ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Bachelor Thesis [BSc^2 Econometrics and Economics]

Improving an evolutionary algorithm for constructing waiting strategies applied to the dynamic vehicle routing problem with one additional customer

Abstract

The problem analyzed in this thesis is a specific case of the dynamic vehicle routing problem (VRP), where one additional customer request appears over time. If the new customer can be inserted in the already existing routes within the maximum time the request is accepted, and otherwise rejected. Waiting strategies are defined as time allocations to customers in a route. The allocated time is used for waiting before continuing the planned route. These strategies are used to optimize the probability of inserting one additional customer. The dynamic VRP is a common real-world problem, thus the results are of great importance if the insertion probability could be significantly increased. First, six simple waiting strategies are introduced and compared to two simple evolutionary algorithms (EAs). Then, two characteristics of the EA are further investigated: the fitness evaluation and the selection method. Primarily, different numbers of additional customers used for fitness evaluation are considered. Moreover, the performance of one dynamic and five static selection methods are studied. The first results indicate that applying a simple waiting strategy can already increase the insertion probability of one additional customer. Moreover, the performance of the initial EA is significantly improved by increasing the test set of customers used for fitness evaluation and changing the selection method. Finally, it is concluded that the improved algorithm performs slightly better than the best simple waiting strategy found.

Name student: Eva Beekmans

Student ID number: 449754

Supervisor: Hoogendoorn, Y.N.

Second assessor: Bouman, P.C.

July 1, 2021

Contents

1 Introduction

The Vehicle Routing Problem (VRP) is one of the most studied models within the Operations Research (Toth and Vigo, 2002). The origin of the problem dates back to 1959, where Dantzig and Ramser (1959) considered the delivery of gasoline from one location to multiple service stations while minimizing the distance traveled. Since then, many different variations of the VRP have emerged. As global integration increased, the VRP became more prevalent (Psaraftis, 1995). Due to an open market structure of the world economy, the market has become more competitive. Hence, efficient logistical services can lead to a reduction in transportation costs and increased customer satisfaction, which could result in the competitive advantage a company needs to survive. Consequently, even though the VRP is already elaborately discussed its relevance did not decline.

One of the distinctions that exist within the VRP is the difference between static and dynamic problems. Static indicates that all information is known before executing the routes, while in the dynamic variations (a part of the) information becomes available over time. Due to advances in information and communication technologies since the 90s, the dynamic VRP has become more popular (Braekers et al., 2016). For example, the Global Positioning System (GPS) and the Geographical Information System (GIS) create the opportunity to receive real-time information, which offers the possibility to take this information into account while looking for optimal solutions (Mitrović-Minić et al., 2004). Moreover, as possessing real-time information becomes more common, learning how to use this knowledge efficiently could lead to a reduction in operational costs, a higher service level, and less environmental damage.

This thesis considers a scenario where dynamic information refers to one additional customer that needs to be visited by vehicles executing already existing routes. The original planning consists of multiple vehicles, each performing their tour, starting from and ending at the depot. The additional customer will have a random location within the original service region and a random time within the entire planning horizon. The routes of all vehicles have a maximum time constraint. If the new customer cannot be inserted without exceeding this time constraint, the request will be denied. Branke and Mattfeld (2005) introduced the problem discussed and emphasized the relevance of anticipation and flexibility within the routing schedule. Hence this research suggested the use of waiting strategies for the dynamic VRP. This approach lets vehicles wait at certain locations during the scheduled route, such that the customers that arrive over time can be integrated efficiently. Hence, the central research question of this thesis is as follows:

What is the best waiting strategy for maximizing the probability of inserting one additional customer in an existing route schedule?

As the research question focuses on optimizing the probability of inserting the customer, this thesis focuses on providing customer satisfaction in contrast with the common objective of the VRP of minimizing costs (Pillac et al., 2013). First, the waiting strategies and evolutionary algorithms (EAs) discussed in Branke et al. (2005) are implemented. Then, this thesis focuses on optimizing the EAs by varying the fitness evaluation and the selection methods used. Specifically, the test set of customers used in the fitness evaluation is increased. Furthermore, two new selection methods are introduced and these are combined with three well-known methods in a dynamic decision

process as a selection method. All different approaches will be compared by a simulation. Finally, the last comparison is made between the best performing strategies found, in order to conclude which waiting strategy results in the highest probability of inserting one additional customer.

This research is scientifically relevant since real-time information becomes more prevalent which raises the question of how to efficiently deal with this additional knowledge. Moreover, this research has a considerable societal impact as many logistic companies deal with the arrival of new customers while driving. For example, the many pick-up and delivery companies that increased enormously during the COVID-19 pandemic (Escudero et al., 2020) need to deal with this issue. Moreover, this research contributes scientifically by introducing new selection methods for the EA and exploring the use of a dynamic decision process as a selection method.

The rest of this thesis is structured as follows. First, a review of relevant literature related to the problem is discussed in Section 2. The focus will be on the classifications of the VRP and the development of the EAs over time. Then, Section 3 introduces the specifics of the problem and the mathematical formulations. This is followed by the methodology used in this thesis and a presentation and discussion of the results found, in Sections 4 and 5 respectively. Lastly, a conclusion is formulated in Section 6 and future research suggestions are presented in Section 7.

2 Literature review

A division is made in the summary of the relevant literature. Primarily, there will be a focus on the classifications within the general VRP. Then some practical applications and solution approaches for the dynamic VRP are exhibited. Finally, a section is dedicated to the advancements of the EA in optimization problems.

2.1 The vehicle routing problem

As mentioned before the VRP is extensively studied over multiple decades. As a consequence, many variants of the problem are formulated and reviewed. To illustrate, Zhang et al. (2021) distinct nine classifications based on different extensions and applications in the literature. Additional extensive surveys about the general problem can be found in Psaraftis (1995), Bertsimas and Simchi-Levi (1996), Pillac et al. (2013), and Braekers et al. (2016).

The most prominent characteristics of the problem concern the data. The largest distinctions can be made on the quality and the evolution of the information (Pillac et al., 2013). The quality reflects the uncertainty of the data in the problem, while the evolution represents the timing of the availability of the data. The quality is called deterministic when the exact numbers are known and stochastic when the values depend on a (known) distribution. Furthermore, the problem is called static when all information is known at the beginning of the problem, and dynamic when new information reveals itself over time within the problem (Mitrović-Minić et al., 2004). The dynamic VRP is also referred to as the online VRP or the real-time VRP (Braekers et al., 2016).

With the current quality of GPS applications, real-time information about travel times can be communicated at each instance nowadays. Hence, it is possible to consider varying travel times

within a VRP. This is an example of the real-time traffic information-based VRP, RTVRP (Zhang et al., 2021). However, most studied is the variant where information about customers is dynamic (Braekers et al., 2016). For instance, the demand of the customer could be revealed upon arrival or the customer presents itself after an initial tour is constructed (and partially executed). These cases are called the dynamic demand-based VRP, DDVRP (Zhang et al., 2021). In this scenario, the customers known at the beginning of the problem can be referred to as advance requests and the dynamic customers as immediate requests (Zeimpekis et al., 2007).

The VRP is known to be NP-hard, meaning that it can be difficult to find an optimal schedule for all instances of the problem (Laporte and Nobert, 1987). Moreover, real-life applications of the problem usually consider large and complex situations. Therefore, heuristics are often introduced for solving these types of problems. The typical heuristic methods can be classified into three groups according to Zhang et al. (2021): construction, improvement, and sub-heuristics. Construction heuristics extend a current solution in each iteration until a result is found, whereas improvement heuristics start with an initial solution and iteratively search for a solution with a better performance. An example of a construction heuristic may be the two-stage approach or the nearest neighbor method. For improving an initial construction λ-opt algorithms are well-known (Jünger et al., 1995). Lastly, tabu search, neural networks, and other algorithms belong to the sub-heuristics class. This last class represents algorithms that search for a global optimum in a certain search space.

As might be expected, solution approaches for the static and the dynamic VRP differ. For a static, deterministic problem a fixed set of routes might provide the optimal solution. However, introducing an algorithm that solves the problem as best as possible based on the known information is often the solution for a dynamic problem (Berbeglia et al., 2010). An important thing to note is that for the general VRP the most common objective is to minimize the transportation time or costs. However, for the dynamic problem, the objective depends more on the application and can, for example, relate to the service level or customer satisfaction (Pillac et al., 2013).

2.2 The dynamic VRP

Within the dynamic VRP, multiple variants exist. The problem that appears first in literature is the dial-a-ride problem (DARP) considering one vehicle (Wilson and Colvin, 1977). The DARP considers the case of transporting people from an origin to a destination. The problem is often discussed in the context of transporting disabled or elderly people by taxi. A similar variant of the problem is the dial-a-flight problem (DAFP), which focuses on air transport (Pillac et al., 2013).

Gendreau and Potvin (1998) define distinctions in the dynamic VRP based on the range of the area of requests and the relevance of routing. The range of the area of requests can either be local, in the DARP, or global, in the DAFP. Moreover, in the application of emergency services one truck is dedicated to a single customer. Hence, the aim of the problem is the allocation of vehicles rather than the routing of vehicles. This characteristic refers to the relevance of routing to the problem. Local routing issues are further categorized by the number of pick-up and delivery locations. When commodities need to be transferred from multiple locations to multiple destinations, the problem is

called many-to-many. However, when commodities need to be transferred from or to a single depot it is classified as a one-to-many or many-to-one problem respectively. Furthermore, a distinction can be made between a capacitated VRP (CVRP) or an uncapacitated VRP. In the CVRP the vehicles are capable of transferring a limited number of commodities, while this restriction does not apply to the uncapacitated instance.

As mentioned before, the solution approaches for solving a dynamic VRP differ from the approaches suitable for the static VRP. Ichoua et al. (2000) categorizes the solution methods in three classes: adaptation of static methods, stochastic methods, and metaheuristics. The adaptation of static methods is generally performed using a rolling horizon. Berbeglia et al. (2010) go into more depth and acknowledge a distinction between updating a static solution or re-optimizing the solution when new information becomes available. A drawback of re-optimization is the computation time, which becomes more critical when the arrival of additional information happens frequently. On the other hand, updating the current solution is easy to implement, but the results are short-sighted. Therefore, most research focuses on updating the initial solution and then perform local search methods.

A specific example of first updating the initial solution and then performing a local search can be found in Mitrović-Minić et al. (2004). Mitrović-Minić et al. (2004) studied the pick-up and delivery problem with time windows (PDPTW). This problem is characterized by a fleet of vehicles that serves pick-up and delivery demands, within a specific time window, while minimizing the total travel distance. A double horizon framework is implemented, where the short-run aims to find the cheapest insertion for the immediate request, while the long-run also considers the flexibility of the solution. Mitrović-Minić et al. (2004) use the cheapest insertion method and perform a tabu search thereafter. The results of the paper concluded that the double horizon leads to lower routing costs compared to the standard rolling horizon.

Branke and Mattfeld (2005) examine a dynamic job-shop problem, where several jobs need to be processed by multiple machines. The jobs occur dynamically over time. Branke and Mattfeld (2005) introduce a second objective, which represents the flexibility of the solution. Many heuristics are implemented, of which one is an EA. The outcomes show that anticipation of future changes results in more flexible and better solutions.

Anticipation and flexibility can be implemented through waiting strategies. A waiting strategy allocates time to each location in a route. The time represents how long a vehicle waits at that location before continuing the planned route. The advantage of waiting strategies is that an additional customer might be more efficiently integrated into the route, leading to less total travel time. Mitrović-Minić and Laporte (2004) evaluate four waiting strategies for the PDPTW, where the pick-up and delivery locations reveal over time. Similar to Mitrović-Minić et al. (2004) and Gendreau et al. (2006), Mitrović-Minić and Laporte (2004) use cheapest insertion followed by a tabu search procedure. Also, they discuss two simple strategies; drive-first (DF) and wait-first (WF). The DF strategy is the most universal method within transportation and is performed by driving as soon as possible. The WF strategy is achieved by waiting as long as feasible until most information is gathered. Finally, the research proposes the dynamic waiting strategy (DW) and the

advanced dynamic waiting strategy (ADW), which are a combination of the DF and WF strategies. Both dynamic strategies are based on the creation of service zones and use these zones to divide the waiting time over multiple locations. The results conclude that both dynamic solution approaches perform better than the simple approaches. Besides, the ADW strategy seems to perform best.

Lastly, Branke et al. (2005) study the simplified problem of the occurrence of one additional customer over time. The goal is to maximize the probability of feasible inserting this customer similar to this thesis. The paper compares six simple waiting strategies and two waiting strategies that result from an EA. The paper finds multiple waiting strategies that increase the probability of inserting an additional customer compared to the most common strategy of drive first. Specifically, the best waiting strategy they find is one of the simple strategies introduced in the paper.

2.3 Evolutionary algorithms

An evolutionary algorithm can be defined as a stochastic optimization method based on the process of natural evolution (Zitzler, 1999). The goal of an EA is to solve complex real-world problems with a robust and efficient algorithm. Advantages of the EAs compared to traditional search methods are the large range of potential solutions that the algorithm compares (Holland, 1992) and the simplicity of the method (Sarker et al., 2003). However, the mathematical basis for convergence and optimality is weak, which is one of the largest disadvantages compared to other search methods.

Even though four branches can be defined within EAs, they all have a similar basis. Each possible solution is considered a chromosome, or individual, and has a certain fitness that represents the quality of the solution. The process of the algorithm is to start with an initial set of solutions, the first population, and then search for the optimal value by letting this set evolve like organisms based on two principles; survival of the fittest and reproduction (Holland, 1992). Important steps in the process are parent selection, recombination, and mutation (Razali and Geraghty, 2011).

As mentioned, four categories of EAs can be defined: genetic algorithms (GA), evolutionary strategies (ES), genetic programming (GP), and evolutionary programming (EP). To start with, Holland (1975) and De Jong (1975) introduced the GA by applying it as an adaptive search algorithm to combinatorial problems. Around the same time Rechenberg (1973) introduced the ES in Germany. According to Hoffmeister and Bäck (1991), the main difference between the methods is that an individual in the GA represents a genotype, which means that DNA is defined by binary values, while for the ES the individual represents a phenotype, which means that DNA is defined as real values. Furthermore, GP is firstly discussed by Koza (1992) where GAs were applied on tree-structured chromosomes. However, this type is not usually implemented in optimization problems (Sarker et al., 2003). Finally, EP was proposed by Fogel et al. (1966) in order to achieve artificial intelligence. Sarker et al. (2003) noticed that the EP contrasts from the ES in the recombination step. ESs often use discrete or intermediate recombination, while EP uses probabilistic competition.

When Holland (1975) and De Jong (1975) first introduced the GA, the goal was to find a global search algorithm that would outperform the random search strategies known at that time. De Jong (1975) concluded that even a simple reproductive algorithm was capable of this performance and

that many variations of the algorithm could further improve the results. Additionally, Ulder et al. (1990) showed the application of the GA on the traveling salesman problem (TSP), a variant of the VRP with only one vehicle. Later, Potvin (2009) discussed many different EAs that are applied to the VRP. The focus of the implementation was first on the general VRP or the VRP with time windows (VRPTW), which means that requests need to be fulfilled within a certain time interval. However, recent literature also considers the PDPTW and dynamic problems.

