MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

A penalty-based edge-assembly memetic algorithm for the vehicle routing problem with synchronization constraints in city logistics

verfasst von / submitted by

Kseniya Titova

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2018 / Vienna 2018

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 914
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Internationale Betriebswirtschaft
Betreut von / Supervisor:	Univ.-Prof. Mag. Dr. Karl Franz Dörner, Privat.doz.

Abstract

A penalty-based edge-assembly memetic algorithm for the vehicle routing problem with synchronization constraints in city logistics

In this thesis, a metaheuristic approach to solve a vehicle routing problem with synchronization constraints (VRPSC) in city logistics is presented. The algorithm generates solutions that represent decisions for goods deliveries in city centers and shopping streets whereas traffic conditions and infrastructure put time and place limits for delivery processes. The data set includes a pool of customers that receive deliveries from several carriers. The attempt to synchronize these deliveries shall decrease waiting times of customers between deliveries preserving better timing. First, literature review is presented on the VRP in city logistics with focus on synchronization. Second, we describe our algorithm with solutions to the VRPSC formulated by (Sarasola & Doerner, 2018). Most of the components of the algorithm stem from a penalty-based edge assembly memetic algorithm for the VRP with time windows proposed by (Nagata, Bräysy, & Dullaert, 2010). Experiments for parameters setting and solutions for two groups of instances generated on real-life data in the city of Linz, Austria, are presented in the last chapter.

Acknowledgements

I would not have been able to get to the point I am now and to write this thesis without the support of several individuals who I would like to explicitly thank.

First, I would like to thank the department of Production and Operations Management with International Focus and especially Prof. Karl Dörner for giving me the opportunity to write my thesis at his chair. Moreover, I would like to thank him for his valuable input and support throughout my writing of the thesis.

Secondly, I would like to express my gratitude to Briseida Sarasola for her support. Without her valuable contributions, guidance and encouragements I would not have been able to write my thesis to this extent. I would like to thank her for her support and patience. Whenever I encountered a problem, she always had an open ear and mind for me.

Additionally, I would like to thank Prof. Karl Dörner, Prof. Richart Hartl, Briseida Sarasola and Thibaut Barthélemy for introducing me to the topic. Their motivating lectures have lighted my interest and inspiration to write my thesis in this field.

Finally, I want to thank my parents, family and friends for their continuous support not only during the writing of my thesis but throughout my entire studies. They were always at my side and had an open ear and mind whenever I required their assistance or guidance.

Contents

List of Figures

List of Tables

List of Abbreviations

EAMA	**E**dge-**A**ssembly **M**emetic **A**lgorithm
EAX	**E**dge-**A**ssembly Crossover
GA	**G**enetic **A**lgorithm
GRASP	**G**reedy **R**andomized **A**daptive **S**earch **P**rocedure
LS	**L**ocal **S**earch
MA	**M**emetic **A**lgorithm
RCL	**R**estricted **C**andidate **L**ist
SA	**S**avings **A**lgorithm
TSP	**T**raveling **S**alesman **P**roblem
VRP	**V**ehicle **R**outing **P**roblem
VRPSC	**V**ehicle **R**outing **P**roblem with **S**ynchronization **C**onstraints

1. Introduction

"Continuous effort – not strength or intelligence – is the key to unlocking our potential."

– Winston Churchill

Transportation solutions in city logistics are subject to the interests of many actors such as carriers, local enterprises, public administrators and citizens. In recent years, the routing problems in city logistics have gained much interest in literature as a vivid, real-life problem. On the one hand, fast growing cities and urbanization processes contribute much into economic growth and community wealth; on the other hand, they introduce new issues to concern under conditions of limited resources. A frequently used literature definition of "**city logistic**" was given by (Taniguchi & Thompson, 2002) and is: *"the process of totally optimizing urban logistics activities by considering the social, economic, and environmental impact of urban freight movement and it provides an opportunity for the development of innovative solutions that allow to improve the quality of life in urban areas."*

According to (Kauf, 2016; Rad & Gülmez, 2017; Taniguchi & Van Der Heijden, 2000) logistics systems are crucial elements in sustainable development of a city. There is continuous need for choosing improvement strategies, therefore, we would like to describe main trends that come from literature and real-life examples concerning city logistics conceptual development and the role of vehicle routing solutions in this framework.

Due to increasing number of population in urban areas and the rapid growth of big cities nowadays, city logistics systems are gaining more and more influence on business processes (Cattaruzza, Absi, Feillet, & González-Feliu, 2017; Doerner, Huisman, & Suhl, 2014). It is emphasized by Cattaruzza et al. that transportation systems contribute to the gross domestic product not only by services they provide, but also by employment networks. (Macharis & Melo, 2011) specifies three dimensions of city sustainability: environmental, social and economic. Also, changes in these dimensions have considerable impact on urban prosperity, quality of life and competitiveness of regional industries. All three dimensions have found their reflections in operations research. (Rad & Gülmez, 2017) investigated how logistics influences these dimensions and concluded that logistic systems have a huge impact on development of local enterprises and quality of life in the city. In addition, a “Logistic Performance Index” was proposed by World Bank (Rad & Gülmez, 2017). Moreover, each of these three dimensions represents a fundament for many problem formulations to be solved by metaheuristics (Doerner et al., 2014). Thereafter, a broad class of problems called Vehicle Routing Problems (VRPs) considers finding smart solutions for sustainable development of logistics in urban area. Also Green Vehicle Routing Problems deal with air pollution and other environmental problems; the class of Vehicle Scheduling Problems serves for optimizations in public transport systems and numerous other VRP formulations confirm the need for optimization in urban goods transportation processes (Laporte, Ropke, & Vidal, 2014; Mancini, 2013).

There are numerous examples that are described in the literature showing that many city logistics actors put their efforts into finding smart solutions for their city logistics systems. Thus, (Taniguchi & Van Der Heijden, 2000) name several city logistics initiatives in this field and cooperative freight transport systems are among them. Moreover, they mentioned a real-life example of outstanding results achieved by cooperation of concurrent carriers in Kassel, Germany. This cooperation results in remarkable reduction of the quantity of trucks in the downtown and reduces transportation costs. The prerequisite for the successful outcome is a neutral carrier which collects the goods from all depots of concurrent companies and delivers them to the stores in the city center. Therefore, high level of trust between all actors in this system is mandatory.

Unfortunately, lack of trust and many other circumstances may provoke problems in establishment of such cooperative systems. The most common reason is that supplier companies affirm that they lose their privacy in terms of commercial data sharing with others or they simply struggle to keep a direct connection with their customers (stores). Therefore, the above-mentioned example of cooperative systems cannot be adapted under these circumstances. If no smart solution is sought after (no consolidation or synchronization achieved), all delivery processes result in suboptimal distribution of deliveries at each customer point during the day. Lack of optimality is explained such that: a) several concurrent supplier vehicles shall compete for the limited parking space to deliver their goods; b) customer shops must wait between deliveries. If there are longer waiting times between deliveries at one dock (customer node), it is complicated to handle the unloading of goods efficiently; especially if delivery arrives in un-predefined time, as the person who is responsible for the unloading of goods may be preoccupied with other tasks. Consequently, carrier companies have higher drivers' costs which stem from longer time on a route; and customer shops have to deal with irregular and undefined times of deliveries which may affect efficiency of sales. To find a good solution for this problem, a number of attempts to synchronize deliveries of concurrent companies can be found in the literature (Sarasola & Doerner, 2018).

In this thesis we present solutions that accord with synchronization framework particularly. In Section 2 a literature review on the problem and the algorithm are presented. After problem description in Section 33. We present our algorithm in detail in Section 44. Our computational results are presented in Section 55. Finally, conclusions are discussed in Section 6.

2. Literature Review

2.1. The Vehicle Routing Problem in City Logistics

The VRP is one of the most studied logistics problems. The VRP is usually described as an extension of the Traveling Salesman Problem (TSP). The TSP minimize the Hamiltonian to serve deliveries from one depot to several customers where each customer must be visited only once. The VRP is similar to the TSP with the difference that the number of vehicles, which start and arrive to the same depot, is more than one. The broad class of the VRP and its variants is proved to be NP-hard problems; therefore, the exact algorithms cannot be implemented for the bigger instances which are usually suited to real life. Hence, sophisticated heuristics and metaheuristics are commonly used for trying to find solutions to such problems with good quality and reasonable time (Polacek, Hartl, Doerner, & Reimann, 2004).

The most extensively examined variant of these two problems is when restrictions are set on vehicles' capacities. The fleet of vehicles can be either *homogenous* (equal capacities for all vehicles) or *heterogenous*. Since we mentioned that the VRP may include several depots, some variations of the routing problem stem from this condition. For example, a *multi-depot* VRP problem assumes that each vehicle can start from any depot and can return to any other depot at the end of the route (Polacek et al., 2004). In the VRPs distances can be *symmetric* and *asymmetric* (Laporte et al., 2014). Euclidean TSP and VRP problem assumes that the distance between each pair of nodes is the same with no relation to direction. Non-Euclidean distances are often addressed as asymmetric, meaning that the associated costs for distances between two nodes differ in relation to such conditions as direction, time of the day, truck load, etc. It was emphasized by (Glover, Gutin, Yeo, & Zverovich, 2001) that greedy construction heuristics perform worse on asymmetric graph than on symmetric. They developed a heuristic which performs better for non-Euclidean problem in comparison to commonly used insertion heuristics. In regard to the fact that definition of the objective costs is not eventless and the routing problems address a wide range of environments to work in, travel costs between pair of nodes may also change in response to congestion times or fuel consumption due to geographic landscape or car engine (Laporte et al., 2014). An example of an *asymmetric* graph conditions is the class of time-dependent VRPs where congestion times shall be respected. Real-life conditions also set another range of constraints referred to time. A broad class of VRP is adopted for solving routing problems with time windows. A time window is a period of time when the service shall be provided. Time window constraints may differ depending on type of the problem (Laporte et al., 2014).

In context of city logistics, according to (Cattaruzza et al., 2017) the VRP modeling framework is applicable for a big number of urban transportation processes such as: market and shopping centers distribution, parcel and post deliveries, construction works and public transport, waste collection (Buhrkal, Larsen, & Ropke, 2012; V. Hemmelmayr, Doerner, Hartl, & Rath, 2013), street services and many others. Therefore, city environment and logistics systems form a complex system of processes where the city has an impact on logistics actors and logistics actors influence the city environment in return.

Influence can be both positive and negative. Since the positive way of how logistics systems in urban areas contribute to development of a city has are already been mentioned, next we would like to now pay attention to the negative outcomes which arise when logistics system function within a city. (V. C. Hemmelmayr, Cordeau, & Crainic, 2012) indicate such negative influence factors of poor planning of logistics systems on the city environment as: congestions, air and noise pollution and unmet customer demands due to late deliveries.

The city environment has also its art how it may hinder urban logistics systems to some extension: streets infrastructure poorly adapted for augmenting traffic flows, improperly planned urban area infrastructures, traffic restriction due to construction works and uncontrolled relinking of traffic flows in these areas. Limitations in space and parking possibilities are especially prevalent in the downtown areas. Sometimes strict measures are implemented in order to reduce the negative impacts of excessive presence of vehicles in the cities (Cattaruzza et al., 2017).

The current situation can be described as follows: whilst carriers, on the one hand, are looking for solutions which lead to reduction of costs, distances, drivers' travel times, how to avoid congestions; city community, one the other hand, is struggling to solve problems with air and noise pollution in the urban area, late deliveries etc. These disparities of interests are the breeding ground for academic research. Finding solutions to these problems improves quality of life in cities (Cattaruzza et al., 2017)

It has become a common story for many countries that the local authorities of the cities have to react to the growing number of vehicles in the cities. By trying to minimize the harmful impact of logistics sector on social, economic and environmental needs of a city, a range of measures has become popular. Sometimes, authorities set up restrictions in access for specific type or size of vehicles to particular zones of the cities either within pre-defined time periods or even on permanent basis (Mancini, 2013). Under these conditions, carrier companies are forced to switch to smaller vehicles and this leads to augmenting number of vehicles in the cities. Therefore, a ‘carrier-friendly’ decision was found to solve these restrictions in the way that city distribution centers (CDCs) are established (Rad & Gülmez, 2017). Since transportation in larger bulks between the cities minimizes harm to the environment but is unwished within the city centers, this framework implies that the goods are transported by full-load trucks to the CDCs that are situated on site of the cities. Then, more ‘environmentally friendly’ vehicles of usually smaller size shall perform deliveries in the urban area. This routing problem is usually addressed as a two-level (two-echelon) VRP, where goods flow shall be synchronized (Cattaruzza et al., 2017). For this problem formulated as Two-Echelon VRP in city logistics, (V. C. Hemmelmayr et al., 2012) proposed ALNS.

The classical TSP and VRP problem solution is often aimed at minimizing total travel costs. In the problem we solved here it is referred only to total travel time of all vehicles as a contributor to the total costs of the solution. In the literature one can find other formulations of the total costs for a VRP. Each problem formulation may differ in the way how objective costs are calculated: distance in kilometers, in time units, fuel consumption in liters, the vehicle CO_2 emission, the wage of the driver according to the total time spend on the tour etc.

A broad class of green VRP represents problems with focus on control of fuel consumption and minimizations of CO_2 emission. Due to the variety of real life conditions, the family of *green logistics* problems is numerous. Among others, a novel bi-objective extension to the well-known TSP problem was proposed by (Grabenschweiger, Tricoire, & Doerner, 2018) for minimization of both CO_2 emissions and disturbance levels in urban area. For more interest of classifications one may refer to (Laporte et al., 2014).

2.2. Synchronization Constraints

Routing problems with synchronization constraints are widely addressed for solving real life problems and the city logistics problem in particular (Drexl, 2012). (Kauf, 2016) asserts that establishment of common networks where transportation flows within the city are consolidated will decrease transportation flows. It also mentions that the task to synchronize the transportation flows coming from concurrent carriers is also challenging. It was explained by (Drexl, 2012) that the interdependence of vehicles is the main reason for complexity of the VRPSC. The synchronization is needed when a decision shall be made for which of the concurrent vehicles consumes / delivers resources in which order. In common VRP a customer shall be visited only once, in other words, only one vehicle is needed to fulfill a customer's order (Drexl, 2012). If any changes are made in a route of one vehicle, this shall not affect other vehicles' routes. For example, if the order of customers was reversed in one route, this will bring no changes in the orders of customer visits on other routes. In classical VRP formulation no interdependence between the routes is presumed, as it is displayed in Figure 2.2.1. From this point of view the routes are *independent* from each other (Drexl, 2012).

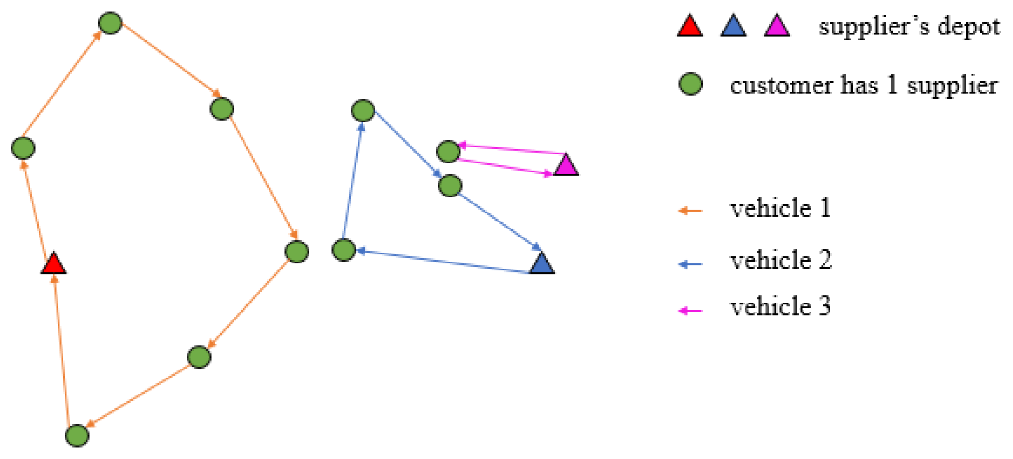


Figure 2.2.1: Independence of routes in the VRP

On the contrary, the VRP constrained with synchronization implies that routes are *interdependent* (Drexl, 2012). If changes are made in one route, this might cause a chain reaction of alterations in some other routes, or in worst case in all routes of the solution.

