness_I, "0.00000000") & vbTab, True)
            Next
            My.Computer.FileSystem.WriteAllText(temp_file_name + "Pop-I.txt",
 vbCrLf, True)
        Next

        My.Computer.FileSystem.WriteAllText(temp_file_name + "Pop-I.txt", vbC
rLf & vbCrLf, True)

    End Sub


    'Print POP-DI on file
    Sub file_population_etr(ByVal POP_DI(,) As Endosymbiotic, ByVal pop_size
As Integer, ByVal current_generation As Integer, _
    ByVal loc_Nb As Location_Index, ByRef temp_file_name As String)

        Dim i, j As Integer
```

```
        My.Computer.FileSystem.WriteAllText(temp_file_name + "Pop-DI.txt", "G
eneration: " & CStr(current_generation) & ": " & _
        CStr(loc_Nb.loc_X) & " , " & CStr(loc_Nb.loc_Y) & vbCrLf, True)

        For j = 0 To pop_size - 1
            For i = 0 To pop_size - 1
                My.Computer.FileSystem.WriteAllText(temp_file_name + "Pop-DI.
txt", Format(POP_DI(i, j).fitness_DI, "0.00000000") & vbTab, True)
            Next
            My.Computer.FileSystem.WriteAllText(temp_file_name + "Pop-DI.txt
", vbCrLf, True)
        Next

        My.Computer.FileSystem.WriteAllText(temp_file_name + "Pop-DI.txt", vb
CrLf & vbCrLf, True)

    End Sub


    'Check population printing
    Sub file_population_check(ByVal POP_DI(,) As Endosymbiotic, ByVal POP_D
(,) As Symbiotic_D, ByVal POP_I(,) As Symbiotic_I, ByVal pop_size As Integer,
 ByVal current_generation As Integer, _
    ByVal loc_Nb As Location_Index, ByVal check_pop As Integer, ByRef temp_fi
le_name As String)

        If EEA.chk_popchk.Checked = True Then
            If (current_generation Mod check_pop) = 0 Then
                If EEA.chk_POPD.Checked = True Then
                    file_population_sub_D(POP_D, pop_size, current_generatio
n, loc_Nb, temp_file_name)
                End If

                If EEA.chk_POPI.Checked = True Then
                    file_population_sub_I(POP_I, pop_size, current_generatio
n, loc_Nb, temp_file_name)
                End If

                If EEA.chk_POPDI.Checked = True Then
                    file_population_etr(POP_DI, pop_size, current_generation,
 loc_Nb, temp_file_name)
                End If
            End If
        End If

    End Sub

End Module
```