An example of the application of EAs in optimization problems can be found in Chu and Beasley (1997). The paper compares the GA with many existing algorithms in the context of an assignment problem. The assignment problem considers a number of jobs and a number of agents, and each job must be assigned to an agent which is associated with a capacity constraint. Chu and Beasley (1997) conclude that the GA outperforms the other algorithms based on solution quality. However, a disadvantage of the algorithm is the relatively large computational time. Sarker et al. (2003) compare GAs in the knapsack problem and show that the solution quality of the GA slowly regresses as the problem instance becomes larger. More detailed surveys about EAs can be found in Ulder et al. (1990), Bäck et al. (1991), Sarker et al. (2003), Potvin (2009), and Crepinsek et al. (2013).

As mentioned before, the algorithm consists of multiple steps; selection, recombination, and mutation. For each of these steps, there exist multiple methods that can be used. Within the parent selection step, the survival of the fittest is an important characteristic. The aim is that the next generation will focus on better solutions than the parent generation. At the same time, it is essential to keep a genetically diverse population to prevent getting stuck in a local optimum. Therefore, the selection method should balance both exploration and exploitation (Razali and Geraghty, 2011).

Jebari and Madiafi (2013) compare the performance of six different selection methods. The paper divides the methods into two categories: proportional and elitist. Proportional methods create new individuals proportionally to the fitness of their parents, while the elitist methods select the best individuals for the next generation. The first class has the benefit of the genetic diversion, however, the second class has a higher speed of convergence. Jebari and Madiafi (2013) conclude that the performance of the EA depends significantly on the selection method. The paper also introduces a promising algorithm for dynamically deciding the selection method. The algorithm considers both the fitness and the genetic diversity of the population while choosing which selection method to implement per generation.

After selecting parents for reproduction, a method needs to be selected for recombination. Michalewicz (1996) theoretically introduces simple and more involved genetic crossover algorithms. Hasançebi and Erbatur (2000) compare the performance of five simplistic crossover techniques and introduce two new approaches. The first new approach is referred to as the mixed crossover and iterates over three simplistic methods. The second method is called the direct design variable exchange crossover and exchanges sub-strings of the chromosome between individuals according to a probability function. Hasançebi and Erbatur (2000) conclude that the mixed crossover implementation obtains a satisfactory performance level, which is better than the simplistic forms.

Nevertheless, the direct design variable exchange provides the best results irrespective of the problem specifics.

Lastly, after a child has been created, mutation takes place. This process adjusts the created chromosome and thereby attributes to the genetic diversity of the population. Michalewicz (1996) describes three different types of mutation operators. The most popular method is the Gaussian mutation and adds random noise to each value of the chromosome. This random noise is normally distributed with the mean equal to 0 and a fixed standard deviation, σ.

GAs are still evolving and new techniques arise. Nevertheless, the research so far has proven that the algorithm is useful for solving complex optimization problems (Ribeiro Filho, J. L. and Treleavan, P. C. and Alippi, C, 1994). Further implementations of the EA within the context of the VRP can be found in Wang et al. (2007), Branke and Mattfeld (2005) and Branke et al. (2005).

3 Problem description

This section elaborates on the specific characteristics of the problem that are considered in this paper and introduces mathematical notations. The problem description is based on Branke et al. (2005). Given are a set of customers, $C = \{1, \ldots, n\}$, and a depot, 0, in the Euclidean Plane. Moreover, there is a set of vehicles, $M = \{1, \ldots, m\}$, that needs to service all customers in C. Every vehicle $k \in M$ is assigned to a route, which starts and ends at the depot, $r_k = (0, c_{k1}, c_{k2}, \ldots, c_{kn_k}, 0)$, where n_k is equal to the number of customers that vehicle k services and c_{ki} is the ith on the route of vehicle k. Furthermore, the customer visited must be from the set, $c_{ki} \in C$, and every customer is visited exactly once, $c_{ki} \neq c_{mp}$ for $ki \neq mp$.

This thesis focuses solely on travel times and therefore assumes that customers do not have any service time. The travel distances, d_{ij}, are computed as the Euclidean distance between any two customers, $i, j \in C$. To transform distance into time we assume, without any loss of generality, that the speed of the vehicle is equal to one distance unit per time unit (Branke et al., 2005). Each vehicle starts its tour at time 0 and finishes before a predetermined time, $T > 0$. A parameter, t_{rk}, is introduced which is equal to the total travel time of visiting all locations in a route r_k. Slack time is defined as $q_r = T - t_{rk}$, which shows how much time is left that can be spent on waiting or serving new customers. It should be noted that diversion is allowed, which means that a vehicle may change directions while driving when a new customer arrives and this is desirable.

Identical as in Branke et al. (2005), the location of the additional customer is uniformly drawn within the predefined service region. Furthermore, the arrival time of the customer, t_c, is uniformly drawn between 0 and T (Branke et al., 2005). After the information about the customer is revealed the vehicle that can service it within their slack time will include the location in its route. If multiple vehicles have the time to service the customer, the vehicle with the smallest detour is selected (Branke et al., 2005). In the situation that no vehicle is capable of servicing the customer without exceeding the time constraint, the request will be rejected. Hence, the customer will not be serviced.

As mentioned before the aim of this paper is to find a waiting strategy that maximizes the

probability of inserting an additional customer. A waiting strategy refers to a distribution of time over all locations on a route. The sum of these waiting times in a route, r_k is at most equal to the slack of the tour, q_r. By waiting before or during the route, the additional customer could lead to a smaller detour, which can increase the probability of insertion in the route.

4 Methodology

In order to find the waiting strategy that maximizes the probability of insertion, multiple heuristics are compared that create a waiting strategy. First, six simple heuristics are formulated and then evolutionary algorithms are used to find strategies. The simple heuristic strategies are identical to the ones introduced in Branke et al. (2005). Moreover, the two EAs implemented are similar to the ones introduced in the same paper. Furthermore, multiple adjustments are made in the EAs with the aim of increasing the probability of inserting an additional customer.

The upcoming sections elaborate on the details of the methods. Firstly, the simple waiting strategies are explained. Then, the simple forms of the EAs are introduced. Thirdly, the method used to improve the evaluation fitness in the EA is described. Finally, multiple selection methods are discussed and a dynamic selection method is presented.

4.1 Waiting strategies

As mentioned in the literature review, waiting strategies can be defined as allocating time to wait at specific locations during the existing route. If the waiting strategies are chosen strategically, they can improve the probability of inserting an additional customer, by inserting the customer more efficiently in terms of a smaller minimum detour. The six simple waiting strategies discussed in this thesis are: `noWait`, `depot`, `maxDistance`, `location`, `distance` and `variable`, all similar to Branke et al. (2005).

The first two methods are comparable to the methods discussed in Mitrović-Minić and Laporte (2004). The `noWait` strategy is straightforward and lets the vehicles drive immediately until they are returned to the depot. All slack time is then spent at the depot when the vehicle is done with its initial route. As mentioned before, this method is most commonly used in logistical problems. Therefore, we will compare other results relative to the results of this strategy. The `depot` strategy lets vehicles wait at the depot for the entire slack time before leaving. The advantage could be that if the new customer is revealed early enough, this could lead to a smaller detour. However, if the information becomes known too late, the customer needs to be inserted in an inefficient route, as slack time is gone when a vehicle leaves the depot. The customer might even be rejected because there is no slack time left in any of the routes. The idea behind the `maxDistance` strategy is to let the vehicle wait at the location in the route that is the furthest removed from the depot. This method aims for letting multiple vehicles wait at large distances from each other so that they are able to cover a large area in case an additional customer appears. Hence, the customer on route r_k that has the largest distance from the depot, will have a waiting time equal to the slack time of the tour, q_r.

Then the remaining three strategies divide slack time over multiple locations. The `location` strategy equally distributes the total slack time of a tour over all customers in the tour, thus, excluding the depot at the start and the end of the route. Hence, each customer c_{ki} in r_k obtains a waiting time equal to $\frac{q_r}{n_k}$. The `distance` strategy is similar to the previous one, however, the slack time is now distributed over all customers proportionally to the distance traveled. Hence each customer, c_{ki}, is assigned a waiting time equal to $q_r \frac{d_{c_{ki-1}c_{ki}}}{\sum_{j=1}^{n_k} d_{c_{kj-1}c_{kj}}}$, where c_{k0} is the depot. As a result, there is again no waiting time distributed over the depot at the beginning and the end of the route.

Finally, the last strategy, `variable`, is based on a principle which is proven by Branke et al. (2005). The paper proved that for the case of two vehicles, both should find the location at which the total travel time left until the depot is equal to the slack time of their route. Starting from that location the vehicle should approach the depot at approximately half of the normal speed. The strategy formulated based on this principle is to drive without waiting until the total travel time left to the depot becomes less than the slack time. From that point forward, the slack time is proportionally distributed to the driving distance remaining over the customer locations left in the route. This calculation is similar to the `distance` method but adjusted so that only the remaining customers left in the tour are considered. Hence, the customer c_{ki}, where $i \geq l$ is assigned a waiting time equal to $q_r \frac{d_{c_{ki}c_{ki+1}}}{\sum_{j=l}^{n_k} d_{c_{kj}c_{kj+1}}}$, where l is the first location at which the distance left to the depot becomes smaller than the slack time and c_{kn_k+1} is the depot. Finally, there are two waiting strategies formed by the use of an EA, similar to the ones used in Branke et al. (2005). The specifics of these algorithms are discussed in the upcoming section.

4.2 Evolutionary algorithm

To begin with, two versions of the EA are implemented; one with all randomly created individuals in the initial population (EA1), and one where the previously introduced simple waiting strategies are six of the initial individuals (EA2). As mentioned in the literature there are various operators in the algorithm that need to be specified. This section first elaborates on the general algorithm and then discusses the parameters and operators implemented.

In the algorithm, a population consists of multiple individuals, which are equal to possible waiting strategies. Each waiting strategy is represented by a vector containing $n + 2m$ values. There is a value for each of the n customers, plus the depot at the beginning and end of a route for all m vehicles. The values in the vector represent the fraction of the total slack time of a specific route that is spent waiting at a specific location. Hence, the summation over all values of the vector is equal to the number of vehicles m.

The general algorithm starts with an initial population, from which we evaluate each individual's performance by a fitness evaluation method. A form of elitism is implemented, where the best-performing individual is selected for the next population (Branke et al., 2005). Then, a selection method chooses two individuals to procreate. After two parents are selected, a recombination method and a mutation method are applied to form a new individual. To create a feasible solution, the fractions of the waiting times for each route need to be normalized. These procreation steps

are repeated until enough new individuals exist for the new generation. Finally, new populations are created until the stopping criterion is met. Lastly, the individuals from the final population are evaluated and the best performing waiting strategy is returned. Repeating Branke et al. (2005), the size of all populations is equal to 100 individuals, meaning that for all generations created 99 individuals are generated and one is taken from the previous population. Furthermore, the stopping criterion is met when 100 populations are produced.

The goal of the fitness evaluation is to approximate the probability of inserting one additional customer for a waiting strategy. First, 100 customers are generated uniformly random in the service region and uniformly random over the maximum time of the tour. Then, these same 100 customers are tested for each waiting strategy in the population to estimate the probability of inserting one additional customer. The performance of a waiting strategy is measured as the number of additional customers that were feasible to insert in the existing routes. With these results, the waiting strategies are ranked in order of non-decreasing probability of inserting one additional customer.

The linear rank selection (LRS) method is used as a selection method. This method assigns rank 1 to the worst-performing individual of the population and rank 100 to the best-performing individual. To compute the selection probability of each individual their rank is divided by $\frac{100(100+1)}{2} = 5050$. Next, the cumulative probabilities are computed and used to choose an individual from the population. An advantage of the LRS is that it tries to prevent premature convergence, by considering ranks instead of the actual fitness value (Jebari and Madiafi, 2013). After one individual is selected, it is removed from the population so that a child is not created from two identical parents. After a child is created, the removed parent is added to the population so it can be selected for a new child again.

When two parents are chosen, the 2-point crossover method is applied. For this crossover, two random integers are uniformly drawn between 0 and $n+2m$. These values decide which sub-string of the chromosome is taken from one parent and which from the other. A visualization example is presented in Figure 1. Note that the sum of the values in the chromosomes are not equal to 1, and therefore the numbers used are not applicable to the problem characteristics of this thesis.

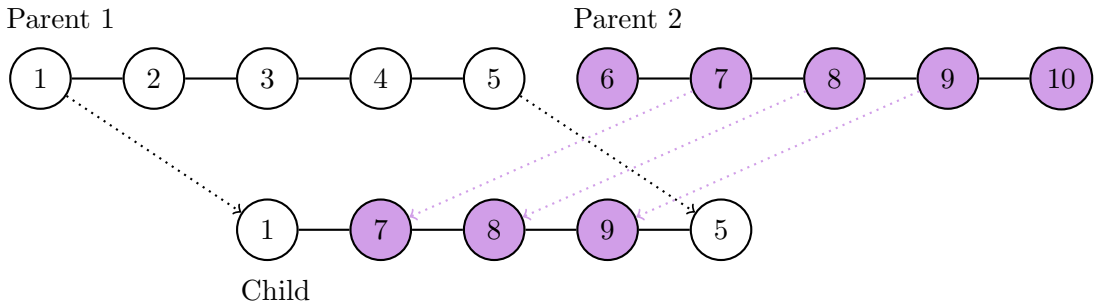

Figure 1: 2-point crossover, with random numbers 2 and 4

After the crossover method is performed, a Gaussian mutation takes place. This is performed by adding a normally distributed random variable to each of the values of a solution. If an element in the waiting strategy becomes smaller than 0, we set it equal to 0, since a negative value does not

apply to the context of the problem. The exact normal distribution is different for each vehicle, $k \in M$, in a problem. The mean of the distribution is always equal to 0, however, the standard deviation is chosen equal to $\frac{1}{6(n_k+2)}$, where n_k is the total number of customers that are included in route k. This standard deviation is chosen with the aim of keeping the mutation relatively small, to retain a correlation between the child and the parents, and set the chance of having to truncate the value to 0 low. For a more detailed discussion of the motivation on this standard deviation, see Appendix A. After the mutations are performed on the total waiting strategy vector, the values need to be normalized per vehicle to obtain feasible solutions for the considered problem. These steps are repeated until a new population is constructed. In Algorithm 1 the broad outlines of the algorithm performed are summarized. Note that this algorithm applies to the simple EAs just discussed, and is different for the upcoming EAs.