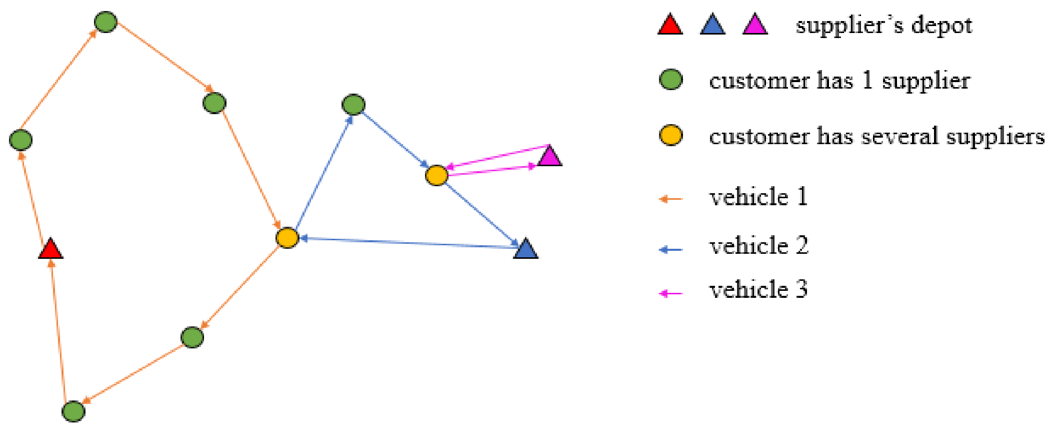


Figure 2.2.2: Interdependence of routes in the VRP

Figure **2.2.2** depicts that vehicle 1 visits 6 customers where one has placed orders by different suppliers. If it happens that vehicle 1 shall visit customers in reverse order (other than in the figure above), the time when deliveries start will differ. This will cause changes in delivery times on the route of vehicle 2 and, consequently, the same will occur on the route of vehicle 3. This chain reaction justifies that the routes are interdependent in this solution.

The synchronization constraints were classified in the literature depending on the origin or shared resources and other circumstances. Thus (Drexl, 2012; Laporte et al., 2014) define five groups of synchronization constraints that are commonly used in combinatorial optimization problems: *task*, *operation*, *movement*, *load* and *resource*.

In the problem we address in this work the operation and resource synchronization constraints are present. We induce self-imposed time windows and the delivery schedule of one vehicle depends on time of deliveries of others. Also, parking space is a resource that shall be shared among concurrent vehicles. Due to city logistics traffic and infrastructure conditions, vehicles are concurrent to each other in terms of place at the delivery dock. This implies that resource synchronization takes place: when one vehicle is performing delivery, the dock is busy and cannot accommodate any further deliveries. If concurrent supplier vehicle has arrived when other delivery is taking place, it shall wait until the ongoing delivery is finished. Only after that it may start its delivery service.

2.3. Genetic Algorithms

"It is not always the magnitude of the differences observed between species that must determine specific distinctions, but the constant preservation of those differences in reproduction."

— *Jean-Baptiste Lamarck*

Genetic algorithms (GAs) represent a vast classification of nature inspired algorithms in computational optimization. The focal idea is that, first, an initial population (chromosomes) of solutions shall be generated. Solutions are then chosen as parent chromosomes which produce offspring solutions by means of crossover techniques. Mutation operator and fitness function are referred to as instruments to keep diversity and quality of population through generations. Genetic algorithms have gained an image of powerful tool showing outstanding results in different problem solutions. In (Gendreau & Tarantilis, 2010) literature review on optimization technique provide us with numerous examples of genetic algorithm. Four criteria were chosen to define rating of algorithms: flexibility, efficiency, effectiveness and speed. Depending on variety of problem formulations, different optimization algorithms may show high competitiveness in one whereas lower in solving differently designed problem.

Memetic algorithms have the same nature as genetic ones. The difference is that in memetic algorithms a local search (LS) phase in embed. Some authors suggest adding LS at the beginning when initial population is generated to refine parent solutions before the crossover part. Others assume adding LS mechanism for improvement of the offspring results. According to (Byron & Iba, 2016; Diaz-Gomez & Hougen, 2007; Gupta & Ghafir, 2012; Morrison & Oppacher, 1998) diversity of population is a must for genetic algorithm's efficiency. Also, better quality of initial solutions might be an advantage (Diaz-Gomez & Hougen, 2007).

3. Problem Description

The VRPSC is defined on a complete graph $G = \{N, A\}$ where N is the set of nodes, E is the set of depots and, $N_0 = N \backslash \{E\}$ is the set of customer nodes, and A is the set of arcs between every pair of nodes. Each value d_{ij} represents the distance between two nodes i and j, $i,\ j \in N$. The fleet of vehicles, K, is homogeneous, each with a capacity of Q units to serve all customers, where each customer $i \in N_0$ has a non-negative order of demand q_i. The capacity of the vehicle cannot be violated:

$$\sum_{j=0}^{N} q_i * x_{ijk} \leq Q \quad \forall i,j \in N, \forall k \in K$$

Each vehicle is assigned to one depot location k_e, where $e \in E$. Every customer has a service time h_i and a self-imposed time window $[e_i, l_i]$, in which all services must start and end. If a vehicle arrives at customer node i before its current e_i, the e_i and l_i will be updated such that e_i shall equal to arrival time of the first vehicle arrived at customer i and l_i will be calculated in response to time-window length constraint (more in Section 4.1). If a vehicle would arrive after l_i at the customer node i, then the schedule would be infeasible. Each customer can handle only one delivery at a time. If more than one vehicle arrives at customer node, deliveries shall be started sequentially. The service start shall be conducted on first come first serve basis and the vehicle which arrived later shall wait until the dock becomes idle. In0 we give more information on the calculation of the self-imposed time-windows which was previously developed and adapted for test instances by (Sarasola & Doerner, 2018). More precisely, the waiting time of the dock as referred by them to as time between deliveries when the dock (customer) remains idle, also called as idle dock time. The idle dock time is the only contributor to the "bad" synchronization of current solution.

The time on route T_{max}^{day} is limited to 12 hours (work day length) and presumes that each vehicle shall return to the depot before or exactly at this point of time. No split-deliveries are allowed; hence each order shall be supplied in one visit of the corresponding vehicle. The synchronization of deliveries is meant to be met when the self-imposed time windows are not violated by any of the order deliveries. The route length time is met when all vehicles return to their depot location before or at the end of the work day.

The objective of the problem is to minimize the total travel costs:

$$\sum_{i\in N}^{n}\sum_{j\in N}^{n}\sum_{k\in K}^{m} x_{ijk} * d_{ij} \rightarrow min$$

where:

$x_{ijk} = \{0,1\}\ \forall i \neq j \in N, \forall k \in K$. If $x_{ijk} \equiv 1$, the corresponding vehicle k visits customer node i before j on a route, else $x_{ijk} \equiv 0$.

4. Solution Method

In this section we provide a more detailed description of the algorithm and insights about decisions made for implementation. Most of the components of the (Nagata et al., 2010) algorithm were used to solve the problem presented here. The algorithm components are depicted in the Figure 4.1: **EAMA** :

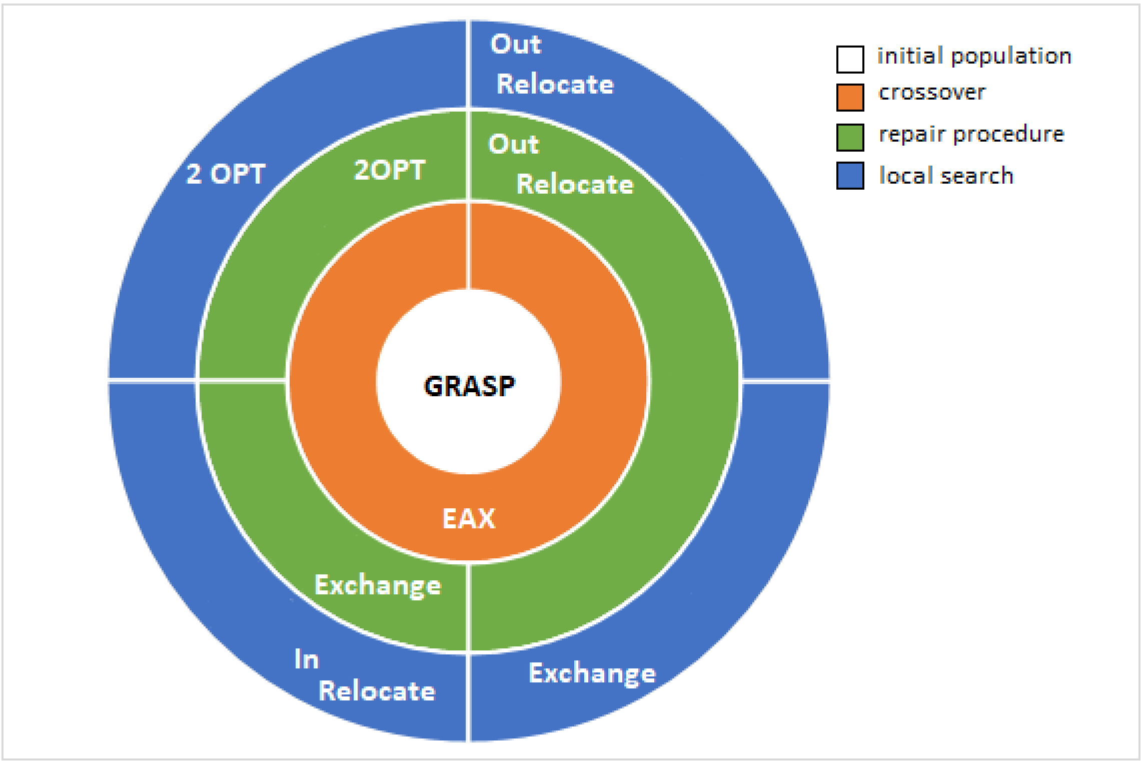


Figure 4.1: EAMA components

First, initial solutions are generated. Similar to (Nagata et al., 2010) we use a randomized construction heuristic which generates solutions which may differ in the number of routes. The crossover operator requires a pair of parent solutions that have the same number of routes. Therefore, during the construction heuristic, those solutions that fit the requirements (feasible and containing a particular number of routes) are kept for the initial population of the memetic algorithm (MA). Thus, we performed pre-run tests for a group of instances to define the number of routes as criteria for solutions to be kept for the MA or deleted from the population (Appendix B & C). Next, a crossover operator is initiated in order to produce intermediate solutions. The number of intermediate solutions to be processed for each pair of parents is also subject to adjustments since the runtime of the algorithm must be kept in reasonable time scales. When an intermediate solution is generated, a feasibility check shall be performed. If the solution is feasible it goes directly to the LS phase; if not, it undergoes a repair procedure. A selection criterion for both repair and local search operators is set to be random. Repair and LS procedures are stopped when the maximum time is reached. Depending on the size of instances, the time planned for the LS and the repair procedure was adjusted. The stopping criteria for the whole run of the EAMA was set to either the number of generations or a time limit of 10 minutes for the instances.

4.1. Self-Imposed Time Windows

We handle synchronization constraints by keeping records of both time windows and delivery schedule for each customer node. In this master thesis we use self-imposed time windows, hence the beginning and the end of time intervals when the services should take place are not predefined but calculated during the run of the algorithm. The way how self-imposed time windows are calculated was adjusted to the needs of our problem. All delivery records are directly linked with schedule records. In our case, the schedule contains the following data for each customer: arrival time of a vehicle at the customer node, service start time, time of the service end (Figure A.3: Deliveries records).

The time windows are assigned when the initial routes (containing only one order each) are built. It is explained by the fact that the arrival time of the vehicle at its initial tours represents the earliest possible arrival time to a customer. The time-window start of a customer i, denoted as e_i, is assigned with the arrival of the first vehicle. When initial tours are built, e_i is calculated as start of the working day time plus distance from the depot to the customer, and time-window end l_i refers to as:

$$l_i = e_i + (w + 1) * h_i * n_{ik}$$

where n_{ik} is the number of vehicles k currently assigned to this customer i, w is a user defined maximum time allowed, and h_i is service length time at customer node i.

Calculation of the allowed waiting time is controlled by the number of deliveries assigned to the customer, such that the more orders customer i has, the longer the allowed time-window is. When the algorithm merges two routes (ex. route A and route B), each merge reduces the number of routes by one (orders of route B will be appended at the end of route A, then route B will be deleted). This move will bring the following updates in the schedule: the vehicle which serves orders on route B will be deleted from customers schedules of this route and will be replaced by the vehicle of route A. Proving whether this merge is feasible or not, we need to recalculate schedules and, consequently, time windows of all customers that were assigned to the route B. Synchronization constraint violation consists of two elements: whether violation of the time window is at the customer i or at any of the subsequent customer nodes of this route. As a result, the time windows feasibility check shall be executed not only at customer i, but at all subsequent customer nodes of the route B. Time window violation is true at customer i if the end of the last service assigned to customer i exceeds the time window end of this customer. As soon as this statement is true for at least one customer of the route, merging of the routes will not be accepted.

4.2. Construction Heuristic

Construction heuristics for the VRP vary in different characteristics; they can be greedy or randomized. The preferences for using different construction heuristics mostly depend on the type of problem, compliance with further steps of the metaheuristics, computational time consumption and input data. A range of issues were addressed in the literature to define what makes a construction heuristic a good one for a specific problem and why. In our case, the test instances are built on the asymmetric graph, and this issue was considered when implementing the construction technique.

Another point to mention is that solutions resulting from the construction phase shall meet the needs of further steps in the algorithm. In terms of GAs, construction heuristics shall provide a well-diversified initial population of solutions. Sometimes, the feasibility of initial solutions is not a prerequisite for acceptance to participate in the genetic part of the algorithm, but in our case the initial solutions are always feasible.

The construction heuristics in this work is a GRASP metaheuristic based on the Savings Algorithm (SA). The SA was first described in 1964 by (Clarke & Wright, 1964). It is a commonly used construction heuristic for a set of the VRP problems (V. C. Hemmelmayr et al., 2012). The calculation of savings is the central part of the algorithm.

This is a heuristic method and it justifies that this algorithm is able to provide good quality solution for small size problems. In respect to the medium and big size problems (more than 15 orders) this algorithm can serve as a construction heuristic, as a pre-step for the metaheuristic phase in optimization processes. The SA has become the one that is widely used for constructing initial solutions as it is fast and easy to implement.

The saving means the reduction of costs of a solution if two tours are merged. The Figure 4.2.1, depicts how two tours are connected in a way that the saving in costs is present. Node 0 is the depot, nodes i and j are the customer nodes.

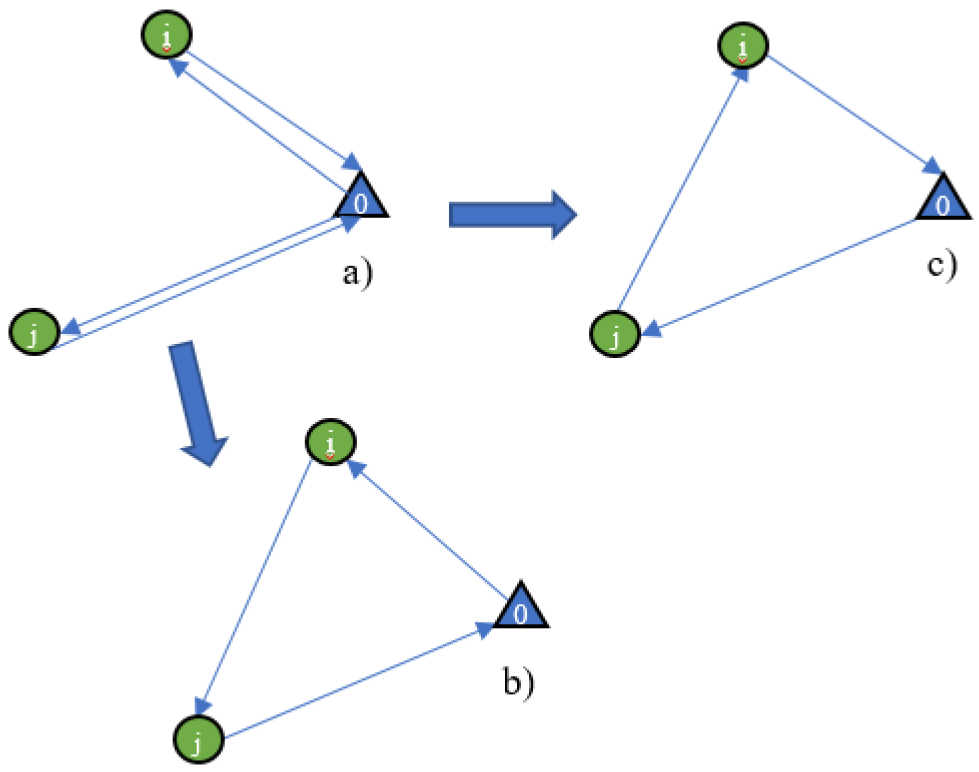


Figure 4.2.1: Merging two tours based on savings

In Figure 4.2.1 (a) there are two tours that contain one customer each: i and j. The vehicles start at the depot 0, visit the customers and return to the depot. In this case a) the customers are visited separately and it results in transportation costs (in our case distance) D_a:

$$D_a = d_{0i} + d_{i0} + d_{0j} + d_{j0}$$

Where d_{ij} is the distance costs between nodes i and j. On the figure the b) and c) illustrate that two tours are merged in different ways with respect to direction. Therefore, for the b) illustration the distance costs are calculated as:

$$D_b = d_{0i} + d_{ij} + d_{j0}$$

Next, for the c) illustration the distance costs are calculated as:

$$D_c = d_{0j} + d_{ji} + d_{i0}$$

To calculate the saving, we now use the a) and b) illustrations, and it amounts to the difference in costs between the total distance of two solutions:

$$S_{ij} = D_a - D_b = c_{i0} + c_{0j} - c_{ij}$$

The saving in merging the routes as in the c) illustration is calculated equivalently to the formula above.