Algorithm 1: Evolutionary algorithm

```
Start with an initial population with a size of 100 individuals
numGenerationsCreated ← 0
while numGenerationsCreated < 100 do
    Draw 100 customers uniformly random in the service region and uniformly random
     over time
    Evaluate the fitness of each individual in the current population
    Order the population by non-decreasing probability of insertion
    Select the best individual and add this one to the new generation
    numChildsCreated ← 1
    while numChildsCreated < 100 do
        Select two parents using the LRS
        Perform 2-point crossover
        Mutate the waiting times of the child
        Normalize the values of the waiting times
        Add the new individual to the new generation
        numChildsCreated ← numChildsCreated + 1
    end
    numGenerationsCreated ← numGenerationsCreated + 1
end
Draw 100 customers uniformly random in the service region and uniformly random over
 time
Evaluate the fitness of each individual in the current population
return The individual with the highest fitness
```

4.3 Different numbers of customers used for fitness evaluation

In their paper, Branke et al. (2005) conclude that both EAs are outperformed by the simple `variable` waiting strategy. However, as this strategy is one of the seeds of the original population in the second EA, it seems that the EA cannot recognize the true performance of a strategy.

Therefore, we will further investigate whether the EA can be improved in this direction. Branke et al. (2005) suggest that their research is limited by the evaluation method of the individuals of a population. So far, the individuals are evaluated using a test set of 100 additional customers. This number might be too small, which could result in an approximated probability of insertion which depends too much on the test instance and therefore does not represent the actual fitness of the individual.

For the purpose of creating a more realistic fitness evaluation, the size of the set of customers used in the EA is increased and the performance of the algorithm is evaluated. During the evolution of the populations, the performance is measured as the average and the highest insertion probability. On the other hand, increasing the set of customers used for evaluation is likely to increase the computational time of the algorithm. Therefore, the total computational time of the EA is also measured.

In order to make sure that a fair comparison can be made between multiple scenarios, the average and best performance of each population are computed over a test set equal to the maximum number of customers used for the fitness evaluation, that is 1500 customers. Hence, even though the LRS uses the evaluation after testing on 100 customers, the best and average performance are measured on a larger test set. Moreover, even though for each generation a new test set is created, these test sets are kept equal across the different scenarios. Except for the number of customers used for the fitness evaluation, no changes are made compared to the EA2 discussed before.

4.4 Dynamic decision process as a selection method

In the literature review, it is mentioned that the choice of the selection method significantly affects the solution outcomes. Therefore, comparing different methods for this step might be valuable. Especially the use of a dynamic decisions process for the selection method seemed promising regarding the literature. The idea of the decision process is based on Jebari and Madiafi (2013), however, adjustments have been made. Similar to Jebari and Madiafi (2013), the dynamic selection process starts with creating multiple potential populations, one for each selection method that could be chosen. Then, for each of these potential populations, their relative fitness and their genetic diversity are computed. Both these characteristics are transformed into a single score and the population with the highest score is chosen to be the next generation.

Compared to Jebari and Madiafi (2013), different selection methods are used. Three of the selection methods are common methods in EAs: roulette wheel selection (RWS), linear rank selection (LRS), and tournament selection (TOS). These methods are the same as the algorithms considered in Razali and Geraghty (2011). The RWS method is a simplification of the LRS, where the probability of choosing an individual does not depend on the relative rank, but on the relative fitness of the solution. With the TOS method, multiple random individuals are chosen from the population and the one with the highest fitness survives until the next generation. For this thesis, the binary TOS is used, which means that two random strategies compete against each other.

Besides these commonly used methods, we implement two more selection methods: the top 50 roulette wheel selection (RWS50) and the single knock-out selection (SKO) method. The RWS50

is very similar to the original RWS method, but firstly selects the top 50% individuals of the population, and then performs the original RWS on this selection. The idea behind this deviation is that the worst-performing 50% of the population is already neglected so that the deterministic choice falls in the best performing half of the population. For the SKO method, 8 random strategies are selected, which will compete against each other in pairs. For each comparison, the worst-performing strategy will be removed from the options and the winners will continue to the next round of battles. Lastly, the winner of the final is selected. The concept of this method is to select an individual deterministically from the population. With the purpose of fair comparisons, the same set of customers is used for evaluation in all battles during a specific round. Furthermore, to let the selection not be based on a single test set, new test sets are created for the three rounds of the knock-out battles.

After a child is created based on the selection methods, the mutation takes place by means of the normal distribution introduced before. In the evolutionary algorithms introduced preceding this section, a form of elitism was implemented where the best-performing individual of the population was immediately used as the first individual of the new generation. With the goal of focusing on the different selection methods, we have removed this step. To prevent any confusion, EA^e indicates that the form of elitism is implemented in the algorithm and EA^0 indicates that it is not. As mentioned priorly, after creating potential populations for each of these methods, 2 characteristics need to be compared between the populations: the relative fitness and the genetic diversity. The computations of these characteristics are different from the formulas introduced in Jebari and Madiafi (2013). To evaluate the fitness of the potential population the average fitness of the top five individuals per population is computed, f_{top5}. Then, the relative quality of the method is computed as:

$$quality = \frac{f_{top5}}{f_{max}} \tag{1}$$

Where f_{max} is the maximum fitness encountered over the five different populations. This formula aims to compare the best performances of each generation across the different potential populations.

For the genetic diversity, Jebari and Madiafi (2013) considered a formula that compares the total Euclidean distance between the individuals to the maximum Euclidean distance within a population. In this thesis a simplification of this formula is used:

$$diversity = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} dist_{ij}}{\max_{ij}\{dist_{ij}\} \frac{n(n+1)}{2}} \tag{2}$$

In this formula, $dist_{ij}$ is equal to the Euclidean distance between two solution vectors i, j, where n is equal to the size of the total population, 100. Furthermore, $\max_{ij}\{dist_{ij}\}$ is equal to the maximum Euclidean distance between two individuals of the population. The $\frac{n(n+1)}{2}$ is equal to the total number of unique distances added together. Unique refers to the fact that since the distance from i to j is the same as from j to i, this distance is only counted once. Hence, the genetic diversity represents the average Euclidean distance between individuals in a population with respect to the maximum distance. In case that a population only contains exactly equal

waiting strategies, then the diversity is set equal to 0.

Finally, the total score of each of the potential populations is computed as follows:

$$score = \frac{1}{t} diversity + \frac{t-1}{t} quality \tag{3}$$

In this formula, t represents the generation that is considered. Hence, for the first generations, when t is small, diversity is relatively more important than quality. However, over time the quality will become relatively more important compared to the diversity. Since *quality* and *diversity* are both values between the 0 and 1, no further scaling operations are needed to insert these numbers in the *score* formula. The potential population with the highest score is chosen. The goal of this adaptation compared to the EA2, is to improve the algorithm and find a better performing solution.

In order to evaluate the results of the dynamic decision process, the evolution of the algorithm using the dynamic process is compared to the algorithm using the five static selection methods. Over all populations, the average performance, best performance, and genetic diversity are stored. To create a fair comparison, all the performance measures are evaluated using the same set of additional customers.

5 Results

This section presents the results of the discussed methods, applied to seven different VRP instances from the Beasley's OR library (Beasley, 1990): c50, c75, c100, c100b, c120, c150, and c199. From these instances, the first set of optimal routes presented is used as the initial routes. Each problem instance has a maximum time allowed, T, equal to the longest route over the set of vehicles in a problem instance, $\max_{rk}\{t_{rk}\}$. The predefined service region of each instance is defined as a rectangle characterized by the minimum and maximum x and y coordinates of the original set of customers in that instance. Furthermore, all results presented are averages of 20 simulation runs and for each run, a new seed is used for the genetic algorithm. Lastly, a computer with an i7 processor and 16 GB of memory is used while implementing the algorithms in Eclipse IDE version 4.18.0.

5.1 Comparing simple waiting strategies and simple evolutionary algorithms

To start with, the performance of the simple waiting strategies and the two evolutionary algorithms are compared. It is important that all strategies are evaluated against the same 20 sets of 1000 additional customers. The focus of this thesis is on the probability of inserting an additional customer, however, also the minimum detour is evaluated to provide additional information.

Table 1 shows the number of customers out of 1000 customers that were rejected because they could not be inserted without exceeding the maximum time, for each strategy and each problem instance. Furthermore, Table 2 provides the average over all problem instances of the relative performance of each strategy compared to the `noWait` strategy. Firstly, the results for the problem instance c100 stand out compared to the other problem instances and compared to the results

presented by Branke et al. (2005). For this specific problem, the absolute number of customers that are rejected is surprisingly low. For some strategies this number falls below 200, meaning that more than 80% of the new customers can be inserted. However, looking at the details of the problem characteristic it can be concluded that one of the vehicles in the problem has a slack time of 81.9% compared to the maximum time. Hence, this could cause a relatively high probability of inserting an additional customer.

For all other problem instances, the `depot` strategy results in the highest number of rejected customers. This strategy is also associated with a relative performance of 151.5%. This could be explained by the fact that if an additional customer appears after the vehicle has left the depot, it can never be inserted in that route. Therefore, only the customers that appear relatively early have a possibility to be inserted. After the `depot` strategy, the `noWait` and `maxDistance` strategies appear to have the lowest relative performance on average, 100.0% and 98.8% respectively. Hence, it can be concluded that spreading the slack time over multiple locations instead of waiting at a single location improves the probability of inserting an additional customer.

Considering the methods that distribute their slack time over multiple locations, it can be concluded that the `location` strategy performs slightly better than the `distance` strategy for all problem instances. However, this difference in relative performance is small (1.7 percentage points) and differs across the problem instances. Furthermore, the `variable` strategy has the best average relative performance over all methods, as in Branke et al. (2005). However, if the individual problem instances are considered it shows that for problems c100 and c100b no real performance difference is observed compared to the `location` strategy. In addition, the EAs perform slightly better for problems c199 and c100b. However, as the `variable` strategy shows the best robust results over all problem instances, it can be concluded that this method performs best so far, which is similar to the conclusions of Branke et al. (2005).

The relative performance of the two different evolutionary algorithms is similar for all problem instances. Moreover, for the different problem instances, there is no clear conclusion on whether one of the two EAs performs better. However, as also noticed by Branke et al. (2005), it is surprising that the EA2 does not perform at least as well as the `variable` strategy, since this strategy is chosen as a seed for the EA2. Nevertheless, the EAs have the lowest rejected customers on average, after the `variable` strategy considering the average relative performance.

Comparing the results with the outcomes reported in Branke et al. (2005), it is easily spotted that this thesis has lower absolute numbers of customers that are rejected for each problem and each strategy. This is even observed for the `noWait` strategy, which is very straightforward to implement. Therefore, it can be concluded that it is likely that there are some differences in the algorithms and/or problems implemented. These different results in the absolute numbers also affect the relative performances per problem instance, however, the conclusions based on the average relative performance are similar. The last difference that can be noted, is that the average relative performance of the strategies that divide their slack time over multiple locations is roughly around 5 percentage points better than the performances reported by Branke et al. (2005).

Table 1: Number of customers that could *not* be inserted for the different waiting strategies and all test instances

Problem	**Waiting strategy**	**Customers not inserted**	**Relative perf. (%)**
c50	`noWait`	395.1 ± 3.7	100.0
	`depot`	887.9 ± 1.6	224.7
	`maxDistance`	497.4 ± 3.4	125.9
	`location`	457.0 ± 3.9	115.7
	`distance`	460.5 ± 3.9	116.6
	`variable`	387.2 ± 3.6	98.0
	`EA1`^{e}	439.4 ± 6.1	111.2
	`EA2`^{e}	441.8 ± 6.9	111.8
c75	`noWait`	354.1 ± 1.9	100.0
	`depot`	557.1 ± 3.1	157.3
	`maxDistance`	366.3 ± 2.4	103.4
	`location`	314.3 ± 2.2	88.8
	`distance`	318.2 ± 2.0	89.9
	`variable`	313.2 ± 2.0	88.4
	`EA1`^{e}	316.0 ± 3.8	89.2
	`EA2`^{e}	315.8 ± 2.8	89.2
c100	`noWait`	388.0 ± 3.1	100.0
	`depot`	365.5 ± 4.0	94.2
	`maxDistance`	250.4 ± 3.2	64.5
	`location`	178.5 ± 2.8	46.0
	`distance`	177.8 ± 2.7	45.8
	`variable`	178.7 ± 2.9	46.1
	`EA1`^{e}	179.1 ± 2.8	46.2
	`EA2`^{e}	179.5 ± 2.8	46.3
c100b	`noWait`	435.9 ± 2.7	100.0
	`depot`	609.1 ± 3.2	139.7
	`maxDistance`	407.2 ± 2.5	93.4
	`location`	379.7 ± 2.5	87.1
	`distance`	398.3 ± 2.3	91.4
	`variable`	381.1 ± 2.4	87.4
	`EA1`^{e}	383.0 ± 4.3	87.9
	`EA2`^{e}	379.5 ± 2.7	87.1
c120	`noWait`	372.7 ± 2.9	100.0
	`depot`	621.2 ± 3.2	166.7
	`maxDistance`	414.7 ± 3.0	111.3
	`location`	386.1 ± 2.4	103.6
	`distance`	401.5 ± 3.0	107.7
	`variable`	348.6 ± 2.9	93.5
	`EA1`^{e}	371.0 ± 3.5	99.5
	`EA2`^{e}	367.2 ± 3.3	98.5
c150	`noWait`	343.5 ± 3.4	100.0
	`depot`	517.5 ± 3.4	150.7
	`maxDistance`	345.8 ± 3.8	100.7
	`location`	292.9 ± 3.9	85.3
	`distance`	295.4 ± 4.0	86.0
	`variable`	291.0 ± 3.8	84.7
	`EA1`^{e}	295.2 ± 4.0	85.9
	`EA2`^{e}	295.6 ± 4.4	86.1
c199	`noWait`	340.7 ± 2.6	100.0
	`depot`	432.5 ± 2.6	126.9
	`maxDistance`	315.1 ± 2.6	92.5
	`location`	281.5 ± 2.7	82.6
	`distance`	283.4 ± 2.6	83.2
	`variable`	285.4 ± 2.7	83.8
	`EA1`^{e}	281.0 ± 2.8	82.5
	`EA2`^{e}	282.0 ± 2.7	82.8

Note. For each case, the mean, standard error, and the relative performance in percent with respect to `noWait`, are reported.

Table 2: Number of additional customers that could *not* be inserted, relative percentage, compared with not waiting at all

Waiting strategy	**Relative performance (%)**
`noWait`	100.0
`depot`	151.5
`maxDistance`	98.8
`location`	87.0
`distance`	88.7
`variable`	83.1
`EA1`^{e}	86.1
`EA2`^{e}	86.0

To summarize, dividing the slack time of a route over multiple locations increases the chances of being able to feasible insert a new random customer overall. The best strategy found so far is the `variable` strategy which is proven to be optimal for two vehicles (Branke et al., 2005), but appears to also perform well when multiple vehicles are considered. After this strategy, the evolutionary algorithms seem to perform best.

Next, Table 3 shows the average detour for each problem instance and each strategy, and Table 4 presents the average relative performances of each strategy compared to the `noWait` strategy in terms of the detour. As mentioned before, waiting at a location before the entire route is finished, may result in more insertion options when the additional customer becomes known. Therefore, it is expected that all strategies perform better than the strategy of only waiting at the depot in the end. From Table 3 it is visible that the `depot` strategy results in a shorter detour compared to the `noWait` strategy in three out of the seven problem instances. Hence, the strategy of waiting at the depot before driving can be beneficial compared to first driving the total tour. However, if this is the case, in reality, seems to depend on the problem.