The larger the saving is the more reasonable it is to merge the routes. In the savings in the b) and c) illustrated solutions will differ on the asymmetric graph.

In this thesis we apply the parallel version the SA that means more than one route is being built at a time.

It is also important to mention that since we are dealing with asymmetric distances, all savings are calculated for both variants, as if to merge $0 - i - 0$ with $0 - j - 0$ routes into either $0 - i - j - 0$ or the saving will be more attractive for $0 - j - i - 0$ route. It results into 2 savings units to be calculated and added to the savings list for the same pair of customers i, j, $\in N$.

The role of Restricted Candidate List (RCL) for GRASP is fulfilled by list of savings from Clark & Wright's Algorithm. Thus, the range of diversification can be controlled through candidate ratio, α.

$\alpha \equiv 0$	Savings algorithm
$0 < \alpha < 1$	GRASP
$\alpha \equiv 1$	Random heuristic

Table 4.2.1: Algorithm type in respect to $\boldsymbol{\alpha}$ value

Thus, some randomness has been added into the greedy SA that provides diversification of the initial solutions. First, we choose the number of instance to be read, and customer ratio α, that is to provide a random choice from savings list. Next, there is waiting time allowed to be indicated which is a crucial component for building up and updating self-imposed time windows each time an order gets assigned to a route.

The larger the idle time indicator is the longer is the time window constraint. Large waiting time indicator value results in solutions with longer dock idle times. In order to meet better synchronization, we want to minimize the total idle dock times, that means we allow relatively low level of waiting time between the deliveries (Sarasola & Doerner, 2018). Additionally, if the allowed waiting time is suspended meaning all deliveries shall take place immediately after each of the previous one, the algorithm will result in solutions with large number of routes which will harm our attempt to minimize the total transportation costs. That's why the allowed waiting time is the instrument in order to find the tradeoff between minimizing total idle dock times and total transportation costs.

Stepwise the algorithm goes as follows:

1. Distances for all elements (customer locations, depot locations) are read from the file and saved to a distance matrix. The distances are represented as travel times and are given in minutes, with three decimal positions and the time units are multiplied by 1000 for ease in calculations with integer numbers as was also done by (Sarasola & Doerner, 2018).
2. The user is asked to indicate RCL ratio value $\alpha = [0.0, 1.0]$ and the waiting time allowed indicator $w = [0, max]$.
3. Savings values are calculated and recorded in a list. Since each customer node is not linked to all depot locations, there is no need to calculate savings for arcs which will never be built. There are several depots that share same customer locations and it should be considered by building the savings list. Thus, the depot id is attached to each saving in the list.
4. Initial routes are built with only one customer order included into each route. The cost is the sum of time consumed by each vehicle to go from the depot to the corresponding customer node and back to the depot. The self-imposed time window of each customer node shall be calculated whereas the start is the time of the earliest arriving vehicle, and the time window end is set. The schedule of services is established for the vehicles according to the first come first serve rule. A vehicle can start the delivery service as soon as it arrives at the customer node and if the customer is idle, otherwise the vehicle has to wait.
5. Until the savings list is not empty pairs of routes which belong to the same depot are tried to be merged:

   i. A saving value for a pair of customer nodes u_i and u_j is picked out from the RCL;

   ii. The chosen customers must be on positions such that: a) u_i is the last customer visited on a route A; b) u_j is the first visited customer on a route B; c) $A \neq B$;

iii. If such tours are found, constraints violation is examined, else step **v.** is executed. Constraints are proved to be held if: a) tour loads do not exceed vehicles' capacity; b) time window constraints and maximum route time restrictions at each customer node of all routes are not violated;

iv. If merge is feasible, routes are merged;

v. The current saving record is deleted from the RCL.

4.3. Memetic Algorithm

4.3.1. Cross-over Operator

The edge-assembly crossover operator (EAX) was chosen because it provides good results in the literature when dealing with the TSP and the VRP. In (Nagata et al., 2010) the EAX operator was implemented to find solutions for the VRP with time windows. Because of this and since we are using self-imposed time windows, we implement the EAX for our problem. (Nagata et al., 2010) have proposed the EAX for directed graph which takes into account edge direction when building intermediate solutions. This has made the EAX applicable for time constrained problems.

The crossover procedure goes as follows. Two parent solutions are randomly chosen. A G set is the set of edges that is built in such a way that it contains only unique edges from parent A and parent B, and no similar edges from these two solutions. In our case we change the vector solution representation of the parent solutions into the edge representation[1], and add all edges of parent A into the G set. Next, we choose each edge of parent B and add them to the G set if such an edge is not present yet in the G set. If it happens that the edge already exists in the G set, we don't add it and we delete the similar one found from the G set. As a result, the G set will contain only unique edges from both parents. Important to mention that during this step the two edges are defined as similar ones if and only if both the nodes and the direction are the same (Nagata, 2006a, 2006b).

The G set is then used in order to build the AB cycles which will form an E set, as described in (Nagata et al., 2010). The AB cycles are formed by iteratively adding edges from different parent solutions which share either an arrival or a departure node.

As soon as the AB cycles are built, the E set can be formed. Based on the E set an intermediate solution will be generated. The E set is a set of AB cycles that are chosen to build an intermediate solution. During the single EAX strategy, only one AB cycle is chosen randomly to represent the E set, whereas in block strategy more than one AB cycles form the E set (Nagata, 2006a, 2006b). (Nagata et al., 2010) use the single strategy at the beginning of the genetic part. After a predefined number of iterations the algorithm switches to the block strategy. The authors explain it by the fact that after a certain number of iterations the population

[1] It was decided to use the edge-like solution representation for the EAX phase for ease of implementation.

achieves uniformity. If the single strategy is applied, offspring solutions have much in common with parent *A* (Nagata et al., 2010). Because of this, diversity of population can be harmed after a while, resulting in solutions similar to each other. To prevent this, the block strategy is accepted (Nagata et al., 2010). Thus, the block strategy works as a more robust diversifier to generate intermediate solutions and consequently guard diversity in the population. As soon as the intermediate solution is built, we translate the edge solution representation into the vector in order to start the repair procedure and the LS thereafter.

4.3.2. Repair Procedure

As proposed by (Nagata et al., 2010), the repair procedure consists of three neighborhood operators: *2-Opt*, *Out Relocate* and *Exchange* (see Section 4.5). These operators restore feasibility of the intermediate solutions by minimizing the penalty costs. Before the start of the repair procedure, penalties are assigned onto capacity and time constraint violations in accordance to the generalized cost function. The sensitivity tests performed by (Nagata et al., 2010) showed that penalty costs ratios work best when both are set at 1.0. The demand excess of a solution is calculated as the sum of total demand excess on each route. The time windows violation is calculated as the sum of all late arrivals of the vehicles to customers together with late arrivals of the vehicles at their corresponding depots (Nagata et al., 2010). The repair operators work in accordance to the best acceptance strategy. This was explained by (Nagata et al., 2010) as the reason not to conduct too many changes to the offspring solution during the repair procedure. The minimum of difference between the offspring solution before and after the repair procedure is highly desired. Also, here only those changes that minimize penalty costs are made and the changes in the distance costs values are not considered. As soon as penalties have been assigned to the intermediate solution, the procedure continues with choosing an infeasible route. Then three neighborhood operators try to restore the route feasibility. The order in which neighborhood operators are chosen is random. In one iteration three repair operators are activated only once in random order. Once one operator is chosen, it looks for the best possible improvement on infeasible routes. The routes for repair are chosen only among the infeasible ones and in random order. If the current repair operator is unable to find any improvement, another operator is randomly chosen. If feasibility of the offspring solution has been reached, the algorithm will proceed with the LS phase, if not, the child solution will be deleted from the population. If all three randomly chosen operators weren't able to find improvement, a last attempt is made, i.e. new round of random repair procedure is set. It is justified by the fact that the following situations are often observed: first round of three randomly chosen operators result in poor capacity penalties. Such 'leftovers' would throw this solution out of the population. Therefore, it was decided to try a new strategy: to add a second round for repair phase to eliminate the 'leftover' penalties. When the second round is started, a randomly chosen operator fixes the remaining small penalties, hence, the solution is repaired and kept in the population. By processing bigger instances, the repair procedure sometimes becomes very exhausting in terms of time spend on repairing it. During experiments, adding the second round for repair procedure produced better results.

4.3.3. Local Search Procedure

Local search implies that there is a space, i.e. neighborhood, which shall be exploited by a search technique. The definition of a neighborhood of a solution is presented by (Neri, Cotta, & Moscato, 2012) and is a set of solutions which result from applying one transaction on all corresponding elements of a solution. Typically, a LS mechanism starts with one solution and applies a single pre-defined move triggered by a neighborhood operator to generate new solutions. The aim is to find one that shows a better objective value. These solutions are compared to the initial solution and if the new neighbor is better it is accepted by replacing the initial solution. Otherwise the initial solution is kept. The LS procedure is executed while the stopping criteria are not met. Usually, such criteria are the predefined number of iterations when no improvements were found or the time limit. A general description of LS procedure was demonstrated by (Gendreau & Potvin, 2010) and is presented below:

Algorithm 1 A local search algorithm

```
Procedure Local-Search-Engine (current);
begin
    repeat
        new ← GenerateNeighbor(current);
        if $F_g(new) \prec_{\mathcal{F}} F_g(current)$ then
            current ← new;
        endif
    until TerminationCriterion() ;
    return current;
end
```

Figure 4.3.3.1: LS algorithm[2]

There are two widely used techniques to implement neighborhood operators: the first acceptance and the best acceptance strategy (Bräysy & Gendreau, 2005). The common feature for both strategies is that the better solution replaces the worse one. What differs is which solutions shall be compared. The first acceptance implies that at each iteration the current neighbor solution is compared to the initial solution which gets immediately replaced if the current neighbor proves to be better. If replacement takes place, the next iteration's transaction will involve the newly accepted better solution. In contrast, the best acceptance strategy works under the condition that the LS procedure runs over all possible neighbors of the initial solution, picks out the best one which shall be subsequently compared with the initial solu-

[2] (Gendreau & Potvin, 2010)

tion and replaces it if the above-mentioned condition is met. In practice, the choice of acceptance strategy relies on the computational time limitations, ease of coding or the complexity of iteration transaction (Gendreau & Potvin, 2010; Neri et al., 2012). For example, one step may be a swap of two variables or a change of a value that will lead to a chain of further calculations based on this one step. Also, justification for using one or another technique can be related to the nature of the algorithm behavior (Nagata et al., 2010).

In our case, four neighborhood operators are used: *2-Opt*, *In Relocate*, *Out Relocate* and *Exchange*, as it was done by (Nagata et al., 2010), whereas only three of these excluding *In Relocate* are used in repair procedure. It was explained in (Nagata et al., 2010) by the fact that, for repair procedure, *In Relocate* cannot lead to any improvement in terms of penalties as the operators work only on the infeasible routes. During repair procedure only, infeasible routes of intermediate solutions are chosen for conduction of transactions. Therefore, *In Relocate* operator would not be able to bring any improvements in terms of feasibility. *In Relocate* increases number of customers in a route which is not wanted when the capacity or time window violation is present.

In the Section 0, two scenarios are shown: an iterative and random LS. On the one hand, an iterative LS implies that the LS operators try to perform improving moves on all routes of the current solution until no improvement can be found or the time limit for LS is hit. On the other hand, in the random LS scenario, a route of the current solution is chosen randomly and LS operators try to find any improvements. As soon as the random LS does not discover any improvement moves, the LS procedure is restarted. For all experiments and solutions presented, the number of restarts for the random LS was set to 5. If the number of restarts is more than 5 or the time limit is hit, the random LS is stopped.

5. Computational Results

The algorithm was implemented in C++ on Intel® Core™ i7-4600M 2.9 GHz 16MB process memory, x64 precision floating point number representation. In this section, test files are presented along with the parameters setting and decisions based on iterative trials of different parameters values and the results in comparison to ALNS and CPLEX.

All results presented here refer to the test instances of (Sarasola & Doerner, 2018) downloaded from http://homepage.univie.ac.at/briseida.sarasola. The instances are based on real life data; they include distances, orders and carrier depot locations in the city of Linz, Austria. Two groups of instances were used. First group contains from 5 to 15 orders each. Second have 50, 100, 150, 200, 250 and 300 orders.

Experiments have been conducted in parameter settings for GRASP and MA, solutions are collected for two groups of instance files and the results are presented as average and the best of 5 runs of MA for the second group in comparison to ALNS. First group of instances consist of small files (5 to 15 orders) and are added in 0E . Parameters which were subject to adjustments are: a restricted candidate list ratio, g_{max} number of generations, size of P_{init} initial population, r^{init} number of routes in initial solutions, t_{init} maximum time allowed to generate initial population, size of P_{ch} offspring population for each pair of parents, t_{repair} maximum time allowed for repair procedure and t_{ls} maximum time allowed for the LS phase (both for each offspring solution respectively). Parameters which were constant: w waiting time allowed, χ and ψ penalties for constraint violation. Travel distances are presented in time units (minutes). The vehicle capacity Q is set to 18.0 demand units and service time s is 15 minutes for each order delivery. Total costs are calculated as the sum of total travel time, time spend for services is excluded.

On graphs with representation of geographical location of the customer and depot nodes, presented in Appendix DD, it is shown that the customer nodes are spread randomly across the area, forming several locations with high density of nodes. These areas represent the city center or shopping streets. The depot nodes are placed around the area in suburban, far from the city center located regions.

5.1. Parameters setting

Solution quality of genetic phase depends on numerous factors. We would like to address each of them. First, we had to think of the characteristics which are wanted and/or needed for initial population. (Nagata et al 2009) implemented a route minimization heuristic and in section where they describe results they mentioned that route minimization heuristic has a big influence on efficiency of the whole memetic algorithm as it provides solutions with possible minimum number of routes, hence relatively low total transportation costs. If good and diverse solutions are constructed, GA shall perform better. Chasing solutions with less quantity of routes is a common strategy for total costs minimization. Therefore, (Nagata et al., 2010) performed pre-run tests to define what the quantity of routes shall be in a solution so that the latter shall be used as initial population entity for the memetic part.

For these reasons we conducted the same experiment with our GRASP construction heuristics to define the number of routes needed for a solution to fulfill the 'parenting' criteria.

Determination of the minimum number of routes. The possible minimum number of routes $r^{min}$for problems with homogenous fleet is usually calculated as total sum of demands divided by vehicle capacity:

$$r^{min} = \frac{\sum_{i=0}^{n} q_i}{Q}$$

It is commonly suggested that optimal solution shall contain number of routes equal or very close to r^{min}. Thus, minimization of quantity of routes in a solution shall usually lead to total costs reduction. On Figure 5.1.1 we investigate how much the quantity of routes generated by the SA is far from r^{min} for the second group of test files (from 50 to 300 orders).

It is clear from Figure 5.1.1**: Comparison of the SA and approx. minimum of routes** that SA is not efficient enough for big and complex instances to meet the r^{min} level for all except the first file with 50 orders, where the gap is 0%. On average for this set of instances the SA generates solutions with at 108% bigger quantity of routes than the assumed r^{min}. Additionally, it shall be mentioned that r^{min} is only assumed and cannot be considered as the exact one for our problem because of asymmetric distances and time window conditions. But we can clearly see that there exists enough space for optimization in terms of distance total costs and the number of routes in genetic phase shall be minimized.