Besides the `noWait` and `depot` strategies, the `maxDistance` and `variable` strategy seem to have the worst relative performance compared to the `noWait` strategy. Even though other strategies have a smaller average detour, these two strategies already decrease the detour by almost 30% on average. Interesting is that the `maxDistance` strategy, had a lower probability of insertion but compared to the `noWait` strategy has a relatively good performance for the average detour. Concluding from these results, it is likely that this strategy can perform well if customers become known relatively early in time, or the customer with the largest distance from the depot is relatively late in the route. Furthermore, both the EAs decrease the average detour by a relatively large amount. However, no clear performance distinction can be made between the two algorithms.

Lastly, the average performances of the `location` and `distance` strategies are again very close to each other. This is also not that surprising as the allocation of the waiting times is also similar. These two strategies attain the smallest average detour out of all strategies considered. All in all, the average results of these relative performances are very similar to the results that are presented by Branke et al. (2005).

Table 3: Average detour per customer inserted, for the different waiting strategies and all problem instances

Problem	Waiting strategy	Average detour	Relative perf. (%)
c50	noWait	5.12 ± 0.06	100.0
	depot	2.07 ± 0.06	40.4
	maxDistance	4.32 ± 0.05	84.4
	location	3.08 ± 0.03	60.2
	distance	3.16 ± 0.04	61.7
	variable	4.96 ± 0.06	96.9
	EA1[e]	3.72 ± 0.08	72.7
	EA2[e]	3.70 ± 0.08	72.3
c75	noWait	7.91 ± 0.08	100.0
	depot	11.17 ± 0.21	141.2
	maxDistance	5.89 ± 0.05	74.5
	location	5.12 ± 0.06	64.7
	distance	5.15 ± 0.06	65.1
	variable	6.20 ± 0.07	78.4
	EA1[e]	5.38 ± 0.10	68.0
	EA2[e]	5.29 ± 0.09	66.9
c100	noWait	18.12 ± 0.18	100.0
	depot	8.60 ± 0.14	47.5
	maxDistance	5.84 ± 0.06	32.2
	location	6.60 ± 0.06	36.4
	distance	6.06 ± 0.05	33.4
	variable	6.57 ± 0.06	36.3
	EA1[e]	6.59 ± 0.10	36.4
	EA2[e]	6.78 ± 0.13	37.4
c100b	noWait	16.54 ± 0.21	100.0
	depot	14.32 ± 0.18	86.6
	maxDistance	13.97 ± 0.17	84.5
	location	12.16 ± 0.16	73.5
	distance	11.98 ± 0.17	72.4
	variable	12.43 ± 0.15	75.2
	EA1[e]	12.43 ± 0.19	75.2
	EA2[e]	12.68 ± 0.18	76.7
c120	noWait	11.68 ± 0.19	100.0
	depot	23.23 ± 0.30	198.9
	maxDistance	11.47 ± 0.23	98.2
	location	8.37 ± 0.22	71.7
	distance	9.35 ± 0.23	80.1
	variable	7.60 ± 0.20	65.1
	EA1[e]	8.53 ± 0.21	73.0
	EA2[e]	8.33 ± 0.26	71.3
c150	noWait	8.48 ± 0.07	100.0
	depot	11.71 ± 0.19	138.1
	maxDistance	5.23 ± 0.04	61.7
	location	4.54 ± 0.05	53.5
	distance	4.39 ± 0.05	51.8
	variable	5.77 ± 0.06	68.0
	EA1[e]	4.73 ± 0.06	55.8
	EA2[e]	4.73 ± 0.08	55.8
c199	noWait	7.97 ± 0.07	100.0
	depot	11.95 ± 0.15	149.9
	maxDistance	4.81 ± 0.06	60.4
	location	4.29 ± 0.04	53.8
	distance	4.00 ± 0.04	50.2
	variable	5.10 ± 0.05	64.0
	EA1[e]	4.40 ± 0.05	55.2
	EA2[e]	4.33 ± 0.04	54.3

Note. For each case, the mean and the standard error, as well as the relative performance in percent with respect to `noWait`, are reported.

Table 4: Relative average distance of detour to integrate an additional customer compared with not waiting at all

Waiting strategy	**Relative performance (%)**
`noWait`	100
`depot`	114.7
`maxDistance`	70.8
`location`	59.1
`distance`	59.2
`variable`	69.1
`EA1`^{e}	62.3
`EA2`^{e}	62.1

Finally, Figure 13 summarizes all results in a scatter plot, where both the insertion probability and the minimum detour are included. All things considered, it can be concluded that introducing waiting strategies is beneficial for both the performance characteristics. The `maxDistance` strategy can reduce the average detour compared to not waiting, but does not perform exceedingly better for the relative percentage of customers that are not inserted. Considering the strategies that divide the slack time over multiple locations in a tour, both EAs perform very similarly and result in improvements for both performance characteristics. Moreover, the `location` and `distance` strategies are the best performing strategies regarding the average minimum detour. Finally, the `variable` method, might not provide the smallest average detour but results in the lowest number of customers rejected.

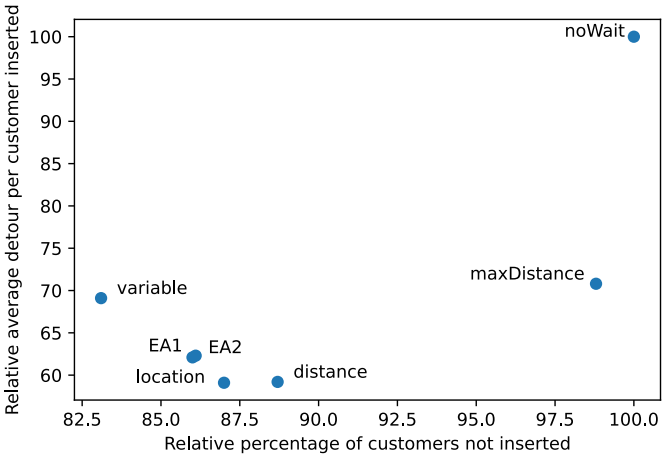

Figure 2: Comparison of the heuristics with respect to the average number of customers not inserted and the average length of the detour for a successful insertion of a new customer

Note. Strategy `depot` has been omitted for clarity because it performs much worse than the other strategies.

Note. Both EAs use the form of elitism discussed.

5.2 Comparing different numbers of customers used for fitness evaluation

To improve the evolutionary algorithm, multiple sizes of the test set of customers used in the fitness evaluation step are considered. The fitness evaluation aims to approximate the insertion probability

of a specific waiting strategy. This information is used so that better-performing strategies can be used as parents for generating new populations. Hence, the intention of improving this step is to make sure better-performing individuals are chosen, which could result in populations and individuals associated with a higher probability of inserting one additional customer.

The different sizes compared are the initial 100, 500, 1000, and 1500 customers. Over all populations, the average insertion probability and the insertion probability of the best-performing individual are reported. As mentioned before, the test sets of computing these results are equal across the different test sizes, in order to be able to make a fair comparison. The test sets of determining the average and best performance consist of 1500 customers, to minimize a possible bias towards the test set. Lastly, for all runs, the computational time is reported in milliseconds. A detailed overview of all results per instance can be found in Appendix B. In this section representative graphs of the results and the averages of the relative computational time compared to the size of 100 customers are shown.

Figures 3 and 4 present the average and highest probability of inserting one additional customer per generation for problem instance c120 and c199 respectively. The figures show a positive trend between the number of customers used for evaluation and the average performance of the population. Specifically, the lower the size of the test set, the lower the average performance of the population independent of the generation. Before looking in more detail at the graphs, it is important to note that the scale of the y-axis is not the same. Taking this into consideration, it is visible that the relative differences between the test instances depend on the specific problem. For example, in c120 the average performance using the test size of 100 is considerably worse than using 1500 customers, a little over one percentage point, while for c199 this difference is approximately half a percentage point. Furthermore, it can be noticed that the differences in the average performance can already be established after approximately 10 generations for both problem instances. Hence, choosing parents based on a fitness evaluation calculated using more test customers, leads to a higher average performance of the population. However, it is important to note that these are relatively small differences.

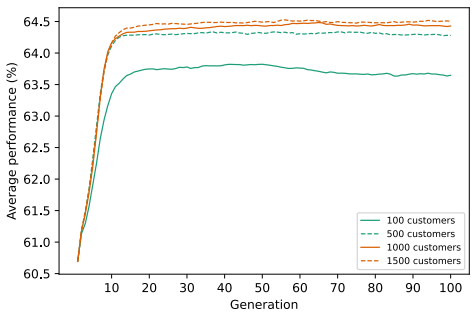

Figure 3: Average probability of insertion over generations for different sizes of customers used for the fitness evaluation for problem c120.

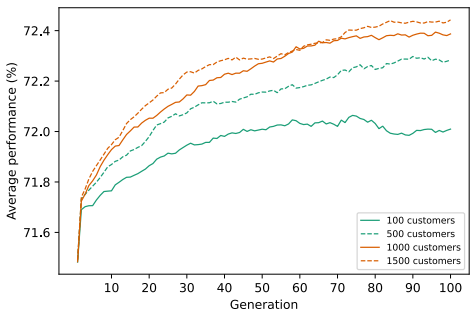

Figure 4: Average probability of insertion over generations for different sizes of customers used for the fitness evaluation for problem c199.

Figures 5 and 6 present the best-performing individual of the population for the different test sizes and uses the same problem instances as before. Again, it is important to note that for problem c120 the y-axis covers over 1 percent, while for c199 this is 0.7 percent. However, taking this into consideration, a similar trend as with the average performance can be seen. Specifically, the higher the number of customers in the test set, the better the performance of the best individual across populations. Interesting to notice is that for problem c120, using 100 customers results in a lower-performing best individual than that is encountered in the first population. Hence, this could indicate that 100 customers might be too small to acknowledge the true performance of an individual. To conclude, selecting individuals based on larger test sets for fitness evaluation, leads to a higher performance of the best individual of the population.

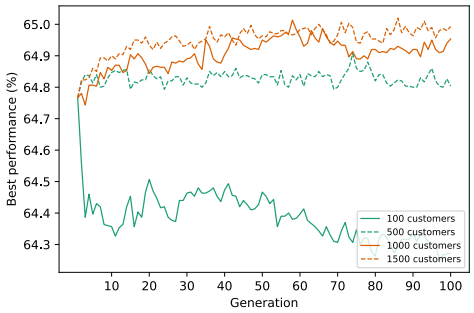

Figure 5: Highest insertion probability over generations for different sizes of customers used for the fitness evaluation for problem c120.

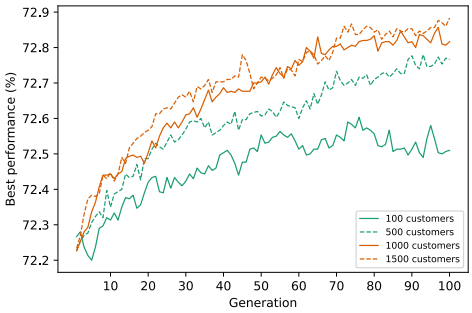

Figure 6: Highest insertion probability over generations for different sizes of customers used for the fitness evaluation for problem c199.

Lastly, the computational times are considered and presented in Table 5. As expected the computational time increases when the fitness evaluation is based on a larger test sample. This observation holds for all problem instances. It is important to note that the largest part of the computational time is likely associated with computing the average and best performance based on a test set of 1500 customers. Therefore, the relative differences between the computational times would be even larger when this step would not be necessary.

Table 5: Relative computational time of the EA^e for different sizes of the group of customers used for fitness evaluation compared with 100 customers

Customers used for evaluation	**Relative computational time (%)**
100	100.0
500	127.0
1000	160.5
1500	184.3

To conclude, using more customers to evaluate the population for selecting new parents, improves both the average performance of the population and the best-performing individual. Hence, the initial 100 customers used, might not be enough to accurately evaluate the insertion probability of a waiting strategy. However, the differences in the average performance and the best performance are relatively small. Moreover, increasing the size of the test set considerably increases the

computational time of the algorithm. Hence, if time is not restricted and the goal is to obtain the best possible result, a test set of 1500 customers should be used for evaluating the fitness of a waiting strategy. Nevertheless, the figures presented in this section show that it is already beneficial to increase the test set to 500 customers. Moreover, it seems as when the size of the test set is large, an increase of 500 customers leads to a relatively smaller increase in performance, compared to when the test set is small. Hence, the benefit of a higher performance relative to the increase in computational time appears to decrease when the test set size grows.

5.3 Comparing selection methods including the dynamic decision process

This section will report the results of comparing five static and one dynamic selection methods. From the results in the previous section, it could be concluded that increasing the size of the test set used in the fitness evaluation improved the performance of the evolutionary algorithm. However, it also showed that it would significantly increase the computational time of the algorithm. Due to time limitations, we will evaluate the fitness of the populations for different selection methods using a set of 500 customers. Furthermore, the average and highest insertion probability are computed over a test set of 1000 customers. As before, to make sure comparisons between methods are fair, all test sets are equal for the same generation across the different selection methods. Also, all the results reported are again the average of 20 simulations with each time a new seed for the EA.

In Appendix C the detailed results of all problem instances are presented. As the results are comparable for the different instances and for convenience, only two problem instances are discussed. Figures 7, 8, and 9 present the average insertion probability, the highest insertion probability, and the genetic diversity, respectively, for the problem instance c50. Considering the average insertion probability for c50, we notice that the average performance over all generations is relatively low for the RWS and TOS method, compared to the other four options. Furthermore, these two methods have a relatively low highest insertion probability over all generations. Hence, it can be concluded that the RWS and the TOS method do not achieve high performance compared to the other methods. Moreover, it can be seen that the average insertion probability is relatively unstable for the TOS algorithm compared to all other methods.

Looking at the other four selection methods, it is noticeable that the differences in average and best performance lay within approximately 1.5 percentage points. Hence, these methods perform similarly. However, from Figure 7 it can be observed that for LRS and RWS50 the average performance converges faster than for the dynamic and SKO method. It can also be noticed that the LRS and RWS50 result in a slightly higher performance of the best individual of the population, compared to the dynamic and SKO method. To conclude, only considering the performance of the algorithm, the LRS and RWS50 selection methods seem to perform best for the c50 problem instance.

Lastly, we consider the genetic diversity of the population. The main motivation of the dynamic selection method is that it can prioritize the genetic diversity at the beginning of the search and later focus on performance. Hence, it was expected that high diversity in the beginning stage of the search, could lead to higher performance for individuals later. The first interesting thing is

that the genetic diversity becomes 0 after approximately 35 generations for the TOS method. This is a bit surprising as the chosen individuals still undergo mutations. However, this result indicates that after normalizing the waiting strategy, the mutations are not of a significant effect anymore. Another interesting observation is that even though the RWS and the TOS performed poorly while looking at the insertion probabilities, the RWS has opposed to the TOS the highest genetic diversity over the different generations. From this, we may conclude that having a high genetic diversity, does not generally lead to high performance. Furthermore, where LRS and RWS50 achieved slightly higher performance results, the dynamic and SKO method seem to attain a slightly higher genetic diversity.