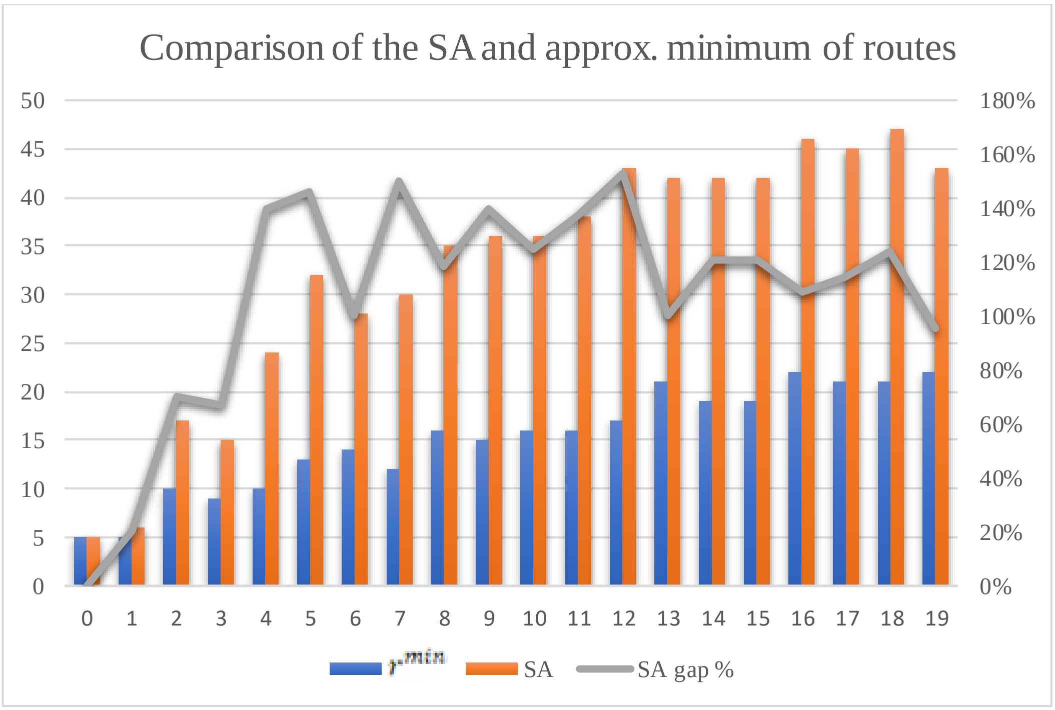


Figure 5.1.1: Comparison of the SA and approx. minimum of routes

Since in our case the GRASP was chosen as the construction heuristic, we would first like to be sure that construction phase generates better solutions than the pure SA. Second, we define a new minimum number of routes r^{init} as acceptance criterion for a GRASP solution to be added to the initial population. Third, we had to make a decision how we should collect solutions from GRASP is such a way that the obtained solutions: a) remain as much as possible in terms of objective function value better than the SA; b) contain the same number of routes equal to r^{init}; c) computational time to generate initial solutions remain reasonable. Normally, the r^{init} is set less than r^{min}. It is explained by the fact that the GRASP solutions are still far from the optimal ones in terms of both transportation costs and quantity of routes for medium and big instances. This is justified on the Figure 5.1.2**: Comparison of quantity of routes of GRASP and SA solutions** which depicts that best 10% of GRASP solutions may frequently contain less quantity of routes than the greedy SA but if we combine this with findings from the figure above it is clear that GRASP is still far from the r^{min}. Better results of GRASP in terms of r values are ensured by randomization which is allowed to a particular extension – a value. Solution costs on average are also dependent on RCL ratio a.

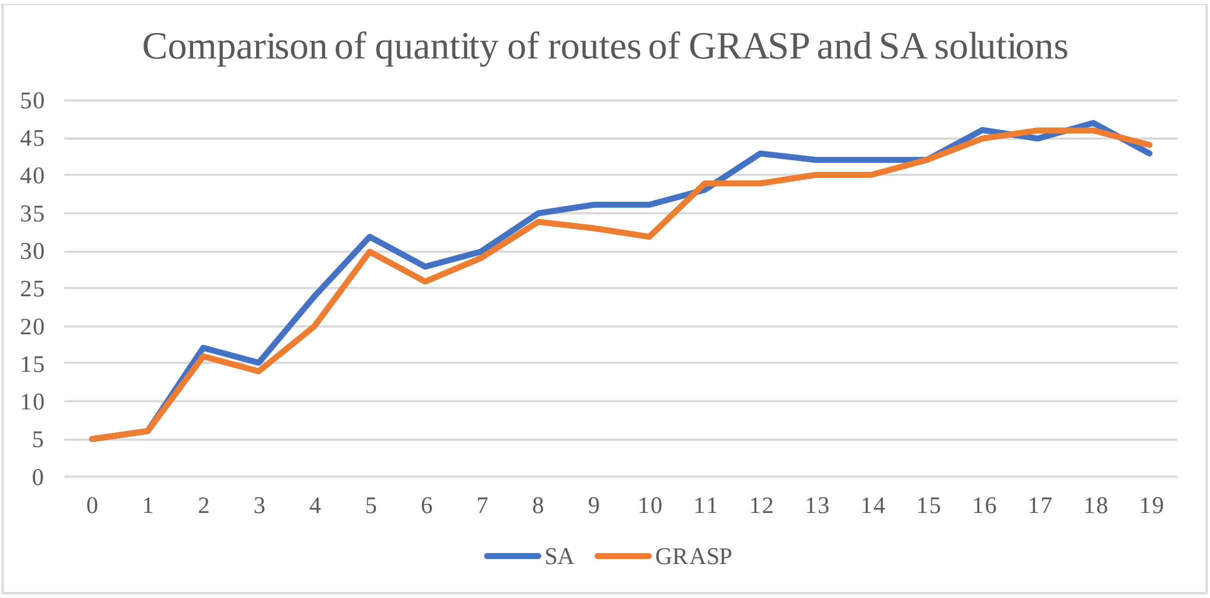


Figure 5.1.2: Comparison of quantity of routes of GRASP and SA solutions

Determination of the RCL ratio. Based on these conclusions, we decide to investigate the relationship between RCL ratio and solution quality and choose a constant new r^{min} to accumulate solutions with less routes and lower costs possible. These solutions will be the best fit to form the initial population for further MA phase. This decision process consists of several steps. For the GRASP we conduct tests with different RCL ratio (a). By choosing iteratively values of a the aim was to disclose which a gives better costs. The tests go as follows. The GRASP algorithm ran 100 times to form a group of 100 feasible solutions whereas each run was started with different value of a. Then, the best cost from each group were kept in respect to its a value. An example of the results for a file with 150 orders is as follows[3]:

RCL (a)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
0.0002	28	940 439	0	178 185	1
0.0003	24	**877 152**	0	150 335	2
0.0005	25	891 796	0	141 603	3
1	31	1 533 452	0	167 894	7428

Table 5.1.1: Solution costs by different RCL ratio values for an instance with 150 orders

This test was conducted for each instance to see which RCL ration shows better results in comparison to other a values. Table 5.1.1 shows that the lowest total costs were reached when only to best savings positions are included into the RCL. For all test instances the following was found: a) by choosing a particular RCL ratio, GRASP outperforms the greedy SA

[3] Results of all tests for medium and big instances are in (A)

in each test instance; b) solution costs become more sensitive to RCL ratio with larger instances; c) there is no fixed border on continuous horizon of a values which generate better solutions; what can be detected is an approximate a value when GRASP tends to generate better solutions more often.

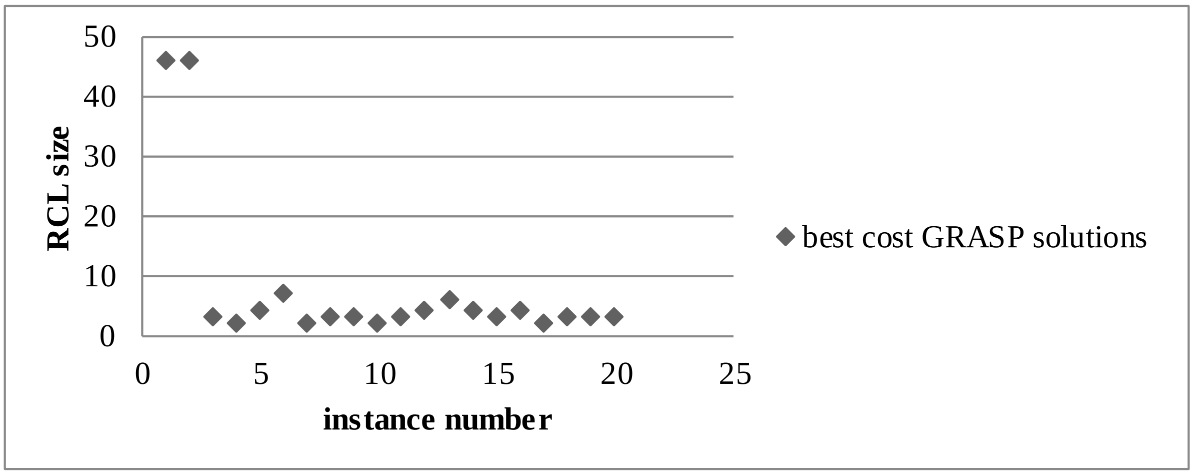


Figure 5.1.3: Best GRASP solutions in respect to RCL size

As in the Figure **5.1.3: Best GRASP solutions in respect to RCL size** we can see the results of best GRASP solutions for instance # 2 with 100 orders. It is shown that better solutions are frequently met when relatively small randomness is allowed. For example, best GRASP solutions for 18 of 20 files were achieved with such a values that RCL size was between 2 and 7 positions which is less than 0.002% of total number of savings positions. Two smallest instances 50 orders each showed to be an exception with 46 positions in RCL for best GRASP solutions.

Determination of the r^{init} for the initial population. Thereafter, the a was chosen and it was possible to proceed to the next step. First, we set GRASP to generate 100 solutions with recently found best a value. Second, we investigate the results upon four characteristics: minimum and maximum quantity of routes met, routes number most frequently met in 100 solutions and in 10% best solutions respectively.

The tests described above helped us to explore to which extend GRASP can improve the SA solutions on 20 files of different size. We were able to access which RCL ratio tends to give better results and at what level r^{init} shall be set for the step when initial population shall be generated. The parameters we chose also gave the possibility to generate initial population in the next steps in time less or equal constant t_{init} for files of 50 to 300 orders.

Summary of both tests. We investigated number of routes in the solutions and we used the best RCL ration obtained in the test above. The results in Table **5.1.2: Quantity of routes and CPU times for solutions with different a values** generalize our findings from both tests together. The time to generate one solution is less than 1 second for each of the instances.

	RCL (a)	Minimum number of routes	Maximum number of routes	Frequent number of routes	Frequent number of routes among best 10 % solutions
vrpsc-300-6-19.csv	0.00024	42	49	46	44
vrpsc-300-6-18.csv	0.00024	46	54	49	46
vrpsc-300-6-17.csv	0.00024	45	53	49	46
vrpsc-300-6-16.csv	0.00020	45	52	48	45
vrpsc-250-5-15.csv	0.00040	41	47	43	42
vrpsc-250-5-14.csv	0.00030	38	46	42	40
vrpsc-250-5-13.csv	0.00030	38	47	43	40
vrpsc-250-5-12.csv	0.00050	37	45	42	39
vrpsc-200-4-11.csv	0.00050	36	44	41	39
vrpsc-200-4-10.csv	0.00040	31	42	37	33
vrpsc-200-4-9.csv	0.00030	31	38	36	33
vrpsc-200-4-8.csv	0.00040	32	38	36	34
vrpsc-150-3-7.csv	0.00050	27	35	32	29
vrpsc-150-3-6.csv	0.00030	30	35	32	26
vrpsc-150-3-5.csv	0.00096	30	35	32	30
vrpsc-100-2-4.csv	0.00090	18	25	22	20
vrpsc-100-2-3.csv	0.00050	13	20	16	14
vrpsc-100-2-2.csv	0.00070	15	21	19	16
vrpsc-50-1-1.csv	0.01900	6	7	6	6
vrpsc-50-1-0.csv	0.01900	5	6	5	5

Table 5.1.2: Quantity of routes and CPU times for solutions with different a values

Based on these conclusions, we decided to use the frequent number of routes in best 10% solutions as the new r^{init} and the a as constant parameters for each file in further computational experiments. Since time consumption as depicted above stayed in reasonable scales, these parameters ensured ability to generate initial population for the MA in less than 3 seconds also for files with 300 orders.

Concerning further steps, 1 minute tests were conducted for all files in order to meet a decision about the remaining parameters for the MA such as: g_{max} number of generations, size of P_{init} initial population, size of P_{ch} offspring population for each pair of parents, t_{repair} maximum time allowed for repair procedure and t_{ls} maximum time allowed for the LS phase, whereas t_{init} was set to 3 seconds, a and r^{init} as in Table **5.1.2: Quantity of routes and CPU times for solutions with different *a* values**. The decisions were made by iterative trials. Generally, time for calculation of scheduling conditions and self-imposed constraints were most challenging in terms of time consumption. This resulted in the following parameters setting for the EAMA phase:

Instances groups by # of orders	**50**	**100**	**150**	**200**	**250**	**300**
g_{max}	50	25	25	25	25	25
P_{init}	25	6	6	6	6	6
P_{ch}	5	4	4	3	3	3
t_{repair}	2s	4s	5s	6s	6s	7s
t_{ls}	3s	6s	10s	10s	10s	10s

Table 5.1.3: Parameters for the EAMA

5.2. Results of the EAMA

In this section we present results of our EAMA compared to ALNS for instances with 50 to 300 orders. Comparison on small instances 5 to 14 orders are provided in comparison to ALNS and CPLEX solutions and are attached in Appendix E (Sarasola & Doerner, 2018). The EAMA algorithm was executed 5 times and the results are presented in the Tables 5.2.1 and 5.2.2. The total costs of each run of the EAMA represent only feasible solutions. For each of the instances the total average and the best costs are recorded. For each run, the best known feasible solution was recorded after 10 minutes (including the construction phase) and then the algorithm stopped.

In the previous sections we described two LS scenarios we used. The results in **Error! Reference source not found.** were generated by means of the iterative LS procedure along with the corresponding RCL ratio and the minimum number of routes for the construction phase.

Instance	Orders	ALNS	EAMA Average		EAMA Best		a	r^{init}
0	50	198 770	200822.80	1.03%	200035.00	0.64%	0.019	5
1	50	145 840	147891.20	1.41%	147294.00	1.00%	0.019	6
2	100	408 130	451564.00	10.64%	446114.00	9.31%	0.0007	16
3	100	350 090	385355.40	10.07%	382176.00	9.17%	0.0005	14
4	100	408 590	461089.20	12.85%	449393.00	9.99%	0.0009	20
5	150	564020	649223.40	15.11%	633844.00	12.38%	0.00096	30
6	150	570730	608570.80	6.63%	594825.00	4.22%	0.0003	26
7	150	501060	609517.00	21.65%	596683.00	19.08%	0.0005	29
8	200	673430	811134.80	20.45%	787674.00	16.96%	0.0004	34
9	200	698160	844387.40	20.94%	823575.00	17.96%	0.0003	33
10	200	647190	723978.20	11.86%	714737.00	10.44%	0.0004	32
11	200	650640	757230.00	16.38%	733981.00	12.81%	0.0005	39
12	250	767190	921409.00	20.10%	892473.00	16.33%	0.0005	39
13	250	861680	1033166.00	19.90%	1025850.00	19.05%	0.0004	40
14	250	778000	951851.40	22.35%	919154.00	18.14%	0.0003	40
15	250	945940	1121290.00	18.54%	1107490.00	17.08%	0.0004	42
16	300	1103970	1338148.00	21.21%	1323070.00	19.85%	0.0002	45
17	300	1115430	1310964.00	17.53%	1250050.00	12.07%	0.00024	46
18	300	965370	1178646.00	22.09%	1139910.00	18.08%	0.00024	46
19	300	1048080	1224656.00	16.85%	1185430.00	13.10%	0.00024	44
Average:		**670115.50**	**786544.73**	**15.38%**	**767687.90**	**12.88%**		

Table 5.2.1: EAMA results for instances 0 to 19 - time limit 10 minutes - Iterative LS

The size of initial population was set to 25 and 6 for 50 orders and more respectively. In all instances except 0 and 1 which consist of 50 orders the time limit was hit before the 25th generation was processed.

The results from Table 5.2.1 depict that the algorithm produces solutions that lay in smaller gap from ALNS for instance #3, in comparison to instances #2 and #4 even if quantity of orders remains the same. It may depend on distances between the nodes and how they are spread over the area. In Appendix D it can be clearly seen that the nodes in the instances #2 and #4 are located more dense and closer to each other than in the instance #3 where some customer locations are spread more evenly over the bigger area.

The best solutions generated by the EAMA with the iterative LS strategy are at 12.88% from the ALNS. The average results of the EAMA are at 2.5% worse than the best ones.

Instance	Orders	ALNS	EAMA Average		EAMA Best		a	r^{init}
0	50	198 770	199897.00	0.57%	198940.00	0.09%	0.019	5
1	50	145 840	148371.80	1.74%	147853.00	1.38%	0.019	6
2	100	408 130	460319.60	12.79%	452895.00	10.97%	0.0007	16
3	100	350 090	386102.80	10.29%	379840.00	8.50%	0.0005	14
4	100	408 590	473955.60	16.00%	469293.00	14.86%	0.0009	20
5	150	564020	652040.20	15.61%	614774.00	9.00%	0.00096	30
6	150	570730	648091.20	13.55%	636647.00	11.55%	0.0003	26
7	150	501060	638005.00	27.33%	628550.00	25.44%	0.0005	29
8	200	673430	837635.20	24.38%	806948.00	19.83%	0.0004	34
9	200	698160	902006.40	29.20%	886976.00	27.04%	0.0003	33
10	200	647190	740423.60	14.41%	724309.00	11.92%	0.0004	32
11	200	650640	769611.60	18.29%	742020.00	14.04%	0.0005	39
12	250	767190	962685.00	25.48%	928773.00	21.06%	0.0005	39
13	250	861680	1080086.00	25.35%	1056980.00	22.67%	0.0004	40
14	250	778000	990607.00	27.33%	973816.00	25.17%	0.0003	40
15	250	945940	1170380.00	23.73%	1156150.00	22.22%	0.0004	42
16	300	1103970	1416150.00	28.28%	1405390.00	27.30%	0.0002	45
17	300	1115430	1386426.00	24.30%	1362950.00	22.19%	0.00024	46
18	300	965370	1241098.00	28.56%	1211560.00	25.50%	0.00024	46
19	300	1048080	1232350.00	17.58%	1221650.00	16.56%	0.00024	44
Average:		**670115.50**	**794977.17**	**18.81%**	**779542.20**	**16.62%**		

Table 5.2.2: EAMA results for instances 0 to 19 - time limit 10 minutes - Random LS

The above results show that our EAMA could find results for instances with 50 to 300 orders with around 16.62% gap in comparison to ALNS considering the best solution out of 5 runs per each instance. The random LS used here showed solutions at 3.74% higher costs than the iterative LS scenario. The results show that the instance #3 is easier to solve than the #2 and #4 in both LS scenarios.

The reason to choose two different LS procedures was to try to reach more generations within the same time of 10 minutes. With the random LS scenario one generation is processed faster than with the iterative LS.

The above results showed that even if the iterative scenario lies much lower in speed, approximately 3.5 times slower to go through one generation than the random LS scenario, the exhaustive search techniques almost always deliver better solutions on average over all instances. Speaking about the best solutions reported, the random LS outperformed the iterative procedure only in three positions for instances #0, #3 and #5 containing 50, 100 and 150 orders respectively. It can be explained by the fact that the random LS scenario has processed more generations in 10 minutes and the algorithm has gained its gravity. But for the rest of the test instances the iterative LS scenario generated better solutions than the random LS. Therefore, we observed that even if the iterative LS scenario processes fewer generations during the same amount of time it finds better solutions and improves the quality of solution in each generation better than the random LS. This results in better algorithm performance.

It is also important to mention that most of the computational effort was incorporated in calculating the synchronization. Because of the interdependence of the routes, each transaction of the LS or repair operators triggered longer processing times for recalculation of delivery schedules and self-imposed time windows for the corresponding customer nodes. This has put a challenge in choosing such size of the population and the number of generation that can be processed with this size in 10 minutes. It was a tradeoff decision and it explains the fact that for instances containing 50 orders the initial population size was 25, when for the 100 orders instances and more the size was decreased up to 6 solutions.

Next, we would like to show how the memetic part of the algorithm was incorporated in order to improve the solutions generated by GRASP. The figure below depicts how the MA contributes to improvement of a solution (this graph is built on the part of one of the EAMA solutions for instance #0 with 50 orders).

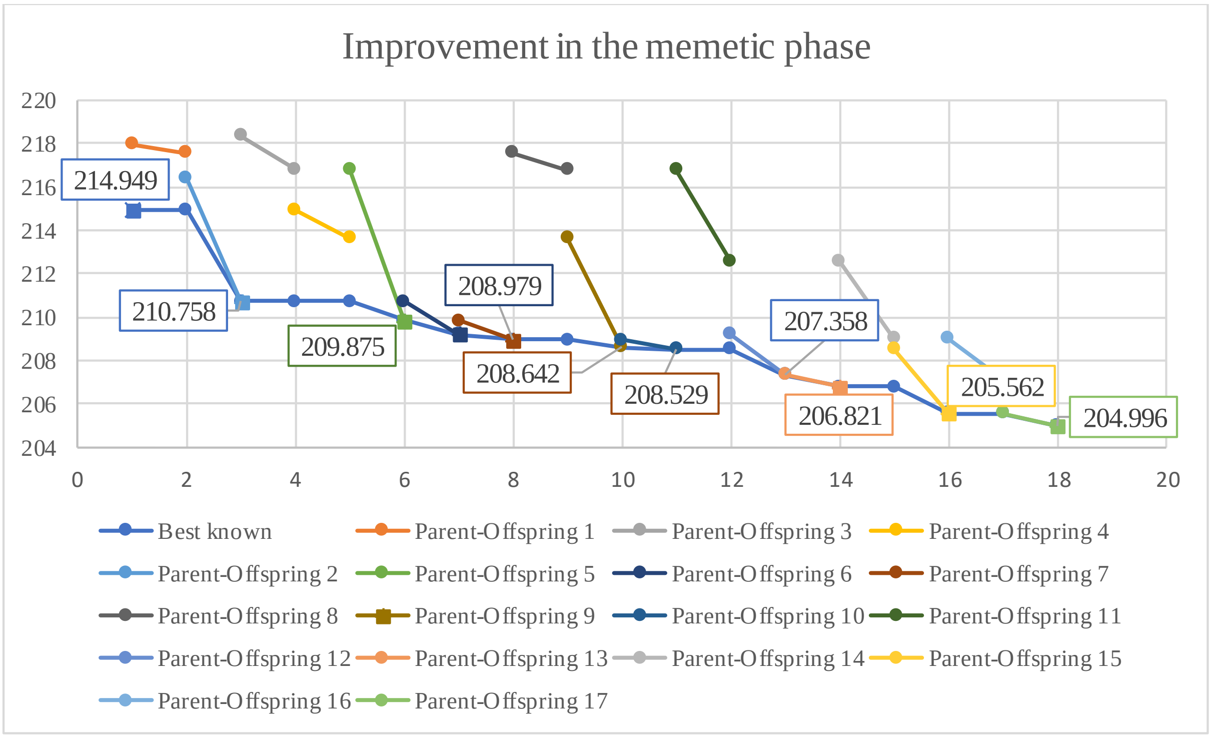


Figure 5.2.1: Improvement in the memetic phase

Figure 5.2.1 contains the records of the best-known solution during one run of the algorithm (the longest sequence starting from 214949 to 204996) and the “parent *A* – offspring” solutions pairs. As we mentioned before, the EAX crossover operator produces offspring solutions that tend to be very similar to parent *A* that is why we illustrate only these pairs here. Interestingly, it may not always be the rule that the parent solutions with lower costs will produce offspring solutions better than those produced by the parent solutions with higher costs. It is demonstrated in Figure 5.2.1. For example, the parent *A* solution, from the pair “Parent-Offspring 4”, with costs 214949 brought an offspring solution which after the repair and LS procedures showed costs at 213638 level. However, the next attempt (“Parent-Offspring 5”) was made and a child solution was built on base of the parent solution with 216836 costs. This parent solution 5 has obviously higher costs than the previous one. But nevertheless, the new offspring solution from parent 5, tuned by LS improvement, could reach the 209875 costs, thus hitting the best-known costs level at that step of the algorithm.

6. Conclusion

Finding smart solutions for city logistics problems contribute to the benefits of all local economic actors. These solutions have also a considerable impact on the quality of life in the city and all city development processes. In this thesis we have provided a literature review on what the city logistics components are, how they interact with each other and which terminology can be used in measuring the effectiveness and efficiency of the city logistics systems.

First, we give a brief description of the VRP problems and their implications in real-life situations in the context of the city logistics. The provided examples stem from the latest research results and show that the routing solutions in urban areas serve for optimization in a broad range of industries, service environments and interactions between all economic, environmental and social units in the city. Neglecting the need for optimization shall drastically harm the development processes and functionality of the city logistics system in general. Several examples of single and multi-objective optimization techniques have been mentioned to strengthen the idea of adapting each problem formulation closest possible to the real-life needs.

Second, the VRP is usually presented as one of the most famous classes of routing problems in the literature and we have presented several examples to support this idea, whereas we included the results from the literature review specifically on city logistics problems. Hence, the VRPSC is here shown to be part of the city development concept. In the problem we address in this thesis the operation and resource synchronization constraints are embodied and several examples have been illustrated to depict the nature of interdependency of the routes in the VRP and how these constraints may increase complexity of the problem. Inclusion of time-dependency in the VRP poses a challenge for optimization techniques, as it was also assumed by (Cattaruzza et al., 2017). It was also shown in this thesis that the interdependence of the routes brings considerable difficulties in the process of optimization in terms of implementation of the metaheuristic part and the computational time to generate good-quality solutions. Starting with general description of synchronization constraints we further provide implementation details and comments on challenging parts we faced in dealing with synchronization constraints in the VRP.

We have implemented the EAMA, proposed by (Nagata et al., 2010) for the VRP with time windows, to solve the VRPSC. The construction heuristics is GRASP based on the savings algorithm. We present decision making aspects of parameters setting for both construction and memetic part of the algorithm. The computational results are shown in comparison to ALNS within 10 minutes processing time of the algorithm. Two different LS scenarios are implemented and compared: the iterative and random LS procedures.

From the perspective of the results of this thesis the follow-up research is needed to develop more straightforward way to calculate the violation of the synchronization constraints under conditions of route interdependence. This shall decrease the computational efforts for processing one generation of the memetic part, giving more possibilities for the algorithm to gain its gravity and to deliver better results. Also, the experiments with the different LS techniques support this position.

A. Appendix

Solution, synchronization of deliveries and self-imposed time windows representation:

Code lines

```
class Solution {
public:
        typedef struct {
                int depot;
                int vehicle_num;
                double duration;
                double
idle_time_veh;
                double
idle_time_cus;
                double load;
                std::vector <int>
customers;
        } Route;

        double total_duration;
        double to-
tal_idle_time_docks;
        double to-
tal_idle_time_veh;
        std::vector<Route>
routes;
};
```

Comments and illustrations

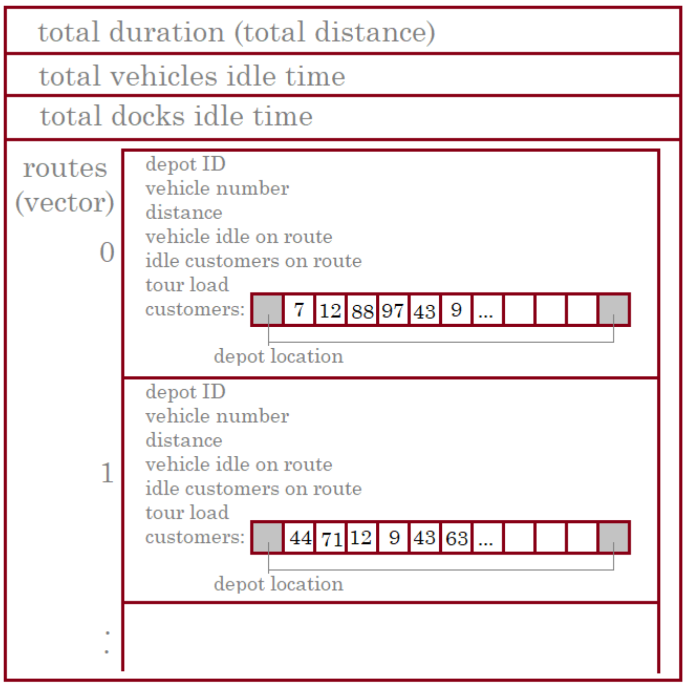


Figure A.1: Solution representation

Vehicle idle time refers to a situation if the vehicle has already arrived at customer point but waits idle and cannot start service because the dock is busy.

Dock idle time occurs each time when service end time of one vehicle is earlier than arrival time of the next vehicle at each customer point.

```
void build_initial_routes(const
vector<vector<int>>
&depots_their_customers, const
vector <depotInfo>
&depotsInfoRead,
       Solution &solution,
       const int
&num_of_orders, const vector
<vector<double>> &distMatrix,
const vector <order>
&customerOrders,
       vector   <vector   <Pair>>
&visitors_schedule,       vector
<vector<double>>  &time_windows,
const   float   &w,   const   int
&work_day_start);
```

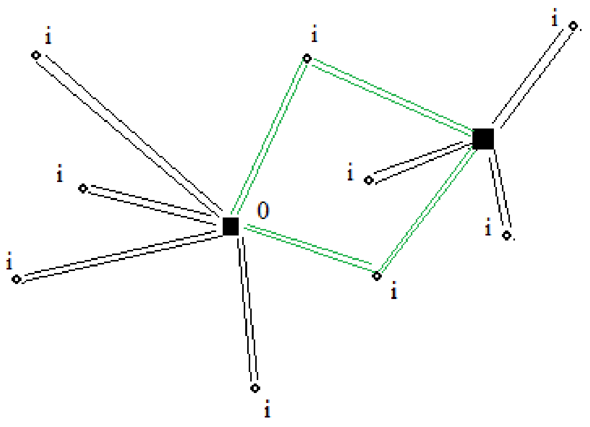


Figure A.2: Initial routes

The green lines visualize routes where vehicles form different depots deliver orders that belong to the same customer locations.

```
struct Pair
{
       int a;
       double b;
};
vector<vector<Pair>> &schedule;
```

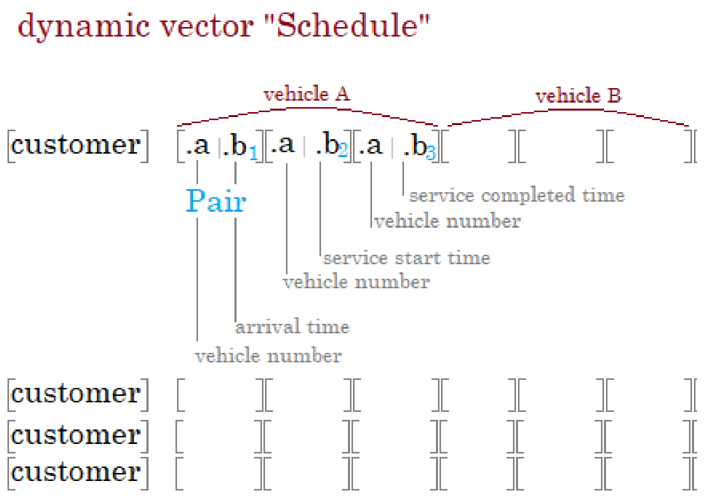


Figure A.3: Deliveries records

This way to keep records gives us possibility to do calculations so that synchronization constraints will not be violated.

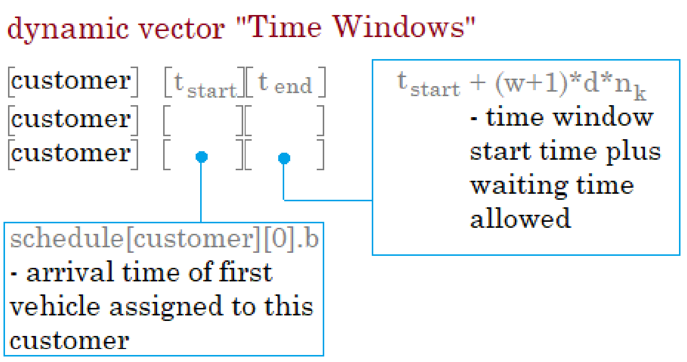
`vector<vector<double>> &tw;`	 dynamic vector "Time Windows" [customer] [t_{start}][t_{end}] [customer] [customer] t_{start} + (w+1)*d*n_k - time window start time plus waiting time allowed schedule[customer][0].b - arrival time of first vehicle assigned to this customer Figure A.4: Self-imposed time windows Recalculation of time windows and schedules take place each time an order is assigned to another vehicle. From code lines on the left we may see how the mechanism of self-imposed time windows work. In different papers self-imposed time windows appear to be designed differently.

Table A.1: Code representation and implementation details

B. Appendix

The records in the table set below are best solutions, each of 100 generated in 1 run of GRASP. Each run was with fixed *a* value indicated in the corresponding column below. The random seed was set to 5.