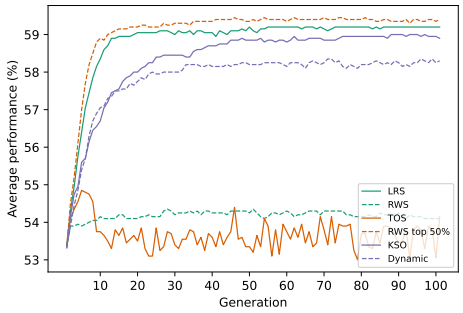

Figure 7: Average insertion probability over generations for different selection methods applied to problem c50.

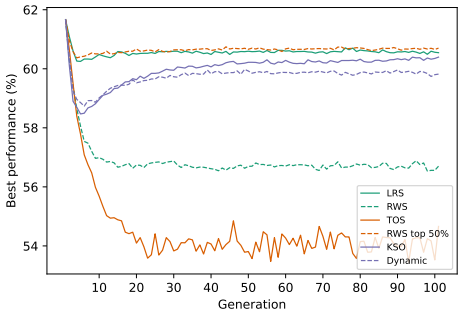

Figure 8: Highest insertion probability over generations for different selection methods applied to problem c50.

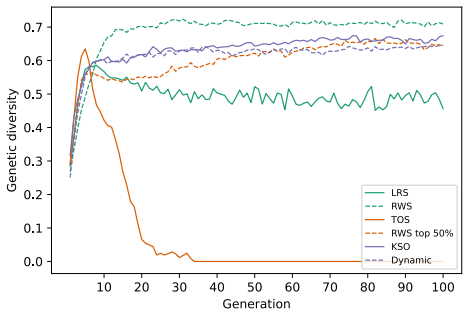

Figure 9: Genetic diversity over generations for different selection methods applied to problem c50.

Next, average insertion probability, highest insertion probability, and genetic diversity of the c100b problem instance are presented in Figures 10, 11, and 12, respectively. From Figure 10, it can be concluded that the average performance of all methods is less stable than they were for c50. However, it can also be concluded that the TOS method generates the most unstable results, which was also found in problem instance c50. Furthermore, the RWS and TOS method again results in the lowest average and best insertion probability over generations. However, the distinction between LRS and RWS50 compared to SKO and the dynamic method, which was still visible for c50, has disappeared for problem c100b.

Moreover from Figure 12, we see a similar trend for the genetic diversity while using the TOS method for problems c50 and c100b. Also, the trend that the LRS method generates is very

comparable to the results generated using the problem c50. However, the distinction between the diversity of the RWS method and other methods disappeared.

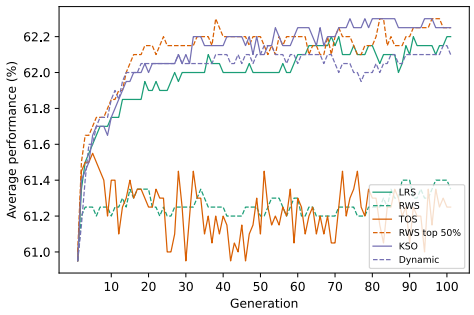

Figure 10: Average insertion probability over generations for different selection methods applied to problem c100b.

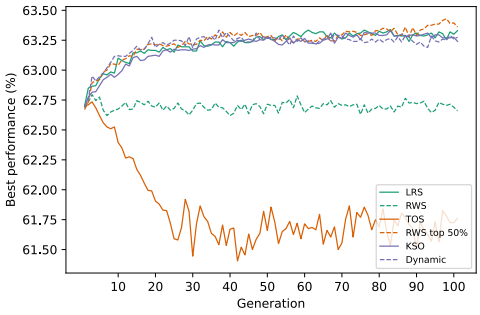

Figure 11: Highest insertion probability over generations for different selection methods applied to problem c100b.

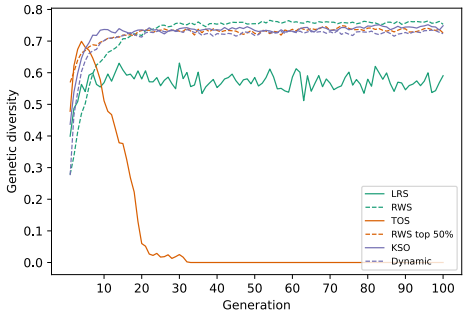

Figure 12: Genetic diversity over generations for different selection methods applied to problem c100b.

Finally, the relative computational times compared to the LRS method are presented in Table 6. The relative times are compared to the LRS method since this is the initial method used in this thesis. The results show that LRS, RWS, TOS, and RWS50 have approximately the same computational time. Moreover, the SKO method has a computational time that is more than six times as large as the LRS method. Additionally, the dynamic method is almost 8 times as large as the time necessary for the LRS method. In Appendix C some additional output is presented per problem instance. This output shows that while using the dynamic decision process as a selection method the relative frequency of the RWS50 and SKO method is around 50%. Moreover, the relative frequency of the other three selection methods falls below the 1% for each instance. Hence, the dynamic decision process uses the two newly introduced methods most often.

Combining all results, several conclusions can be formulated. First of all, the methods RWS and binary TOS achieved lower insertion probabilities compared to all other methods and are therefore not recommended to use. Furthermore, the performance of all other methods are very similar and hence no clear method can be favored while looking at the results solely. However, looking at the computational times, the SKO and dynamic method have a significantly larger computational time. Therefore, the RWS50 and LRS methods are suggested to provide the best performance while keeping the computational time relatively low. Furthermore, the advantage of

Table 6: Relative computational time of the EA^0 for different selection methods compared to the LRS method

Selection method	**Relative computational time (%)**
LRS	100.0
RWS	100.9
TOS	101.1
RWS50	100.7
SKO	657.1
Dynamic	893.8

using a dynamic selection method that was suggested by Jebari and Madiafi (2013) cannot be concluded from the results of this thesis.

5.4 Final comparison

In this final result section, the conclusions of the past sections will be used to improve the initial EA. Hence, the size of the test set of customers used for evaluating the fitness of the population will be increased. The best results are expected for a test set of 1500 customers, however, due to time limitations and a decreasing marginal benefit in performance increase, a set of 1000 customers is used. Furthermore, the LRS, RWS50, SKO, and dynamic selection methods are compared, since they proved to perform best. The performances of these EAs are compared to the `variable` strategy, as that strategy was concluded to be the best strategy in Section 5.1. Furthermore, the original use of elitism, where the best performing individual is immediately used for the next generation is removed from the algorithm, similar to Section 5.3. For each EA a new seed is used and the test sets used per generation for selecting parents are equal like before. Furthermore, the test set used to evaluate the final performance of the waiting strategies is the same for all EAs and the `variable` strategy and consists of 1000 customers. Moreover, the results presented are again the averages and standard errors of 20 simulations.

Similar to before, both the number of rejected customers and the average detour are reported for each method and problem instance. Table 7 presents the customers rejected per problem instance and per waiting strategy. Interesting is that for two out of the seven problem instances (c50 and c120) the `variable` strategy has the lowest average of customers rejected. For four out of the five other problem instances (c75, c100b, c150, and c199), the waiting strategy that results from the $\texttt{EA\_RWS50}^0$ performs best. In the one remaining problem (c100) the waiting strategy $\texttt{EA\_LRS}^0$ has a slightly lower number of customers rejected than the $\texttt{EA\_RWS50}^0$ strategy. Nonetheless, it is important to notice that the differences in relative performance are minimal.

Next, Table 8 presents the average relative performance of the waiting strategies compared to the `variable` strategy. As the best-performing strategy varies over the different problem instances and the differences are small, the average performances should be considered carefully. If we look at Table 8, it shows that all relative performances are in a range with a length of 1.6 percentage points. Hence, as concluded before, the strategies perform on average relatively similarly. Nevertheless, the $\texttt{EA\_RWS50}^0$ strategy seems to perform best with an average relative performance of 99.2%.

Table 7: Number of customers that could *not* be inserted for the different waiting strategies and all test instances

Problem	Waiting strategy	Customers not inserted	Relative perf. (%)
c50	variable	381.4 ± 3.1	100.0
	EA_LRS^0	412.6 ± 3.5	108.2
	EA_RWS50^0	400.4 ± 2.6	105.0
	EA_Dynamic^0	415.2 ± 2.5	108.9
	EA_SKO^0	405.4 ± 2.9	106.3
c75	variable	320.7 ± 3.7	100.0
	EA_LRS^0	324.0 ± 5.6	101.0
	EA_RWS50^0	311.3 ± 3.4	97.1
	EA_Dynamic^0	312.9 ± 3.4	97.6
	EA_SKO^0	312.3 ± 3.5	97.4
c100	variable	175.7 ± 2.5	100.0
	EA_LRS^0	173.2 ± 2.5	98.6
	EA_RWS50^0	173.8 ± 2.4	98.9
	EA_Dynamic^0	174.6 ± 2.3	99.4
	EA_SKO^0	174.0 ± 2.4	99.0
c100b	variable	387.7 ± 2.9	100.0
	EA_LRS^0	382.6 ± 3.8	98.7
	EA_RWS50^0	376.7 ± 2.7	97.2
	EA_Dynamic^0	380.6 ± 2.9	98.2
	EA_SKO^0	379.5 ± 2.8	97.9
c120	variable	355.4 ± 3.6	100.0
	EA_LRS^0	364.5 ± 3.9	102.6
	EA_RWS50^0	359.7 ± 3.5	101.2
	EA_Dynamic^0	368.4 ± 3.5	103.7
	EA_SKO^0	365.4 ± 3.7	102.8
c150	variable	302.0 ± 2.6	100.0
	EA_LRS^0	299.6 ± 2.9	99.2
	EA_RWS50^0	297.2 ± 2.5	98.4
	EA_Dynamic^0	300.2 ± 2.6	99.4
	EA_SKO^0	298.0 ± 2.3	98.7
c199	variable	279.5 ± 3.7	100.0
	EA_LRS^0	272.6 ± 3.8	97.5
	EA_RWS50^0	270.8 ± 3.7	96.9
	EA_Dynamic^0	273.4 ± 3.5	97.8
	EA_SKO^0	272.0 ± 3.4	97.3

Note. For each case, the mean, standard error, and the relative performance in percent with respect to `variable` strategy, are reported.

Table 8: Number of additional customers that could *not* be inserted, relative percentage, compared with `variable` strategy

Waiting strategy	Relative perfromance (%)
variable	100
EA_LRS^0	100.8
EA_RWS50^0	99.2
EA_Dynamic^0	100.7
EA_SKO^0	99.9

Similar to Section 5.1 the average detours are compared between the various waiting strategies. These values can be found for each problem instance in Table 9. Section 5.1 showed that the `variable` strategy did not achieve the minimal average detour compared to the other simple waiting strategies. Comparing the results of the `variable` strategy with the detours of the different EAs, show that in two out of seven problem instances (c100b and c120) the `variable` strategy achieves the lowest average detour. In three out of the seven instances (c50, c75, and c150) the `EA_Dynamic`^{0} strategy achieves the smallest average detour. For instance c75 and c199 the `EA_SKO`^{0} and the `EA_LRS`^{0} strategy attain the lowest detour, respectively. Hence, also for the average detour, the results vary over the different instances. Nevertheless, compared to the relative performance of the customers rejected, the relative differences between the strategies are considerably large.

Table 10 presents the mean of the relative performance of the average detours compared to the `variable` strategy of all waiting strategies. Again, it is important to notice that the relative performance differs over the problem instances and hence the average relative performance should be considered carefully. However, if the values are considered, it is shown that similar as in Section 5.1 the `variable` strategy does not achieve the average smallest detour. The `EA_Dynamic`^{0} strategy achieves the lowest relative performance with a difference of 8.3 percentage points compared to the `variable` strategy.

For consistently comparing the different waiting strategies in this section and Section 5.1, Figure 13 presents a scatter plot of the average relative performance of the different strategies. When interpreting this figure, it is important to notice that the range of the relative percentage of customers not inserted is limited to a length of 1.6 percentage points. Hence, it seems as if the `EA_RWS50`^{0} strategy performs significantly better than the `variable` strategy, however, as seen before this difference is small and inconsistent over multiple problem instances. However, the figure shows that all different EAs have a lower relative average detour per customer inserted than the variable strategy.

All in all, comparing the constructed EAs based on the results of previous sections with the best simple waiting strategy, `variable`, we find inconsistent results over multiple problem instances. Considering the goal of maximizing the probability of inserting additional customers, the `EA_RWS50`^{0} seems to perform best. However, considering that this strategy needs significantly more computational time than the simple `variable` strategy and that the performance difference is minimal, the `variable` strategy can be preferred for convenience. Yet, if the average detour is considered the `variable` strategy performs the worst out of the five strategies compared. Moreover, the best-performing strategy for this measure is the `EA_Dynamic`^{0} strategy.

Table 9: Average detour per customer inserted, for the different waiting strategies and all problem instances

Problem	Waiting strategy	Average detour	Relative perf. (%)
c50	variable	4.93 ± 0.05	100.0
	EA_LRS^0	3.92 ± 0.06	79.5
	EA_RWS50^0	4.14 ± 0.05	84.0
	EA_Dynamic^0	3.76 ± 0.05	76.3
	EA_SKO^0	3.99 ± 0.04	81.4
c75	variable	6.39 ± 0.08	100.0
	EA_LRS^0	5.62 ± 0.11	87.9
	EA_RWS50^0	5.55 ± 0.08	86.9
	EA_Dynamic^0	5.49 ± 0.08	85.9
	EA_SKO^0	5.53 ± 0.08	86.5
c100	variable	6.51 ± 0.06	100.0
	EA_LRS^0	6.53 ± 0.07	100.3
	EA_RWS50^0	6.33 ± 0.06	97.2
	EA_Dynamic^0	6.44 ± 0.06	98.9
	EA_SKO^0	6.32 ± 0.08	97.1
c100b	variable	12.37 ± 0.14	100.0
	EA_LRS^0	12.72 ± 0.18	102.8
	EA_RWS50^0	12.75 ± 0.15	103.1
	EA_Dynamic^0	12.60 ± 0.15	101.9
	EA_SKO^0	12.66 ± 0.16	102.3
c120	variable	7.19 ± 0.20	100.0
	EA_LRS^0	8.22 ± 0.18	114.3
	EA_RWS50^0	8.33 ± 0.25	115.9
	EA_Dynamic^0	7.99 ± 0.22	111.1
	EA_SKO^0	8.19 ± 0.20	113.9
c150	variable	5.78 ± 0.07	100.0
	EA_LRS^0	4.99 ± 0.07	86.3
	EA_RWS50^0	4.93 ± 0.06	85.3
	EA_Dynamic^0	4.80 ± 0.07	83.0
	EA_SKO^0	4.93 ± 0.06	85.3
c199	variable	5.09 ± 0.06	100.0
	EA_LRS^0	4.27 ± 0.06	83.9
	EA_RWS50^0	4.33 ± 0.05	85.1
	EA_Dynamic^0	4.32 ± 0.06	84.9
	EA_SKO^0	4.32 ± 0.06	84.9

Note. For each case, the mean, standard error, and the relative performance in percent with respect to `variable` strategy, are reported.