GRASP_bestof100pop_vrpsc-300-6-19.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
14 880	0.0001	43	1 458 885	0	516 986	1
14 880	0.0002	43	1 458 885	0	516 986	2
14 880	**0.00024**	41	1 415 244	0	436 951	3
14 880	0.0003	43	1 434 589	0	369 100	4
14 880	0.0004	42	1 457 161	17 544	375 650	5
14 880	1	59	2 697 463	6072	485 770	14880

GRASP_bestof100pop_vrpsc-300-6-18.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
14 760	0.0001	47	1 467 110	0	358 986	1
14 760	0.0002	46	1 442 337	0	300 245	2
14 760	**0.00024**	46	1 428 748	0	300 605	3
14 760	0.0003	46	1 429 589	0	355 899	4
14 760	0.0004	46	1 420 588	12 312	377 211	5
14 760	1	56	2 668 543	3 861	465 246	14760

GRASP_bestof100pop_vrpsc-300-6-17.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
14 766	0.0001	50	1 689 635	12 021	504 670	1
14 766	0.0002	46	1 592 224	13 928	395 661	2
14 766	**0.00024**	45	1 585 595	13 928	437 507	3
14 766	0.0003	45	1 589 421	12 021	470 846	4
14 766	0.0004	46	1 597 458	1 907	382 850	5
14 766	1	56	2 836 658	13 563	410 240	14766

GRASP_bestof100pop_vrpsc-300-6-16.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
14 780	0.0001	46	1 643 508	20 932	503 089	1
14 780	**0.0002**	45	1 582 350	0	565 637	2
14 780	0.00024	45	1 599 869	58 570	401 368	3

14 780	0.0003	46	1 598 711	5 932	528 971	4
14 780	0.0004	45	1 589 290	0	314 657	5
14 780	1	57	2 998 321	0	579 480	14780

GRASP_bestof100pop_vrpsc-250-5-15.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
12 304	0.0001	42	1 517 388	26 415	388 358	1
12 304	0.0002	43	1 505 264	26 415	379 217	2
12 304	0.0003	41	1 479 293	0	360 428	3
12 304	**0.0004**	41	1 465 207	29 040	421 124	4
12 304	0.00046	42	1 475 878	3 984	299 498	5
12 304	1	50	2 474 312	0	347 189	12304

GRASP_bestof100pop_vrpsc-250-5-14.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
12 324	0.0001	42	1 10 324	0	485 017	1
12 324	0.0002	40	1 232 661	0	513 554	2
12 324	**0.0003**	38	1 224 367	0	438 197	3
12 324	0.0004	40	1 243 798	0	477 824	4
12 324	1	49	2 272 858	0	430 534	12324

GRASP_bestof100pop_vrpsc-250-5-13.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
12 312	0.0001	43	1 333 302	0	457 398	1
12 312	0.0002	39	1 266 248	0	409 232	2
12 312	0.0003	40	1 272 247	0	320 831	3
12 312	**0.0004**	38	1 258 540	0	374 238	4
12 312	0.00046	40	1 272 169	0	352 094	5
12 312	1	46	2 386 368	0	386 197	12312

GRASP_bestof100pop_vrpsc-250-5-12.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
12 324	0.0001	43	1 218 375	0	464 371	1
12 324	0.0002	38	1 119 017	0	297 369	2
12 324	0.0003	38	1 120 185	0	428 841	3
12 324	0.0004	39	1 122 962	0	311 150	4
12 324	0.00046	38	1 109 437	14 328	284 214	5
12 324	**0.0005**	37	1 094 475	0	352 217	6

12 324	0.0006	38	1 118 804	0	349 280	7
12 324	0.0007	38	1 122 860	0	338 279	8
12 324	1	45	2 162 239	11 285	507 940	12324

GRASP_bestof100pop_vrpsc-200-4-11.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
9 982	0.0002	43	1 208 110	0	527 812	1
9 982	0.0003	37	1 117 511	0	421 833	2
9 982	0.0004	37	1 129 891	0	374 464	3
9 982	**0.0005**	37	1 117 225	0	403 463	4
9 982	0.0006	38	1 123 776	0	333 366	5
9 982	0.0007	38	1 121 782	0	374 341	6
9 982	1	44	1 823 305	0	280 056	9982

GRASP_bestof100pop_vrpsc-200-4-10.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
9 874	0.0002	36	1 009 447	13 799	274 395	1
9 874	0.0003	32	953 240	0	26 115	2
9 874	**0.0004**	31	931 512	0	261 906	3
9 874	0.0005	31	945 566	0	272 508	4
9 874	0.0006	32	948 720	0	243 025	5
9 874	1	44	1 729 349	0	346 373	9874

GRASP_bestof100pop_vrpsc-200-4-9.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
9 808	0.0002	38	1 255 653	0	285 292	1
9 808	**0.0003**	31	1 138 934	0	249 463	2
9 808	0.0004	33	1 157 588	0	264 900	3
9 808	0.0005	32	1 150 880	0	256 426	4
9 808	1	45	2 080 769	0	280701	9808

GRASP_bestof100pop_vrpsc-200-4-8.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
9 918	0.0002	37	1 163 531	0	243 843	1
9 918	0.0003	33	1 113 684	0	259 580	2
9 918	**0.0004**	32	1 085 006	0	265 442	3
9 918	0.0005	34	1 115 340	0	218 356	4

9 918	1	45	1 938 099	0	462 852	9918

GRASP_bestof100pop_vrpsc-150-3-7.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
7 464	0.0002	31	981 172	0	234 619	1
7 464	0.0003	29	952 774	0	205 248	2
7 464	**0.0005**	27	894 503	0	120 094	3
7 464	0.0006	27	895 611	0	1 874 465	4
7 464	1	32	1 428 137	0	157 787	7464

GRASP_bestof100pop_vrpsc-150-3-6.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
7 428	0.0002	28	940 439	0	178 185	1
7 428	**0.0003**	24	877 152	0	150 335	2
7 428	0.0005	25	891 796	0	141603	3
7 428	1	31	1 533 452	0	167 894	7428

GRASP_bestof100pop_vrpsc-150-3-5.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
7406	0.0002	33	1 103 738	0	127 422	1
7406	0.0004	30	1 035 286	0	138 934	2
7406	0.0005	29	1 020 776	0	139 334	3
7406	0.0006	30	1 037 043	0	127 540	4
7406	0.0007	28	1 002 216	0	105 695	5
7406	0.0008	29	1 024 932	0	147 671	5
7406	0.0009	29	1 016 178	0	133651	6
7406	**0.00096**	28	**995 100**	0	129 452	7
7406	0.0011	29	1 018 288	0	122 289	8
7406	0.0012	29	1 006 226	0	139 489	9
7406	1	33	1 520 200	0	218 956	7406

GRASP_bestof100pop_vrpsc-100-2-4.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
4902	0.0003	24	849 214	0	78 095	1
4902	0.0005	19	741 236	0	49 123	2
4902	0.0007	19	737 891	0	37 495	3
4902	**0.0009**	18	731 742	0	27 715	4

4902	0.0012	19	739 677	0	51 931	5
4902	0.004	19	756 848	0	63 437	19
4902	1	21	1 073 765	0	38 021	4902

GRASP_bestof100pop_vrpsc-100-2-3.csv

Savings map size	RCL (a)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
5028	0.0004	15	512 219	0	37 410	1
5028	**0.0005**	13	470 958	0	37 539	2
5028	0.0007	14	479 734	0	51 685	3
5028	0.001	13	463 398	0	37 946	5
5028	1	18	882 907	0	87 689	4950

GRASP_bestof100pop_vrpsc-100-2-2.csv

Savings map size	RCL (a)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
4950	0.0004	17	664 574	0	29 244	1
4950	0.0005	16	640 364	0	25 723	2
4950	0.0006	16	640 205	0	24 217	2
4950	0.00068	15	628 650	0	24 217	3
4950	**0.0007**	15	**627 623**	0	24 716	3
4950	0.00071	15	631 963	0	42 199	3
4950	0.00072	16	640 205	0	24 217	3
4950	0.0008	16	645 244	0	13 350	3
4950	0.001	16	641 424	0	30 750	4
4950	0.0012	16	641 280	0	26 482	5
4950	0.005	16	668 537	0	29 055	24
4950	1	19	1 005 203	0	27 212	4950

GRASP_bestof100pop_vrpsc-50-1-1.csv

Savings map size	RCL (a)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
2450	0.0008	6	161 125	0	0	1
2450	0.0012	6	160 783	0	0	2
2450	0.002	6	160 569	0	0	4
2450	0.004	6	159 981	0	0	9
2450	0.005	5	152 622	0	0	12
2450	0.006	6	158 455	0	0	14
2450	0.007	5	152 106	0	0	17
2450	0.008	5	155 611	0	0	19
2450	0.009	5	154 395	0	0	22
2450	0.01	5	153 416	0	0	24
2450	0.012	5	153 184	0	0	29
2450	0.014	5	151 343	0	0	34

2450	0.016	5	152 756	0	0	39
2450	0.018	5	152 023	0	0	44
2450	**0.019**	5	**151 234**	0	0	46
2450	0.02	5	151 442	0	0	49
2450	0.021	5	152 028	0	0	51
2450	0.022	5	154 268	0	0	53
2450	0.024	5	153 494	0	0	58
2450	0.026	5	153 672	0	0	63
2450	0.028	5	155 683	0	0	68
2450	0.03	5	157 148	0	0	73
2450	0.04	5	159 161	0	0	98
2450	0.05	6	165 930	0	0	122
2450	0.06	5	159 927	0	0	147
2450	0.07	5	163 329	0	0	171
2450	0.08	5	168 780	0	0	196
2450	0.1	5	175 183	0	0	245
2450	0.2	5	190 239	0	0	490
2450	0.4	6	214 271	0	0	980
2450	1	6	303 187	0	0	2450

GRASP_bestof100pop_vrpsc-50-1-0.csv						
Savings map size	RCL (*a*)	Number of routes	Total costs	Total idle dock time	Total idle vehicle time	RCL size
2450	0.0008	5	206 241	0	0	1
2450	0.0012	5	203 974	0	0	2
2450	0.002	5	203 469	0	0	4
2450	0.004	5	205 403	0	0	9
2450	0.005	5	205 305	0	0	12
2450	0.006	5	203 612	0	0	14
2450	0.007	5	203 803	0	0	17
2450	0.008	5	206 160	0	0	19
2450	0.009	5	205 394	0	0	22
2450	0.01	5	205 930	0	0	24
2450	0.012	5	205 293	0	0	29
2450	0.014	5	205 773	0	0	34
2450	0.016	5	207 341	0	0	39
2450	0.018	5	206 676	0	0	44
2450	**0.019**	5	**203 294**	0	0	46
2450	0.02	5	203 855	0	0	49
2450	0.021	5	204 833	0	0	51
2450	1	6	385 507	0	0	2450

C. Appendix

The records in the table set below are the values of 1 algorithm run with random seeds.

GRASP_bestof100pop_vrpsc-300-6-19.csv			
RCL (*a*)	0.00024	Time to build 1 solution	*<0.001s*
Total sols	100		
Min Number of routes	42	Frequent Number of routes	46
Max Number of routes	49	Frequent Number of routes in best 10% solutions	**44**
Number of tours solution 82	42	Duration	1446.27
Number of tours solution 4	44	Duration	1447.75
Number of tours solution 97	43	Duration	1454.11
Number of tours solution 18	43	Duration	1455.24
Number of tours solution 1	43	Duration	1459.35
Number of tours solution 69	42	Duration	1460.67
Number of tours solution 85	44	Duration	1465.99
Number of tours solution 45	44	Duration	1467.02
Number of tours solution 15	44	Duration	1467.77
Number of tours solution 64	44	Duration	1468.36

GRASP_bestof100pop_vrpsc-300-6-18.csv			
RCL (*a*)	0.00024	Time to build 1 solution	*<0.001s*
Total sols	100		
Min Number of routes	46	Frequent Number of routes	49
Max Number of routes	54	Frequent Number of routes in best 10% solutions	**46**
Number of tours solution 55	46	Duration	1428.75
Number of tours solution 77	46	Duration	1430.44
Number of tours solution 64	46	Duration	1432.49
Number of tours solution 52	46	Duration	1442.84
Number of tours solution 90	47	Duration	1442.97
Number of tours solution 20	46	Duration	1443.89
Number of tours solution 51	46	Duration	1444.84
Number of tours solution 84	47	Duration	1446.57
Number of tours solution 0	47	Duration	1450.13
Number of tours solution 36	48	Duration	1457.15

GRASP_bestof100pop_vrpsc-300-6-17.csv			
RCL (*a*)	0.00024	Time to build 1 solution	*<0.001s*
Total sols	100		
Min Number of routes	45	Frequent Number of routes	49
Max Number of routes	53	Frequent Number of routes in best 10% solutions	**46**

Number of tours solution 72	45	Duration	1585.6
Number of tours solution 26	45	Duration	1602.34
Number of tours solution 45	46	Duration	1602.34
Number of tours solution 64	46	Duration	1616.02
Number of tours solution 93	46	Duration	1620.93
Number of tours solution 58	46	Duration	1622.02
Number of tours solution 0	47	Duration	1623.53
Number of tours solution 17	47	Duration	1624.27
Number of tours solution 23	47	Duration	1626.67
Number of tours solution 27	46	Duration	1632.05

GRASP_bestof100pop_vrpsc-300-6-16.csv			
RCL (*a*)	0.0002		
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	45	Frequent Number of routes	48
Max Number of routes	52	Frequent Number of routes in best 10% solutions	**45**
Number of tours solution 66	45	Duration	1582.35
Number of tours solution 28	45	Duration	1584.74
Number of tours solution 54	45	Duration	1608.67
Number of tours solution 33	45	Duration	1613.86
Number of tours solution 93	46	Duration	1618.54
Number of tours solution 59	45	Duration	1619.46
Number of tours solution 84	46	Duration	1622.61
Number of tours solution 31	47	Duration	1624.05
Number of tours solution 45	46	Duration	1628.17
Number of tours solution 58	47	Duration	1628.36

GRASP_bestof100pop_vrpsc-250-5-15.csv			
RCL (*a*)	0.0004		
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	41	Frequent Number of routes	43
Max Number of routes	47	Frequent Number of routes in best 10% solutions	**42**
Number of tours solution 37	41	Duration	1465.21
Number of tours solution 95	41	Duration	1467.51
Number of tours solution 12	41	Duration	1482.69
Number of tours solution 71	42	Duration	1489.69
Number of tours solution 81	42	Duration	1494.73
Number of tours solution 85	42	Duration	1494.91
Number of tours solution 70	42	Duration	1496.42
Number of tours solution 78	42	Duration	1496.48
Number of tours solution 58	42	Duration	1499.83
Number of tours solution 92	42	Duration	1502.54

GRASP_bestof100pop_vrpsc-250-5-14.csv			
RCL (*a*)	0.0003		
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	38	Frequent Number of routes	42
Max Number of routes	46	Frequent Number of routes in best 10% solutions	**40**
Number of tours solution 49	38	Duration	1224.37
Number of tours solution 94	39	Duration	1233.79
Number of tours solution 84	40	Duration	1235.61
Number of tours solution 67	39	Duration	1240.51
Number of tours solution 19	40	Duration	1244.17
Number of tours solution 7	40	Duration	1250.09
Number of tours solution 45	41	Duration	1251.39
Number of tours solution 30	41	Duration	1258.16
Number of tours solution 24	40	Duration	1262.1
Number of tours solution 77	41	Duration	1262.12

GRASP_bestof100pop_vrpsc-250-5-13.csv			
RCL (*a*)	0.0004		
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	38	Frequent Number of routes	43
Max Number of routes	47	Frequent Number of routes in best 10% solutions	**40**
Number of tours solution 77	38	Duration	1258.54
Number of tours solution 22	38	Duration	1272.3
Number of tours solution 16	40	Duration	1293.48
Number of tours solution 91	40	Duration	1295.38
Number of tours solution 47	42	Duration	1302.66
Number of tours solution 29	40	Duration	1305.89
Number of tours solution 19	40	Duration	1306.76
Number of tours solution 51	41	Duration	1306.88
Number of tours solution 1	42	Duration	1308.25
Number of tours solution 31	42	Duration	1308.76

GRASP_bestof100pop_vrpsc-250-5-12.csv			
RCL (*a*)	0.0005		
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	37	Frequent Number of routes	42
Max Number of routes	45	Frequent Number of routes in best 10% solutions	**39**
Number of tours solution 93	37	Duration	1094.47
Number of tours solution 64	38	Duration	1116.56
Number of tours solution 44	39	Duration	1125.47

Number of tours solution 78	39	Duration	1139.53
Number of tours solution 99	39	Duration	1140.41
Number of tours solution 54	40	Duration	1142.42
Number of tours solution 96	39	Duration	1144.46
Number of tours solution 8	40	Duration	1151.35
Number of tours solution 73	40	Duration	1152.84
Number of tours solution 80	41	Duration	1153.08

GRASP_bestof100pop_vrpsc-200-4-11.csv			
RCL (*a*)	0.0005		
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	36	Frequent Number of routes	41
Max Number of routes	44	Frequent Number of routes in best 10% solutions	**39**
Number of tours solution 90	37	Duration	1117.22
Number of tours solution 39	36	Duration	1126.06
Number of tours solution 91	37	Duration	1130.49
Number of tours solution 89	37	Duration	1134.87
Number of tours solution 35	38	Duration	1139.28
Number of tours solution 54	39	Duration	1141.99
Number of tours solution 12	39	Duration	1145.7
Number of tours solution 60	38	Duration	1146.53
Number of tours solution 62	39	Duration	1148.33
Number of tours solution 11	39	Duration	1151.61

GRASP_bestof100pop_vrpsc-200-4-10.csv			
RCL (*a*)			
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	31	Frequent Number of routes	37
Max Number of routes	42	Frequent Number of routes in best 10% solutions	**33**
Number of tours solution 30	31	Duration	931.512
Number of tours solution 4	32	Duration	948.853
Number of tours solution 94	32	Duration	954.959
Number of tours solution 11	33	Duration	960.512
Number of tours solution 27	32	Duration	968.113
Number of tours solution 24	32	Duration	968.772
Number of tours solution 79	32	Duration	971.98
Number of tours solution 72	34	Duration	973.546
Number of tours solution 57	34	Duration	973.934
Number of tours solution 89	33	Duration	977.345