Table 10: Relative average distance of detour to integrate an additional customer, compared with `variable` strategy

Waiting strategy	Relative performance (%)
variable	100
EA_LRS^0	93.6
EA_RWS50^0	93.9
EA_Dynamic^0	91.7
EA_SKO^0	93.1

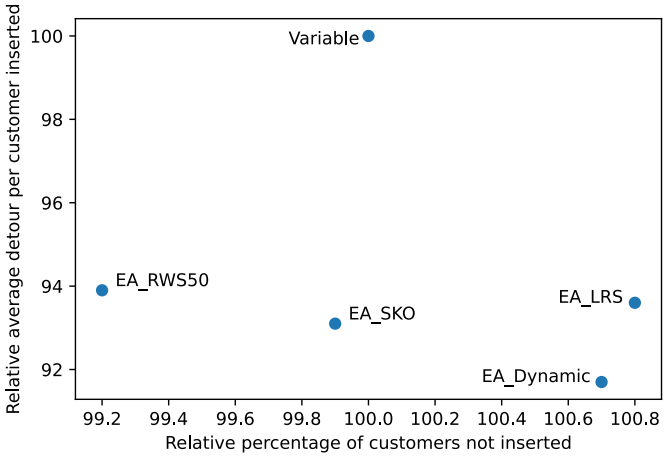

Figure 13: Comparison of the heuristics with respect to the average number of customers not inserted and the average length of the detour for a successful insertion of a new customer

Note. The EAs do not use any form of elitism.

6 Conclusion

This thesis considered a form of the dynamic VRP, where one additional customer request appears over time, which can be rejected or accepted. Based on the literature, waiting strategies were used to optimize the probability of feasibly inserting this additional customer. Waiting strategies have the potential of decreasing the detour necessary for insertion and in that way increase the likelihood of insertion. The research question of this thesis is: "What is the best waiting strategy for maximizing the probability of inserting one additional customer in the existing route schedule?"

To answer this question, first six different static waiting strategies and two complex strategies resulting from EAs were compared. Afterward, the research focused on improving the EAs by first increasing the number of customers used for fitness evaluation. Furthermore, one dynamic and five static selection methods were compared. Lastly, a final comparison was simulated to conclude the research question.

The first results showed that waiting strategies can significantly improve the insertion probability of an additional customer, compared to not waiting at all. Furthermore, the `variable` strategy resulted in the lowest number of customers rejected. This strategy lets vehicle slow their speed when their time left to the depot is equal to their slack time. As expected, further results showed that using a larger number of test customers for performance evaluation in the EA increased the insertion performance of an additional customer. However, the results also indicated a significant increase in computational time for the algorithm associated with the increase in the number of customers used in the test set. Moreover, the performance advantage compared to the increase in computational time seems to decrease when the test set size grows. Finally, it was expected that the dynamic selection method would outperform the static selection methods. Nevertheless, the implementation used in this thesis did not provide such results. The results showed that two out

of the six methods were significantly worse-performing than the other selection methods. However, from the four methods remaining, no clear conclusion could be made.

Lastly, the four selection methods based on a larger test set than initially were compared to the best performing simple waiting strategy, the `variable` strategy. The results of this comparison were inconsistent over the multiple problem instances. Considering the average relative performance, the results showed that the EAs had a significantly lower average detour, compared to the `variable` strategy. Moreover, the newly introduced `EA_RWS50`^{0} strategy seemed to attain the highest insertion probability of one additional customer.

All in all, it can be concluded that introducing waiting strategies can both decrease the average detour necessary and increase the probability of inserting one additional customer in existing routes. From the simple strategies introduced the `variable` strategy achieved the highest insertion probability. Moreover, it is established that the performance of the EA introduced by Branke et al. (2005) can be significantly improved by increasing the size of the test set used for fitness evaluation and using other selection methods. However, the updated EAs found in this thesis do not yet consistently outperform the `variable` waiting strategy.

7 Discussion

Even though this thesis renders interesting insights the limitations and questions left for future research should be noted. Firstly, this thesis assumes that the distance between locations is equal to the Euclidean distance and the speed of a vehicle is equal to one unit measure of distance over one unit measure of time. Furthermore, there is assumed that only one additional customer appears over time and the additional customer requests are assumed to be uniformly distributed over time and uniformly distributed over a predefined plane. Since the evaluation of waiting for strategies in a dynamic VRP is still relatively undiscovered, these assumptions make it possible to start evaluating different techniques in a general context. However, as the research suggests that large improvements can be made, it would be interesting to apply the techniques to more realistic assumptions. Hence, testing the techniques on practical applications instead of theoretical routes can emphasize the relevance of waiting strategies in practice.

Furthermore, there exist more specific limitations to the research presented. Firstly, the parameters of the normal distribution used for mutations in the EA are based on reasoning instead of a formal analysis. Appendix A provides a detailed motivation of the intuition applied, however, the intuition is based on assumptions. Furthermore, with the evaluation of the binary TOS selection method, it was noticed that the genetic diversity converged to 0. This implied that the mutations might be too small to have an impact on the child to discover new areas in the search space. Hence, a formal analysis exploring different distributions or different mutation methods might further improve the EA.

Additionally, the performance of the relatively undiscovered dynamic selection method was assessed and compared to static methods. It was noted that the dynamic method did not perform significantly differently from some of the static methods. Hence, it is concluded that dynamically

selecting the method via the specific implementation used in this thesis, does not provide any advances related to the performance compared to static methods. Nonetheless, it is relevant to note that only one implementation of the dynamic selection method is considered. Thus, for future research, it would be interesting to examine multiple quality and genetic diversity measures. Moreover, the RWS50 and SKO methods could be further explored to examine whether different parameters would improve their results.

Lastly, the evaluation method of this thesis can be improved by increasing the number of simulations used to report the mean and standard error. All results in this thesis are computed over 20 simulations. If this number is increased the values of the means and standard errors are more robust and hence will provide insights of better quality. Nevertheless, as the computational time is large for some instances, this is an important disadvantage of this suggestion.

In conclusion, the outcomes of this thesis provide relevant insights into the societal and scientific field of Operations Research. However, as the research area is still relatively undiscovered, there are many directions left for future research. The two most essential directions are further developing the EA to find better results, and relaxing or adjusting assumptions to provide results that are applicable to more realistic settings.

References

Bäck, T., Hoffmeister, F., and Schwefel, H. P. (1991). A survey of evolution strategies. *Proceedings of the fourth international conference on genetic algorithms*, (3).

Beasley, J. E. (1990). OR-Library: distributing test problems by electronic mail. *Journal of the operational research society*, 41(11):1069–1072.

Berbeglia, G., Cordeau, J. F., and Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15.

Bertsimas, D. J. and Simchi-Levi, D. (1996). A new generation of vehicle routing research. *Operations Research*, 44(2):286–304.

Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers and Industrial Engineering*, 99:300–313.

Branke, J. and Mattfeld, D. C. (2005). Anticipation and flexibility in dynamic scheduling. *International Journal of Production Research*, 43(15):3103–3129.

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

Chu, P. C. and Beasley, J. E. (1997). A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, 24(1):17–23.

Crepinsek, M., Liu, S. H., and Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys*, 45(3).

Dantzig, G. B. and Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, 6(1):80–91.

De Jong, K. A. (1975). Analysis of the Behavior of a Class of Genetic Adaptive Systems. *Technical Report No. 185, Department of Computer and Communication Sciences, Univeristy of Michigan.*

Escudero, R., Alves, F., Ehmler, M., Goel, K., and Filonov, V. (2020). *Can Delivery Companies Keep Up With the E-Commerce Boom?* retrieved from = https://www.bcg.com/publications/2020/can-delivery-companies-keep-up-with-the-ecommerce-boom.

Fogel, L. J., Owens, A. J., and Walsch, M. J. (1966). Artificial Intelligence Through Simmulated Evolution. *Proc. of the 2nd Cybernetics Science Symp.*

Gendreau, M., Guertin, F., Potvin, J. Y., and Séguin, R. (2006). Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C: Emerging Technologies*, 14(3):157–174.

Gendreau, M. and Potvin, J.-Y. (1998). Dynamic Vehicle Routing and Dispatching. *Fleet Management and Logistics*, pages 115–126.

Hasançebi, O. and Erbatur, F. (2000). Evaluation of crossover techniques in genetic algorithm based optimum structural design. *Computers and Structures*, 78(1):435–448.

Hoffmeister, F. and Bäck, T. (1991). Genetic algorithms and evolution strategies: Similarities and differences. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 496 LNCS:455–469.

Holland, J. (1992). Genetic Algorithms. *Scientific American*, 267(1):66–73.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. MIT Press.

Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2000). Diversion Issues in Real-Time Vehicle Dispatching. *Transportation Science*, pages 426–438.

Jebari, K. and Madiafi, M. (2013). Selection Methods for Genetic Algorithms. *International Journal of Emerging Sciences*, 3(4):333–344.

Jünger, M., Reinelt, G., and Rinaldi, G. (1995). Chapter 4 The traveling salesman problem. *Handbooks in Operations Research and Management Science*, 7(C):225–330.

Koza, J. R. (1992). *Genetic Programming: on the programming of Computers by means of natural Selection*. MIT Press.

Laporte, G. and Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. *North-Holland Mathematics Studies*, 132:147–184.

Michalewicz, Z. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32.

Mitrović-Minić, S., Krishnamurti, R., and Laporte, G. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(8):669–685.

Mitrović-Minić, S. and Laporte, G. (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(7):635–655.

Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11.

Potvin, J. Y. (2009). State-of-the art review:Evolutionary algorithms for vehicle routing. *INFORMS Journal on Computing*, 21(4):518–548.

Psaraftis, H. (1995). Dynamic vehicle routing problems. *Annals of Operations Research*, 61:143–164.

Razali, N. M. and Geraghty, J. (2011). Genetic algorithm performance with different selection strategies in solving TSP. *Proceedings of the World Congress on Engineering 2011, WCE 2011*, 2:1134–1139.

Rechenberg, I. (1973). Evolutionasstrategie - optimierung technischer systeme nach prinzipen der biologishen evolution.

Ribeiro Filho, J. L. and Treleavan, P. C. and Alippi, C (1994). Genetic Algorithm Programming Environments . *Computer*, 27(6):28–43.

Sarker, R., Kamruzzaman, J., and Newton, C. (2003). Evolutionary optimization (EvOpt) : A brief review and analysis . *International Journal of Computational Intelligence and Applications*, 03(04):311–330.

Toth, P. and Vigo, D. (2002). *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics.

Ulder, N. L. J., Aarts, E. H. L., Bandelt, H.-J., Laarhoven, P. J. M. V., and Pesch, E. (1990). Genetic Local Search Algorithms for the Traveling Salesman Problem. *International Conference on Parallel Problem Solving form Nature*, pages 109–116.

Wang, J. Q., Tong, X. N., and Li, Z. M. (2007). An improved evolutionary algorithm for dynamic vehicle routing problem with time windows. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4490 LNCS(PART 4):1147–1154.

Wilson, N. H. M. and Colvin, N. J. (1977). *Computer control of the Rochester dial-a-ride system*, volume 77. Massachusetts Institute of Technology, Center for Transportation Studies.

Zeimpekis, V., Tarantilis, C. D., Giaglis, G. M., and Minis, I. (2007). *Dynamic fleet management: Concepts, systems, algorithms case studies*, volume 38. Springer.

Zhang, H., Ge, H., Yang, J., and Tong, Y. (2021). Review of Vehicle Routing Problems: Models, Classification and Solving Algorithms. *Archives of Computational Methods in Engineering*, (0123456789).

Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. Ithaca: Shaker.

Appendix A Motivation for the standard deviation of the normal distribution used for mutations

As mentioned the normal distribution has a mean equal to 0 and a standard deviation equal to $\frac{1}{6(n_k+2)}$, where n_k is the total number of customers that are included in route k. This specific distribution is chosen so that the mutation is relatively small compared to the original value so that the child still correlates with the parents. If the mutation would be relatively large compared to the original values of the waiting strategy, the effect of the selection and recombination methods would be diluted. Therefore, the mean is chosen equal to 0 and we aim for a maximum mutation equal to $\frac{1}{2(n_k+2)}$.

Consider the case that route k would serve 8 customers, so $n_k = 8$. Then, the route would have a total of $n_k + 2 = 10$ locations, since the depot is also added at the beginning and the end of the route. This means that if the total slack time would be divided over all locations, each location will be assigned a fraction of $\frac{1}{n_k+2}(= 0.1)$ of the slack time as waiting time. As this equal division of the slack time is not the case in general and the goal is to avoid obtaining negative numbers after the mutation, the value $\frac{1}{n_k+2}$ is divided by 2. Hence, the maximum deviation is desired to be equal to $\frac{1}{2(n_k+2)}$. In order to obtain this result, the empirical rule stating that for the normal distribution almost all data, approximately 99.7%, will fall between three standard deviations, is used. Thus, the standard deviation is set to the maximum deviation desired divided by 3. Hence the standard deviation is equal to $\frac{1}{6(n_k+2)}(= \frac{1}{60})$.

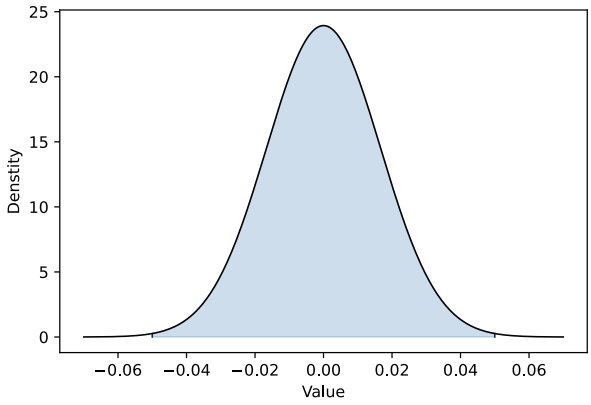

Figure 14: The normal distribution with mean equal to 0 and standard deviation equal to $\frac{1}{60}$, with the shaded area equal to 99.7% of the data points.

From Figure 14 we can conclude that if we use this distribution, indeed almost all data falls between the maximum deviation desired (0.05). To conclude, if the waiting strategy is to divide the slack time equally over all locations in a route, then using the normal distribution discussed there is approximately no chance of obtaining a negative number after adding the mutation. Moreover, if $n_k = 8$, then 99.7% of all proportional waiting times will be on the interval $[0.05, 0.15]$ and 68% of the proportional waiting times will be on the interval $[0.08, 0.12]$.