GRASP_bestof100pop_vrpsc-200-4-9.csv			
RCL (*a*)			
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	31	Frequent Number of routes	36
Max Number of routes	38	Frequent Number of routes in best 10% solutions	**33**
Number of tours solution 94	31	Duration	1138.93
Number of tours solution 68	33	Duration	1156.4
Number of tours solution 51	33	Duration	1162.03
Number of tours solution 46	33	Duration	1163.16
Number of tours solution 27	33	Duration	1165.03
Number of tours solution 87	33	Duration	1165.71
Number of tours solution 41	34	Duration	1170.3
Number of tours solution 13	34	Duration	1170.43
Number of tours solution 62	34	Duration	1170.46
Number of tours solution 45	34	Duration	1173.93

GRASP_bestof100pop_vrpsc-200-4-8.csv			
RCL (*a*)	0.0004		
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	32	Frequent Number of routes	36
Max Number of routes	38	Frequent Number of routes in best 10% solutions	**34**
Number of tours solution 39	32	Duration	1085.01
Number of tours solution 79	34	Duration	1102.7
Number of tours solution 9	34	Duration	1111.19
Number of tours solution 19	35	Duration	1120.75
Number of tours solution 15	34	Duration	1124.15
Number of tours solution 22	34	Duration	1126.2
Number of tours solution 49	35	Duration	1127.96
Number of tours solution 69	35	Duration	1135.38
Number of tours solution 26	36	Duration	1137.34
Number of tours solution 12	34	Duration	1137.58

GRASP_bestof100pop_vrpsc-150-3-7.csv			
RCL (*a*)	**0.0005**		
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	27	Frequent Number of routes	32
Max Number of routes	35	Frequent Number of routes in best 10% solutions	**29**
Number of tours solution 44	27	Duration	894.503
Number of tours solution 39	28	Duration	920.833

Number of tours solution 13	29	Duration	939.834
Number of tours solution 63	29	Duration	943.1
Number of tours solution 61	29	Duration	943.849
Number of tours solution 85	29	Duration	947.161
Number of tours solution 60	30	Duration	951.967
Number of tours solution 31	30	Duration	954.035
Number of tours solution 66	30	Duration	955.128
Number of tours solution 45	30	Duration	957.845

GRASP_bestof100pop_vrpsc-150-3-6.csv			
RCL (*a*)	0.0003		
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	30	Frequent Number of routes	32
Max Number of routes	35	Frequent Number of routes in best 10% solutions	**26**
Number of tours solution 60	30	Duration	1025.16
Number of tours solution 80	30	Duration	1029.15
Number of tours solution 61	30	Duration	1037.1
Number of tours solution 86	31	Duration	1040.39
Number of tours solution 90	30	Duration	1041.18
Number of tours solution 43	31	Duration	1042.17
Number of tours solution 85	31	Duration	1042.38
Number of tours solution 9	30	Duration	1044.39
Number of tours solution 72	30	Duration	1044.5
Number of tours solution 38	31	Duration	1046.14

GRASP_bestof100pop_vrpsc-150-3-5.csv			
RCL (*a*)	0.00096		
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	30	Frequent Number of routes	32
Max Number of routes	35	Frequent Number of routes in best 10% solutions	**30**
Number of tours solution 60	30	Duration	1025.16
Number of tours solution 80	30	Duration	1029.15
Number of tours solution 61	30	Duration	1037.1
Number of tours solution 86	31	Duration	1040.39
Number of tours solution 90	30	Duration	1041.18
Number of tours solution 43	31	Duration	1042.17
Number of tours solution 85	31	Duration	1042.38
Number of tours solution 9	30	Duration	1044.39
Number of tours solution 72	30	Duration	1044.5
Number of tours solution 38	31	Duration	1046.14

GRASP_bestof100pop_vrpsc-100-2-4.csv			
RCL (*a*)	0.0009	Time to build 1 solution	*<0.001s*
Total sols	100		
Min Number of routes	18	Frequent Number of routes	22
Max Number of routes	25	Frequent Number of routes in best 10% solutions	**20**
Number of tours solution 5	18	Duration	731.742
Number of tours solution 85	19	Duration	735.162
Number of tours solution 27	19	Duration	762.936
Number of tours solution 26	20	Duration	764.888
Number of tours solution 66	20	Duration	766.691
Number of tours solution 77	20	Duration	770.953
Number of tours solution 9	20	Duration	776.867
Number of tours solution 72	20	Duration	777.213
Number of tours solution 40	20	Duration	778.517
Number of tours solution 29	20	Duration	780.278

GRASP_bestof100pop_vrpsc-100-2-3.csv			
RCL (*a*)	0.0005	Time to build 1 solution	*<0.001s*
Total sols	100		
Min Number of routes	13	Frequent Number of routes	16
Max Number of routes	20	Frequent Number of routes in best 10% solutions	**14**
Number of tours solution 37	13	Duration	470.958
Number of tours solution 24	14	Duration	471.742
Number of tours solution 82	14	Duration	480.474
Number of tours solution 71	14	Duration	486.276
Number of tours solution 16	14	Duration	486.927
Number of tours solution 28	14	Duration	487.158
Number of tours solution 77	14	Duration	487.348
Number of tours solution 41	14	Duration	487.905
Number of tours solution 6	14	Duration	488.008
Number of tours solution 73	14	Duration	488.008

GRASP_bestof100pop_vrpsc-100-2-2.csv			
RCL (*a*)	0.0007	Time to build 1 solution	*<0.001s*
Total sols	100		
Min Number of routes	15	Frequent Number of routes	19
Max Number of routes	21	Frequent Number of routes in best 10% solutions	**16**
Number of tours solution 36	15	Duration	620413
Number of tours solution 75	16	Duration	640422

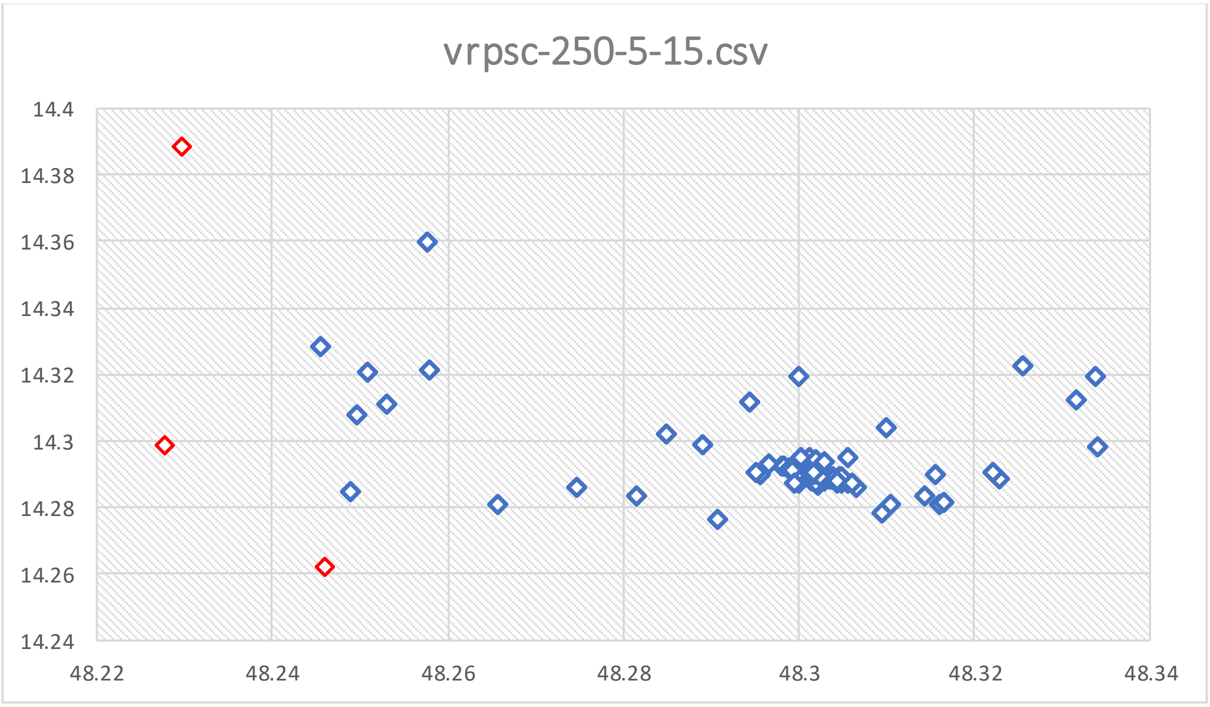
Number of tours solution 98	16	Duration	640422
Number of tours solution 30	16	Duration	644354
Number of tours solution 32	16	Duration	648013
Number of tours solution 97	17	Duration	663618
Number of tours solution 90	16	Duration	663761
Number of tours solution 91	17	Duration	667401
Number of tours solution 37	17	Duration	667687
Number of tours solution 76	17	Duration	667706

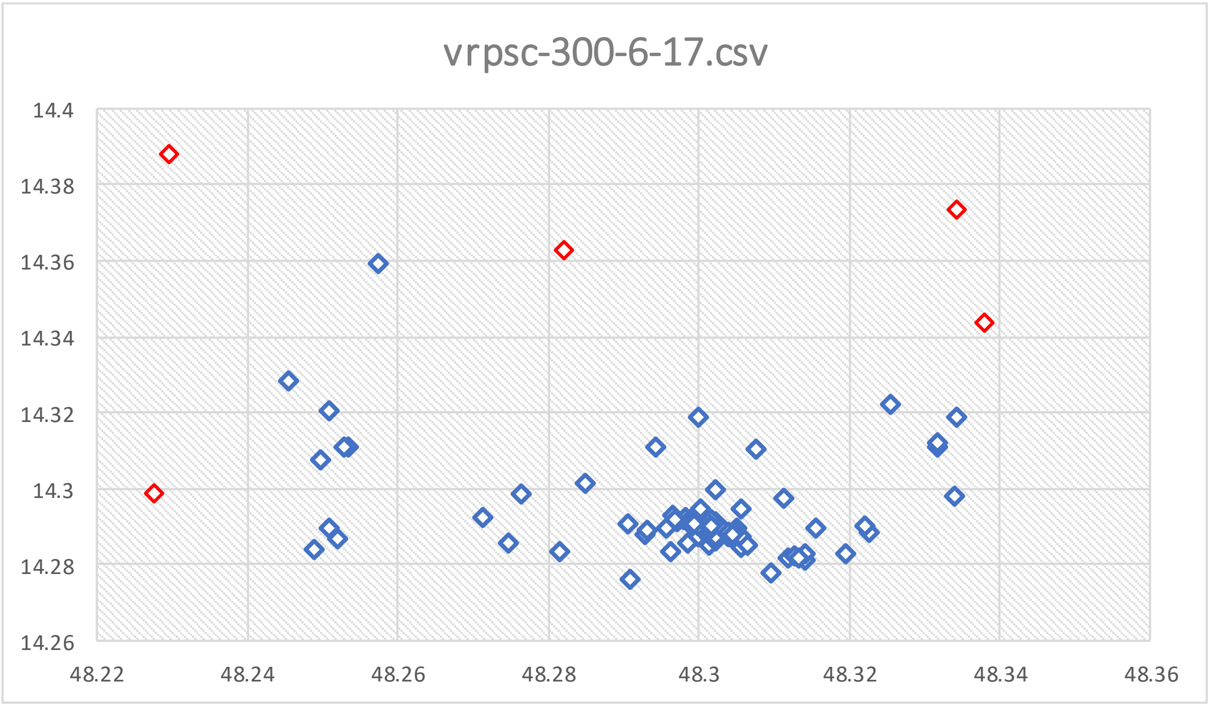
GRASP_bestof100pop_vrpsc-50-1-1.csv			
RCL (*a*)	0.019		
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	6	Frequent Number of routes	6
Max Number of routes	7	Frequent Number of routes in best 10% solutions	**6**
Number of tours solution 72	6	Duration	158.685
Number of tours solution 20	6	Duration	158.751
Number of tours solution 18	6	Duration	159.082
Number of tours solution 86	6	Duration	159.488
Number of tours solution 16	6	Duration	159.521
Number of tours solution 57	6	Duration	159.659
Number of tours solution 7	6	Duration	159.902
Number of tours solution 69	6	Duration	160.172
Number of tours solution 66	6	Duration	160.382
Number of tours solution 89	6	Duration	160.384

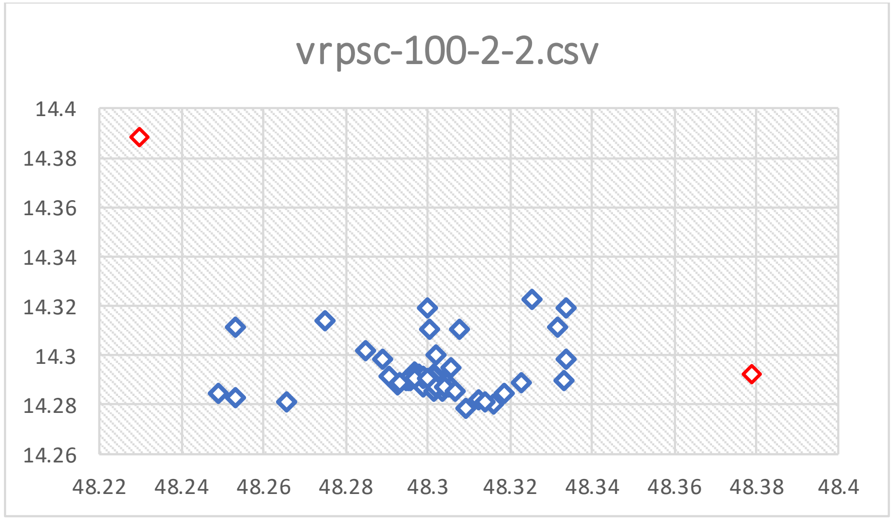
GRASP_bestof100pop_vrpsc-50-1-0.csv			
RCL (*a*)	0.019		
Total sols	100	Time to build 1 solution	*<0.001s*
Min Number of routes	5	Frequent Number of routes	5
Max Number of routes	6	Frequent Number of routes in best 10% solutions	**5**
Number of tours solution 82	5	Duration	206.313
Number of tours solution 16	5	Duration	207.296
Number of tours solution 62	5	Duration	207.608
Number of tours solution 71	5	Duration	207.666
Number of tours solution 17	5	Duration	208.411
Number of tours solution 88	5	Duration	208.841
Number of tours solution 64	5	Duration	208.956
Number of tours solution 99	5	Duration	210.001
Number of tours solution 80	5	Duration	210.147
Number of tours solution 98	5	Duration	210.381

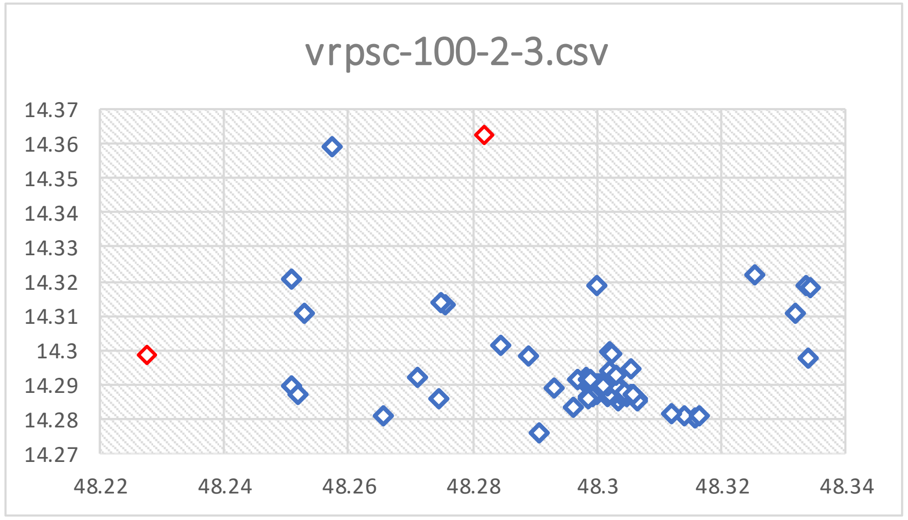
D. Appendix

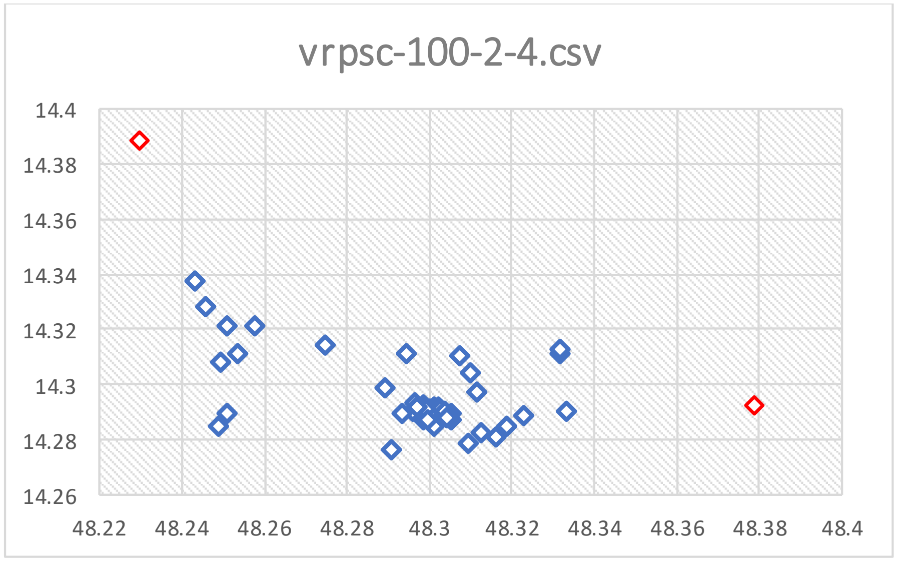
The figures represent geographical locations of customer nodes (blue) and carrier depots (red).

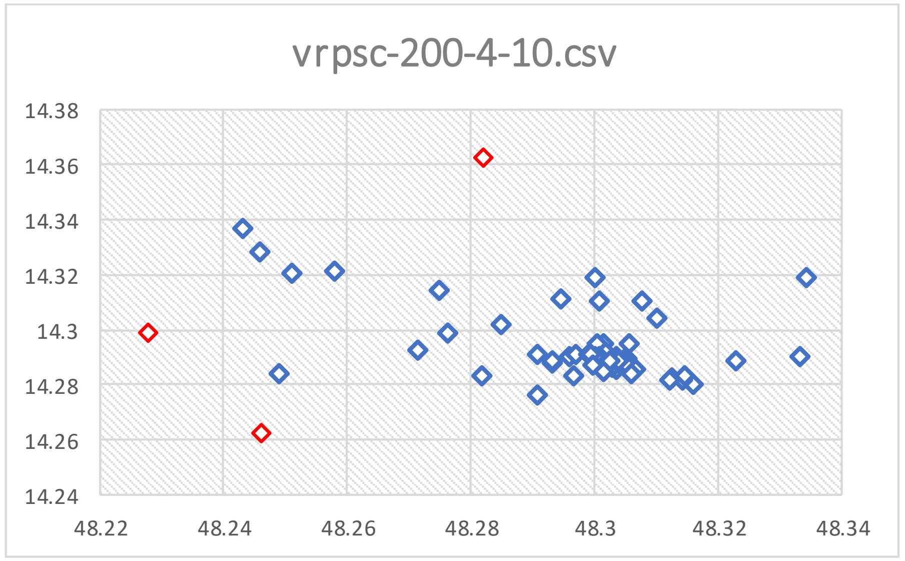





vrpsc-100-2-2.csv
14.4
14.38
14.36
14.34
14.32
14.3
14.28
14.26
48.22
48.24
48.26
48.28
48.3
48.32
48.34
48.36
48.38
48.4


vrpsc-100-2-3.csv
14.37
14.36
14.35
14.34
14.33
14.32
14.31
14.3
14.29
14.28
14.27
48.22
48.24
48.26
48.28
48.3
48.32
48.34


vrpsc-100-2-4.csv
14.4
14.38
14.36
14.34
14.32
14.3
14.28
14.26
48.22
48.24
48.26
48.28
48.3
48.32
48.34
48.36
48.38
48.4


vrpsc-200-4-10.csv
14.38
14.36
14.34
14.32
14.3
14.28
14.26
14.24
48.22
48.24
48.26
48.28
48.3
48.32
48.34

E. Appendix

Instance	**Best results out of 5 runs of the algorithm with diff. *a* values**							
0	32.3740	32.3740	32.3740	32.3740	34.9740	35.4470	36.1240	36.1240
1	42.0920	42.0920	42.0920	42.0920	42.0920	41.7320	44.4510	44.4510
2	52.5180	52.5180	54.3760	55.4150	180.7950	60.9300	82.0520	84.6350
3	45.1350	45.1350	48.1640	48.3210	245.1600	50.4920	53.4060	63.9510
4	140.2740	140.2740	121.3640	107.0440	105.2190	109.0050	120.9920	110.4990
5	159.0810	159.0810	159.0810	160.7100	126.7630	112.9370	110.2320	128.0480
6	164.5930	164.5930	164.5930	133.2090	165.0010	176.4460	149.6010	150.2380
7	102.1640	100.9940	100.9940	102.4130	102.4130	102.3650	107.1550	109.2870
8	188.9780	167.1310	155.0000	167.5800	178.0190	155.2070	155.5780	176.2440
9	146.5100	146.5080	146.5080	84.7400	95.5120	149.6830	156.8920	165.3850
10	167.0350	167.0350	167.0350	152.9020	170.1600	185.3440	173.9250	170.2170
11	103.9020	104.8460	103.9020	104.8460	105.7950	107.3910	103.3070	122.5760
	a =0	*a* =0.04	*a* =0.08	*a* =0.1	*a* =0.2	*a* =0.4	*a* =0.8	*a* =1

Table E.1: GRASP results for small instances 5 to 15 orders

Instance	GRASP (best over 5 iterations)				
	ALNS in hours	GRASP in hours		GRASP in min	RCL *a* values:
0	0.45	0.53956667	0.53956667	32.374	*a* =0 (pure Savings Algorithm) && *a* =0.04 && *a* =0.08 && *a* =0.1
1	0.57	0.69553333	0.69553333	41.732	*a* =0.4
2	0.71	0.8753	0.8753	52.518	*a* =0 (pure Savings Algorithm) && *a* =0.04
3	0.63	0.75225	0.75225	45.135	*a* =0 (pure Savings Algorithm) && *a* =0.05
4	1.17	1.75365	1.75365	105.219	*a* =0.2
5	0.98	1.8372	1.8372	110.232	*a* =0.8
6	1.21	2.22015	2.22015	133.209	*a* =0.1
7	0.99	1.68323333	1.68323333	100.994	*a* =0.04 && *a* =0.08
8	1.41	2.58333333	2.58333333	155	*a* =0.08
9	1.04	1.41233333	1.41233333	84.74	*a* =0.1
10	1.75	2.54836667	2.54836667	152.902	*a* =0.1
11	1.19	1.72178333	1.72178333	103.307	*a* =0.8

Table E.2: Values of *a* in the best GRASP solutions for small instances 5 to 15 orders

	Solomon I1 heuristic		ALNS		CPLEX		GRASP (SA)					
	Average	Best	Average	Best	Cost	time (s)	Average	Best	Gap Avg	Gap Best	Gap w.r.t. Solomon (Avg)	Gap w.r.t. Solomon (Best)
vrpsc-small-5-1-0.csv	32374.00	32374	32374.00	32374.00	32374	20.35	34020.63	32374	5.09%	0.00%	5.09%	0.00%
vrpsc-small-5-1-1.csv	41732.00	41732	41732.00	41732.00	41732	7.67	42636.75	41732	2.17%	0.00%	2.17%	0.00%
vrpsc-small-10-1-2.csv	52518.00	52518	51169.00	51169.00	51169	19.62	77904.88	52518	52.25%	2.64%	48.34%	0.00%
vrpsc-small-10-1-3.csv	45568.00	45568	45008.00	45008.00	45008	242.98	74970.50	45135	66.57%	0.28%	64.52%	-0.95%
vrpsc-small-10-2-4.csv	90505.10	87176	87176.00	87176.00	87176	5.01	119333.88	105219	36.89%	20.70%	31.85%	20.70%
vrpsc-small-10-2-5.csv	78660.20	78621	72408.00	72408.00	72408	21.45	139491.63	110232	92.65%	52.24%	77.33%	40.21%
vrpsc-small-12-2-6.csv	108020.00	103642	87166.00	87166.00	87166	32.08	158534.25	133209	81.88%	52.82%	46.76%	28.53%
vrpsc-small-12-2-7.csv	76252.40	74882	70440.00	70440.00	70440	13.28	103473.13	100994	46.90%	43.38%	35.70%	34.87%
vrpsc-small-14-2-8.csv	106087.00	104853	103720.00	103720.00	103720	445.41	167967.13	155000	61.94%	49.44%	58.33%	47.83%
vrpsc-small-14-2-9.csv	117554.00	117554	78038.30	78038.00	78038	2287.06	136467.25	84740	74.87%	8.59%	16.09%	-27.91%
vrpsc-small-15-3-10.csv	127004.20	124576	123796.00	123796.00	123796	96.95	169206.63	152902	36.68%	23.51%	33.23%	22.74%
vrpsc-small-15-3-11.csv	89406.90	85428	85998.00	85998.00	85998	119.22	107070.63	103307	24.50%	20.13%	19.76%	20.93%
Total	**965681.80**	**948924**	**879025.30**	**879025**	**879025**	**3311.08**	**1331077.25**	**1117362**	**51.43%**	**27.11%**	**37.84%**	**17.75%**
Average									**48.53%**	**22.81%**	**36.60%**	**15.58%**

Table E.3: Comparison of GRASP with Solomon I1, ALNS and CPLEX for small instances 5 to 15 orders

	ALNS			CPLEX		EAMA[4]			
	Average	Best	Runtime (s)	Cost	time (s)	Average	Best	Gap Avg	Gap Best
vrpsc-small-5-1-0.csv	32374	32374	0.01	32374	20.35	32374	32374	0.00%	0.00%
vrpsc-small-5-1-1.csv	41732	41732	0.01	41732	7.67	41732	41732	0.00%	0.00%
vrpsc-small-10-1-2.csv	51169	51169	0.08	51169	19.62	51169	51169	0.00%	0.00%
vrpsc-small-10-1-3.csv	45008	45008	0.02	45008	242.98	45008	45008	0.00%	0.00%
vrpsc-small-10-2-4.csv	87176	87176	0.08	87176	5.01	87176	87176	0.00%	0.00%
vrpsc-small-10-2-5.csv	72408	72408	0.00	72408	21.45	72408	72408	0.00%	0.00%
vrpsc-small-12-2-6.csv	87166	87166	0.03	87166	32.08	87166	87166	0.00%	0.00%
vrpsc-small-12-2-7.csv	70440	70440	1.40	70440	13.28	70440	70440	0.00%	0.00%
vrpsc-small-14-2-8.csv	103720	103720	1.97	103720	445.41	103720	103720	0.00%	0.00%
vrpsc-small-14-2-9.csv	78038	78038	0.04	78038	2287.06	78038	78038	0.00%	0.00%
vrpsc-small-15-3-10.csv	123796	123796	0.06	123796	96.95	124576	124576	0.63%	0.63%
vrpsc-small-15-3-11.csv	85998	85998	0.04	85998	119.22	85042	85042	-1.11%	-1.11%
Total	**879025.30**	**879025**	**3.73**	**879025**	**3311.08**	**878849.00**	**878849**	**-0.02%**	**-0.02%**
Average								**-0.04%**	**-0.04%**

Table E.4: Comparison of the EAMA with ALNS and CPLEX for small instances 5 to 15 orders

[4] The runtime of the EAMA was set as equivalent to ALNS for this comparison.

Bibliography

Bräysy, O., & Gendreau, M. (2005). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation Science, 39*(1), 104-118.

Buhrkal, K., Larsen, A., & Ropke, S. (2012). The waste collection vehicle routing problem with time windows in a city logistics context. *Procedia-Social and Behavioral Sciences, 39*, 241-254.

Byron, J., & Iba, W. (2016). *Population Diversity as a Selection Factor: Improving Fitness by Increasing Diversity*. Paper presented at the Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion.

Cattaruzza, D., Absi, N., Feillet, D., & González-Feliu, J. (2017). Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics, 6*(1), 51-79.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research, 12*(4), 568-581.

Diaz-Gomez, P. A., & Hougen, D. F. (2007). Empirical Study: Initial Population Diversity and Genetic Algorithm Performance. *Artificial Intelligence and Pattern Recognition, 2007*, 334-341.

Doerner, K. F., Huisman, D., & Suhl, L. (2014). Logistics, traffic and transportation. *Flexible Services and Manufacturing Journal, 26*(4), 463-465. doi:10.1007/s10696-014-9196-9

Drexl, M. (2012). Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. *Transportation Science, 46*(3), 297-316.

Gendreau, M., & Potvin, J.-Y. (2010). *Handbook of metaheuristics* (Vol. 2): Springer.

Gendreau, M., & Tarantilis, C. D. (2010). *Solving large-scale vehicle routing problems with time windows: The state-of-the-art*: Cirrelt Montreal.

Glover, F., Gutin, G., Yeo, A., & Zverovich, A. (2001). Construction heuristics for the asymmetric TSP. *European Journal of Operational Research, 129*(3), 555-568.

Grabenschweiger, J., Tricoire, F., & Doerner, K. F. (2018). Finding the trade-off between emissions and disturbance in an urban context. *Flexible Services and Manufacturing Journal, 30*(3), 554-591. doi:10.1007/s10696-017-9297-3

Gupta, D., & Ghafir, S. (2012). An overview of methods maintaining diversity in genetic algorithms. *International journal of emerging technology and advanced engineering, 2*(5), 56-60.

Hemmelmayr, V., Doerner, K. F., Hartl, R. F., & Rath, S. (2013). A heuristic solution method for node routing based solid waste collection problems. *Journal of Heuristics, 19*(2), 129-156. doi:10.1007/s10732-011-9188-9

Hemmelmayr, V. C., Cordeau, J.-F., & Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research, 39*(12), 3215-3228.

Kauf, S. (2016). City logistics–A Strategic Element of Sustainable Urban Development. *Transportation Research Procedia, 16*, 158-164.

Laporte, G., Ropke, S., & Vidal, T. (2014). Chapter 4: Heuristics for the Vehicle Routing Problem. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition* (pp. 87-116): SIAM.

Macharis, C., & Melo, S. (2011). *City distribution and urban freight transport: multiple perspectives*: Edward Elgar Publishing.

Mancini, S. (2013). Multi-echelon distribution systems in city logistics.

Morrison, J., & Oppacher, F. (1998). *Maintaining genetic diversity in genetic algorithms through co-evolution*. Paper presented at the Conference of the Canadian Society for Computational Studies of Intelligence.

Nagata, Y. (2006a). *Fast EAX algorithm considering population diversity for traveling salesman problems*. Paper presented at the European Conference on Evolutionary Computation in Combinatorial Optimization.

Nagata, Y. (2006b). New EAX crossover for large TSP instances. In *Parallel Problem Solving from Nature-PPSN IX* (pp. 372-381): Springer.

Nagata, Y., Bräysy, O., & Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research, 37*(4), 724-737.

Neri, F., Cotta, C., & Moscato, P. (2012). *Handbook of memetic algorithms* (Vol. 379): Springer.

Polacek, M., Hartl, R. F., Doerner, K., & Reimann, M. (2004). A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows. *Journal of Heuristics, 10*(6), 613-627. doi:10.1007/s10732-005-5432-5

Rad, S. T., & Gülmez, Y. S. (2017). GREEN LOGISTICS FOR SUSTAINABILITY. *International Journal of Management Economics & Business, 13*(3), 603-614.

Sarasola, B., & Doerner, K. F. (2018). Adaptive Large Neighborhood Search For a Vehicle Routing Problem with Synchronization Constraints in City Logistics.

Taniguchi, E., & Thompson, R. (2002). Modeling city logistics. *Transportation Research Record: Journal of the Transportation Research Board* (1790), 45-51.

Taniguchi, E., & Van Der Heijden, R. E. (2000). An evaluation methodology for city logistics. *Transport Reviews, 20*(1), 65-90.

Deutsche Zusammenfassung

In dieser Arbeit wird ein metaheuristischer Ansatz zur Lösung eines Vehicle Routing Problems (VRP) mit Synchronisationsbeschränkungen in der Citylogistik vorgestellt. Der Algorithmus generiert Lösungen, die Entscheidungen für Warenlieferungen in Stadtzentren und Einkaufsstraßen darstellen, während Verkehrsbedingungen und -infrastrukturen zeitliche und räumliche Grenzen für Lieferprozesse setzen. Der Datensatz enthält einen Pool von Kunden, die Lieferungen von mehreren Trägerunternehmen erhalten. Der Versuch, diese Lieferungen zu synchronisieren soll die Wartezeiten der Kunden zwischen den Lieferungen verringern und damit ein besseres Timing gewährleisten. Zunächst wird eine Literaturrecherche über das VRP in der Stadtlogistik mit dem Schwerpunkt Synchronisierung vorgestellt. Zweitens beschreiben wir unseren Algorithmus mit Lösungen für das VRP mit Synchronisationsbedingungen, die von (Sarasola & Doerner, 2018) formuliert wurden. Die meisten Komponenten des Algorithmus stammen von einem Penalty-basierten Edge Assembly Memetic-Algorithmus für das VRP mit Zeitfenstern, die von (Nagata et al., 2010) vorgeschlagen wurden. Im letzten Kapitel werden Experimente zur Parametereinstellung und Lösungen für zwei Gruppen von Instanzen vorgestellt, die auf realen Daten in der Stadt Linz, Österreich, erstellt wurden.