Appendix B Detailed results of comparing different sizes of test sets of customers in the fitness evaluation

Table 11: Computational time of the EA for different sizes of the group of customers used for fitness evaluation

Problem	Customers used for evaluation	Computational time (ms)	Relative time (%)
c50	100	50087.0 ± 951.4	100.0
	500	64199.2 ± 984.6	128.2
	1000	82822.3 ± 2213.8	165.4
	1500	87898.4 ± 190.9	175.5
c75	100	123980.5 ± 148.6	100.0
	500	154846.2 ± 264.0	124.9
	1000	193015.8 ± 147.2	155.7
	1500	231503.6 ± 251.9	186.7
c100	100	128629.2 ± 224.8	100.0
	500	158851.0 ± 184.5	123.5
	1000	198622.6 ± 223.7	154.4
	1500	240378.4 ± 863.6	186.9
c100b	100	148633.5 ± 639.2	100.0
	500	184637.6 ± 237.8	124.2
	1000	234580.0 ± 311.0	157.8
	1500	280267.3 ± 252.7	188.6
c120	100	131255.4 ± 245.4	100.0
	500	163155.8 ± 276.9	124.3
	1000	203318.4 ± 261.1	154.9
	1500	243977.6 ± 270.8	185.9
c150	100	270677.7 ± 6125.5	100.0
	500	333723.1 ± 1454.4	123.3
	1000	488990.5 ± 8443.2	180.7
	1500	504913.5 ± 3878.4	186.5
c199	100	490457.4 ± 11482.5	100.0
	500	689841.3 ± 13328.3	140.7
	1000	759960.8 ± 3268.4	154.9
	1500	883612.5 ± 262.8	180.2

Note. For each case, the mean and the standard error, and the relative computational time compared to 100 customers, are reported.
Note. The computational time is measured in milliseconds (ms).

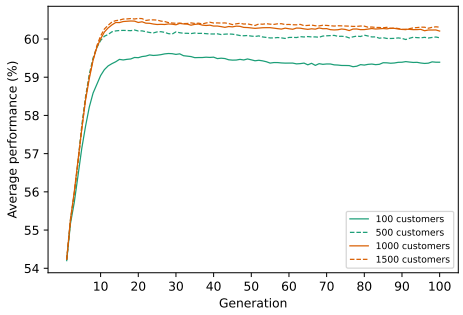

Figure 15: Average probability of insertion over generations for different sizes of customers used for the fitness evaluation for problem c50.

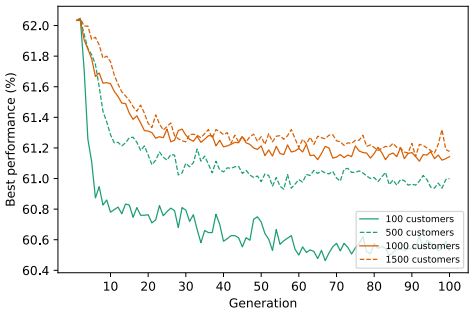

Figure 16: Highest insertion probability over generations for different sizes of customers used for the fitness evaluation for problem c50.

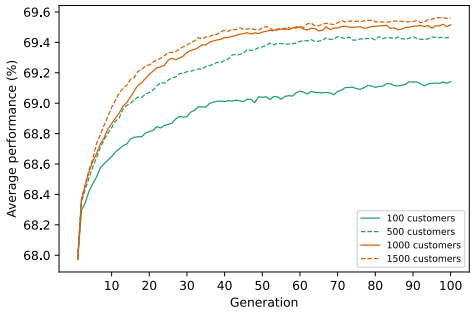

Figure 17: Average probability of insertion over generations for different sizes of customers used for the fitness evaluation for problem c75.

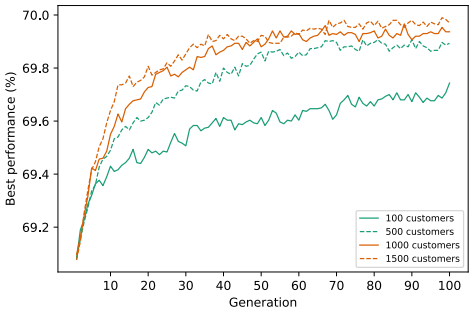

Figure 18: Highest insertion probability over generations for different sizes of customers used for the fitness evaluation for problem c75.

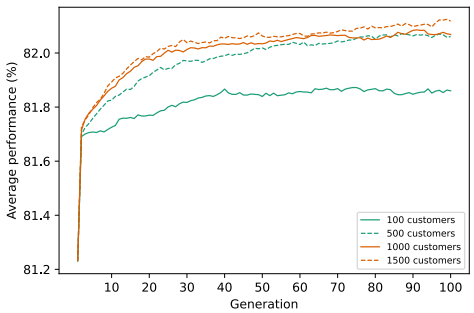

Figure 19: Average probability of insertion over generations for different sizes of customers used for the fitness evaluation for problem c100.

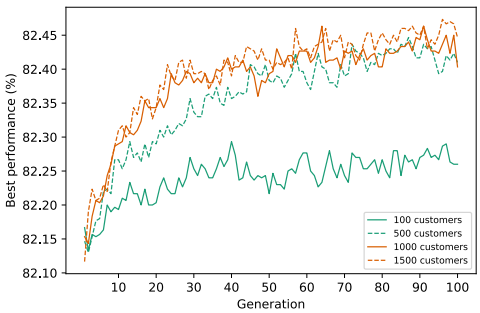

Figure 20: Highest insertion probability over generations for different sizes of customers used for the fitness evaluation for problem c100.

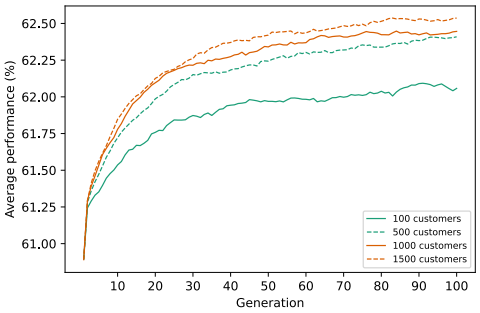

Figure 21: Average probability of insertion over generations for different sizes of customers used for the fitness evaluation for problem c100b.

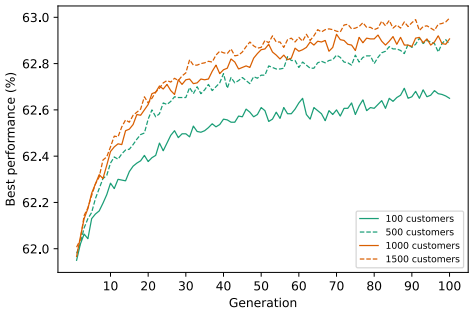

Figure 22: Highest insertion probability over generations for different sizes of customers used for the fitness evaluation for problem c100b.

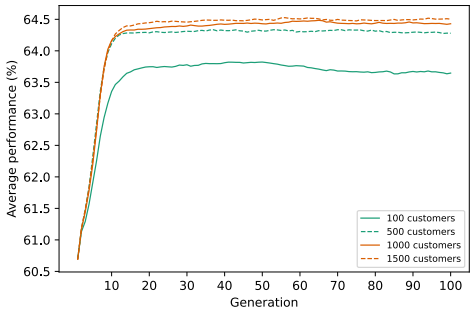

Figure 23: Average probability of insertion over generations for different sizes of customers used for the fitness evaluation for problem c120.

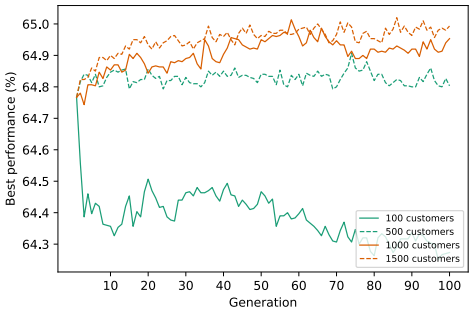

Figure 24: Highest insertion probability over generations for different sizes of customers used for the fitness evaluation for problem c120.

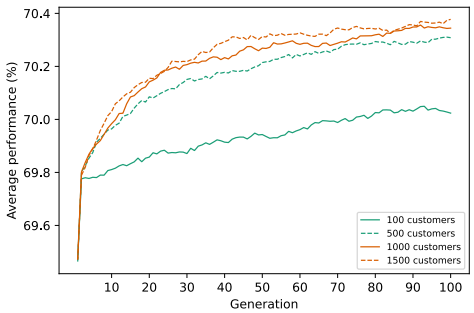

Figure 25: Average probability of insertion over generations for different sizes of customers used for the fitness evaluation for problem c150.

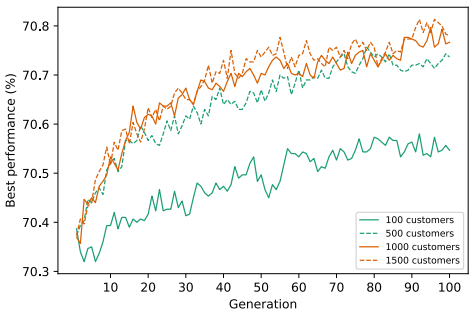

Figure 26: Highest insertion probability over generations for different sizes of customers used for the fitness evaluation for problem c150.

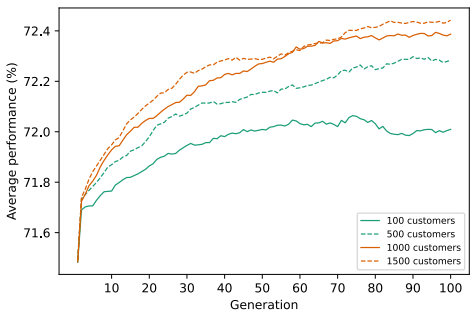

Figure 27: Average probability of insertion over generations for different sizes of customers used for the fitness evaluation for problem c199.

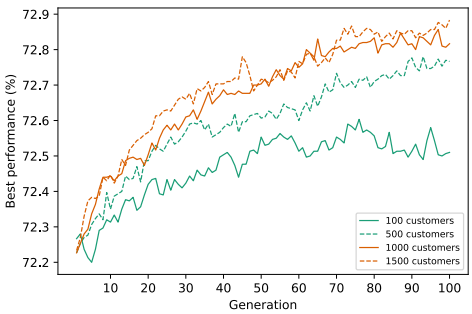

Figure 28: Highest insertion probability over generations for different sizes of customers used for the fitness evaluation for problem c199.

Appendix C Detailed results of comparing different selection methods

Table 12: Computational time of the EA for different selection methods

Problem	Selection method	Computational time (ms)	Relative time (%)
c50	LRS	48275.2 ± 726.3	100.0
	RWS	47459.2 ± 753.2	98.3
	TOS	45581.5 ± 103.4	94.4
	RWS50	45320.7 ± 150.4	93.9
	SKO	319552.4 ± 5415.5	661.9
	Dynamic	366979.1 ± 809.7	760.2
c75	LRS	117137.1 ± 233.1	100.0
	RWS	117500.0 ± 399.1	100.3
	TOS	117100.2 ± 247.7	100.0
	RWS50	116790.7 ± 284.8	99.7
	SKO	764550.4 ± 13623.9	652.7
	Dynamic	930745.2 ± 474.8	795.4
c100	LRS	124060.4 ± 889.0	100.0
	RWS	124939.1 ± 297.9	100.7
	TOS	123744.8 ± 296.1	99.7
	RWS50	124515.2 ± 258.4	100.4
	SKO	782178.5 ± 4943.7	630.5
	Dynamic	1169852.0 ± 255.4	943.0
c100b	LRS	140186.8 ± 117.0	100.0
	RWS	140999.9 ± 227.1	100.6
	TOS	147207.9 ± 923.2	105.0
	RWS50	149270.0 ± 665.5	106.5
	SKO	960912.5 ± 8308.5	685.5
	Dynamic	1165562.9 ± 695.7	831.4
c120	LRS	128384.6 ± 635.1	100.0
	RWS	129819.5 ± 198.3	101.1
	TOS	130345.2 ± 168.0	101.5
	RWS50	129907.0 ± 180.4	101.2
	SKO	839569.9 ± 1516.6	653.9
	Dynamic	1243580.5 ± 1872.5	968.6
c150	LRS	250691.9 ± 148.7	100.0
	RWS	250710.8 ± 154.2	100.0
	TOS	250984.9 ± 238.2	100.1
	RWS50	250971.8 ± 137.8	100.1
	SKO	1687069.1 ± 63138.7	673.0
	Dynamic	2461990.2 ± 9153.5	982.1
c199	LRS	444063.1 ± 171.6	100.0
	RWS	466596.4 ± 6882.2	105.1
	TOS	475163.6 ± 2314.6	107.0
	RWS50	456491.3 ± 1907.9	102.8
	SKO	2850463.6 ± 2409.5	641.9
	Dynamic	4333081.0 ± 19321.4	975.8

Note. For each case, the mean and the standard error, and the relative computational time compared to the LRS method, are reported.
Note. The computational time is measured in milliseconds (ms).

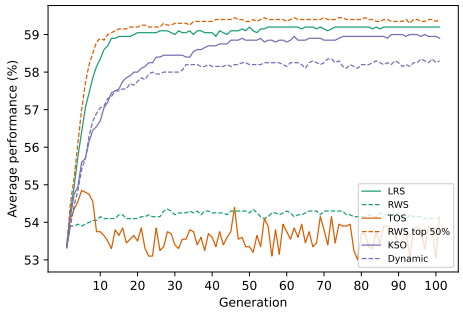

Figure 29: Average insertion probability over generations for different selection methods applied to problem c50.

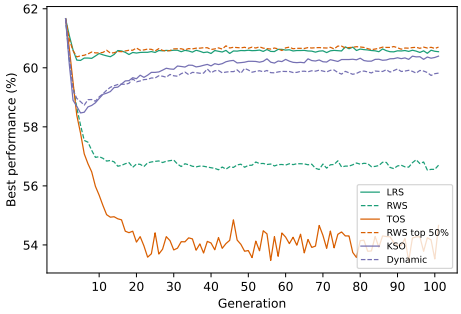

Figure 30: Highest insertion probability over generations for different selection methods applied to problem c50.

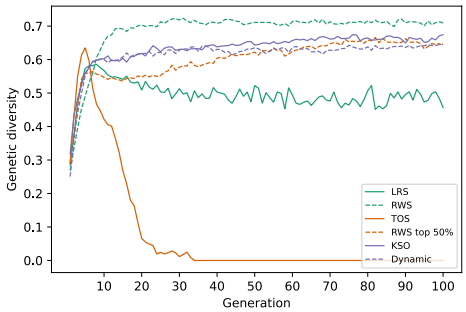

Figure 31: Genetic diversity over generations for different selection methods applied to problem c50.

Table 13: Relative frequency of selection methods chosen over 20 simulations in the dynamic selection method for c50

Selection method	**Relative frequency (%)**
LRS	0.15
RWS	0.50
TOS	0.25
RWS50	50.20
SKO	48.90

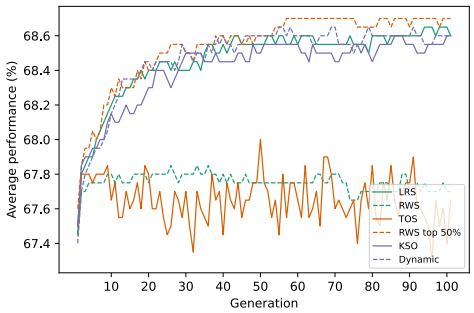

Figure 32: Average insertion probability over generations for different selection methods applied to problem c75.

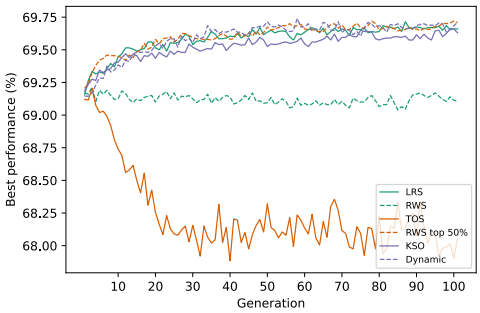

Figure 33: Highest insertion probability over generations for different selection methods applied to problem c75.

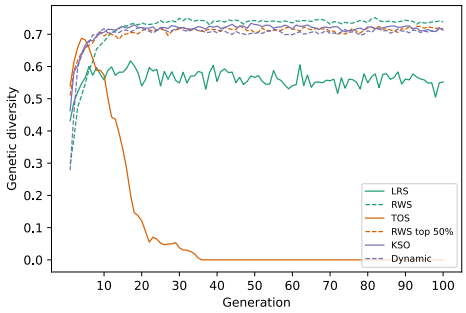

Figure 34: Genetic diversity over generations for different selection methods applied to problem c75.

Table 14: Relative frequency of selection methods chosen in the dynamic selection method for c75

Selection method	**Relative frequency (%)**
LRS	0.00
RWS	0.95
TOS	0.05
RWS50	51.55
SKO	48.45

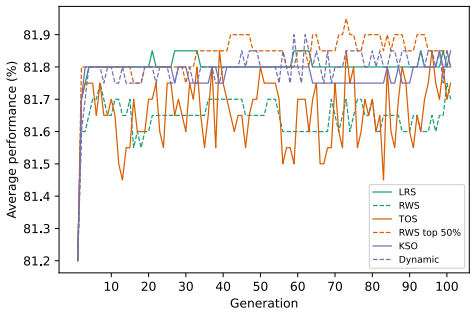

Figure 35: Average insertion probability over generations for different selection methods applied to problem c100.

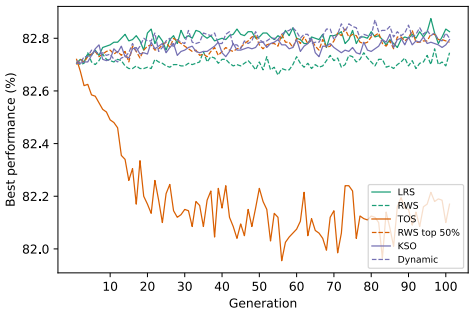

Figure 36: Highest insertion probability over generations for different selection methods applied to problem c100.

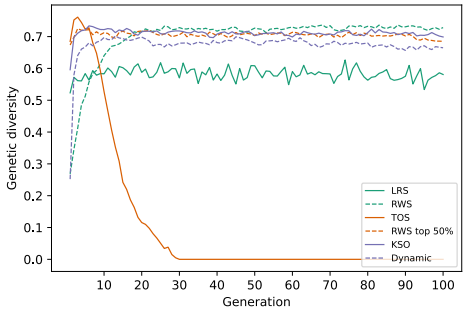

Figure 37: Genetic diversity over generations for different selection methods applied to problem c100.

Table 15: Relative frequency of selection methods chosen in the dynamic selection method for c100

Selection method	Relative frequency (%)
LRS	0.00
RWS	0.85
TOS	0.05
RWS50	47.80
SKO	51.30

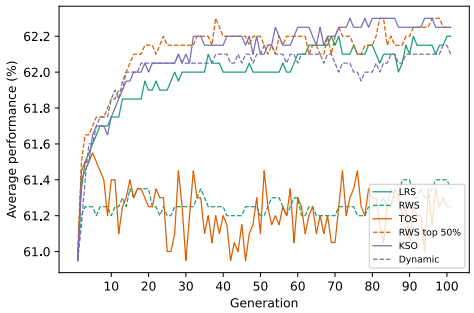

Figure 38: Average insertion probability over generations for different selection methods applied to problem c100b.

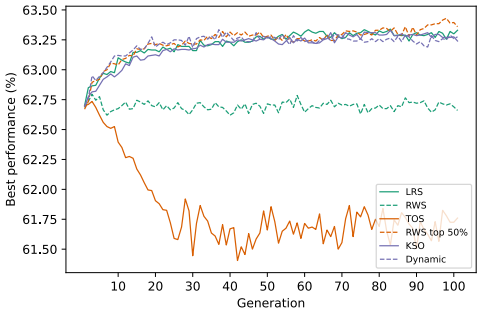

Figure 39: Highest insertion probability over generations for different selection methods applied to problem c100b.

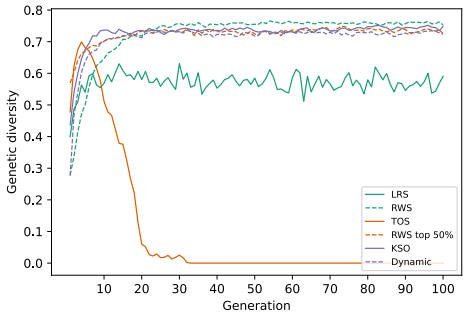

Figure 40: Genetic diversity over generations for different selection methods applied to problem c100b.

Table 16: Relative frequency of selection methods chosen in the dynamic selection method for c100b

Selection method	Relative frequency (%)
LRS	0.00
RWS	0.80
TOS	0.10
RWS50	49.75
SKO	49.35

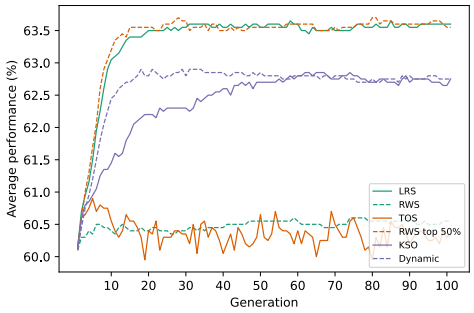

Figure 41: Average insertion probability over generations for different selection methods applied to problem c120.

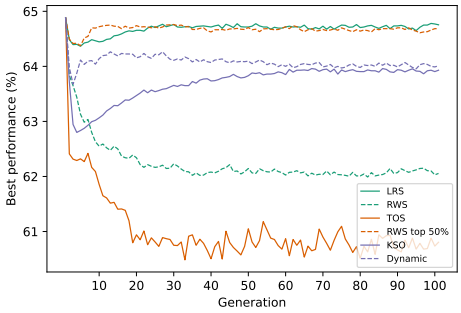

Figure 42: Highest insertion probability over generations for different selection methods applied to problem c120.

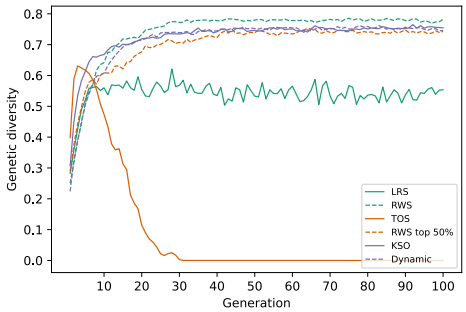

Figure 43: Genetic diversity over generations for different selection methods applied to problem c120.

Table 17: Relative frequency of selection methods chosen in the dynamic selection method for c120

Selection method	**Relative frequency (%)**
LRS	0.05
RWS	0.55
TOS	0.20
RWS50	52.15
SKO	47.05

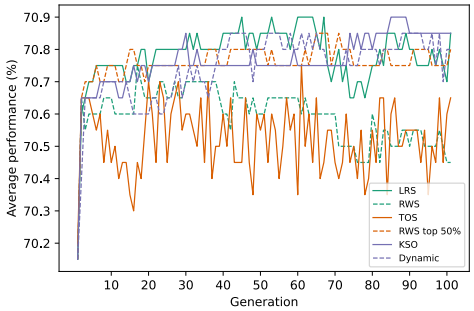

Figure 44: Average insertion probability over generations for different selection methods applied to problem c150.

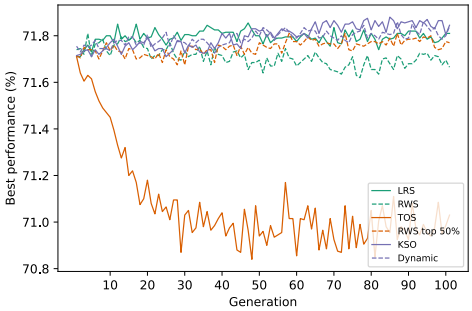

Figure 45: Highest insertion probability over generations for different selection methods applied to problem c150.

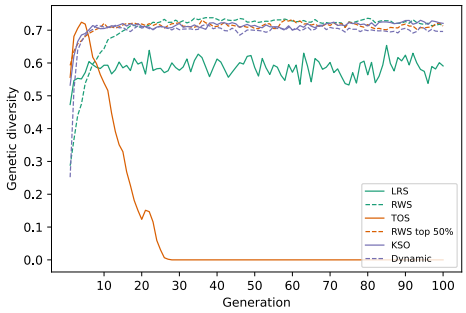

Figure 46: Genetic diversity over generations for different selection methods applied to problem c150.

Table 18: Relative frequency of selection methods chosen in the dynamic selection method for c150

Selection method	**Relative frequency (%)**
LRS	0.00
RWS	0.85
TOS	0.10
RWS50	45.30
SKO	53.75

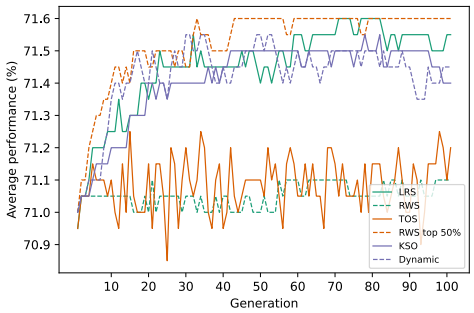

Figure 47: Average insertion probability over generations for different selection methods applied to problem c199.

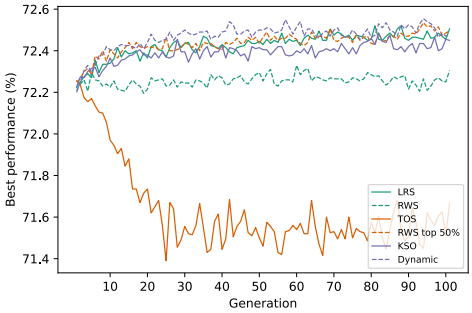

Figure 48: Highest insertion probability over generations for different selection methods applied to problem c199.

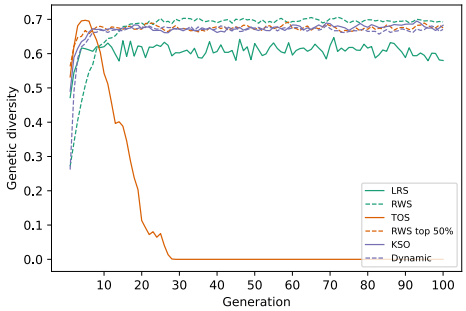

Figure 49: Genetic diversity over generations for different selection methods applied to problem c199.

Table 19: Relative frequency of selection methods chosen in the dynamic selection method for c199

Selection method	**Relative frequency (%)**
LRS	0.10
RWS	0.75
TOS	0.00
RWS50	47.05
SKO	52.10

Appendix D Short description of programming files

With this thesis belong two zip-files containing the programming code used in Eclipse to generate the results and the code used in Python to generate the figures. First, the Eclipse zip-file is shortly discussed and then the Python code is explained.

The source code of the Eclipse file contains six classes: main.java, additionalCustomer.java, customer.java, instance.java, route.java and evolutionaryAlgorithm.java. These classes and the methods in them are briefly discussed.

The main.java file first reads the problem data and creates an instance out of them. Furthermore, it contains the code to run and print all results generated in this thesis. Additionally, this class also contains several methods:

- readInstance: reads the txt file and creates an instance of the data.
- evaluateStrategy: evaluates a single waiting strategy and returns the customers not inserted and the average detour.
- simulation: evaluates a strategy by performing 20 simulations and returning an arrayList containing the performance and the detours necessary for inserting.
- simulationEA: evaluates an arrayList containing 20 different waiting strategies resulting from the EA by performing 20 simulations.
- createTestCustomers: creates a set of random additional customers.
- createRandomCustomer: creates a random customer.
- insertionFeasible: computes whether a customer can be inserted or not.
- getDetour: computes the detour between two customers.
- getDetourOn: computes the detour between a customer and a location given by coordinates.
- getMean: computes the mean of an array.
- getMeanD: computes the mean of an arrayList.
- getStdDev: computes the standard error of an array.
- getStdDevD: computes the standard error of an arrayList.
- averageEvolution: computes the average over the columns of an arrayList matrix.
- averageSelected: computes the average relative frequency of each selection method used in the dynamic decision process.
- printen: prints an array into a txt file.
- createTestSetsEvolution: creates test sets used in the EA.

The additionalCustomer.java class creates an additional customer as an Object and holds the time that it appears and the x- and y-coordinate. This class only contains get methods, that return the three characteristics. The customer.java class is very similar, but instead of having a time of appearance it has a given customer number. In this class, only get methods are implemented as well.

The instance.java class creates on Object that holds the initial existing routes from the data. Hence, to create an instance the set of customers and set of routes are necessary for the input. This class holds a methods that return characteristics of the problem instance, such as the maximum time allowed, and methods that create the simple waiting strategies.

The route.java class creates an Object containing the existing routes from the problem instance. To create a route, the ordered list of customers, the route number and the maximum time are necessary. However, the maximum time can be later adjusted with a set method. Next to some standard get and set methods the class contains methods that compute the total time necessary for the route, the slack time, the time left untill the depot from a given customer and the distance between two customers.

Finally the evolutionaryAlgorithm.java class holds all methods necessary to perform the evolutionary algorithms. To create an evolutionary algorithm the problem instance and a seed is necessary.

- EAlgorithm: used for the first two simple EAs discussed in this thesis.
- EAlgorithmEvolution: used to evaluate the different sizes of the test set used in the fitness evaluation.
- EAlgorithmSelection: used for comparing the different selection methods.
- EAlgorithmFinal: used for the final comparison.
- createRandomWaitingStrategy: creates a random waiting strategy.
- evaluateFitness, evaluateFitnessSingleStrategy, evaluateFitnessWithPerformance, evaluateFitnessWithPerformanceGiven, evaluateFitnessGiven, evaluateFitness: similar methods that compute the performance of one or more waiting strategies. The difference between the methods is what is necessary as input and how much information is returned.
- insertionFeasible: computes whether an additional customer can be inserted.
- LRS, RWS, RWS_top50, SKO, TOS: performs the given selection method.
- createChild: creates a new individual given two parents.
- computeRelPerformance: computes the relative performance of mutliple potential populations.
- getDetour: computes the detour between two customers.
- getDetourOn: computes the detour between a customer and a location.

- sortbyColumn: sorts a matrix in non-decreasing order based on a specific column.
- computeGenDivsersity: computes the genetic diversity of a population.
- euclideanDistance: computes the Euclidean distance between two arrays.

Then, there is the zip-file containing the Python code for creating the graphs. There are three different codes: fitness_evaluation.py, scatterplot.py and selection_methods.py. The different files are used for the sections that their name indicates.