universität wien

MASTERARBEIT

Titel der Masterarbeit

GRASP Heuristics for Team Orienteering Problem with Pick-ups and Deliveries

Verfasst von

Éva Valéria Polozun

angestrebter akademischer Grad

Master of Science (MSc)

Wien, 2014

Studienkennzahl lt. Studienblatt:	A 066 914
Studienrichtung lt. Studienblatt:	Masterstudium Internationale Betriebswirtschaft
Betreuer:	o. Univ.-Prof. Dipl.-Ing. Dr. Richard Hartl

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CURRICULUM VITAE

PERSONAL DATA

Name: Valeria Polozun

Date of Birth: 28.09.1987

Place of Birth: Miskolc, Hungary

Nationality: Hungarian

EDUCATION

10/2011 – 02/2014 Master Studies in International Business Administration
Specialization: Supply Chain Management
University of Vienna, Austria

09/2006 – 07/2010 Bachelor Studies in Business Administration
Specialization: Finance and Accounting
Corvinus University of Budapest, Hungary

10/2008 – 02/2009 Exchange Studies- ERASMUS Student Exchange Programme
Eberhard Karls University of Tuebingen, Germany

09/2002 – 06/2006 Fazekas Mihaly Secondary Grammar School, Debrecen, Hungary

ABSTRACT

Der Leser erhält Einblick in eine neue Variante des Vehicle Routing Problems, dem Team Orienteering Problem mit Pick-ups und Deliveries (TOPPD). In diesem Optimierungsproblem müssen ausgewählte Kunden mit einer Flotte von Fahrzeugen mit vorgegebener Transportkapazität bedient werden. Es gibt zwei Arten von Kunden, Kunden mit Überschüssen (Pick-up Kunden) und Knappheit (Delivery Kunden). Beiden sind Gewinnwerte zugeordnet und unser Hauptziel ist es, den Gesamtgewinn durch die Ermittlung von Strecken, die Überschuß und Knappheit an Waren bei den Kunden ausgleichen können, zu maximieren. Wir beschreiben das mathematische Modell und stellen verschiedene Varianten von Suchverfahren (greedy randomised search methods) nach Lösungen für dieses Problem vor. Die Effizienz der vorgeschlagenen Methoden wurden durch computergestützte Tests untersucht. Der Schwerpunkt lag darin, die Methoden zu beurteilen und die Parameter zu identifizieren, die wesentlichen Einfluss auf die Lösungsqualität haben. Der Hauptbeitrag der Arbeit ist die Diskussion der Lösungsmethoden, die darauf abzielen, den Lagerbestand in der Lieferkette zu kontrollieren (z.B. zwischen Einzelhändlern) und die damit verbundenen Kosten zu reduzieren.

ABSTRACT

The reader gets insights into a new class of the Vehicle Routing Problems, the Team Orienteering Problem with Pick-ups and Deliveries (TOPPD). In this routing problem selected customers need to be served by a fleet of capacitated vehicles. There are two types of customers, customers with surpluses (pick-up customers) and shortages (delivery customers). Both are associated with profit values and our main goal is to maximise the total profit by building routes which balance the goods from the pick-up customers to the delivery customers. We describe the mathematical model and introduce different variants of greedy randomised search methods to the problem. The efficiency of the proposed methods has been studied in computational tests. The main focus was to evaluate the methods and identify the parameters which significantly influence the solution quality. The main contribution of the thesis is the discussion of solution methods which are aimed to control the inventory stock within the supply chain (e.g between retailers) and reduce the associated costs.

1. Introduction

Today the Vehicle Routing Problem is seen as one of the most challenging combinatorial optimization problems. In these tasks the aim is to satisfy the customer demand by giving the optimal set of tours, which are then performed by one or more vehicles. Over the years the applications of the Vehicle Routing Problems have been appeared in many fields, including transportation, distribution and logistics.

The current thesis topic is related to the Team Orienteering Problem with Pick-ups and Deliveries (TOPPD), which take a special place in the family of the Vehicle Routing Problems. It belongs to the Pick-up and Delivery problems, in which the customer demands are differentiated according to the flow of goods at the specific stations. If the goods needs to taken away from a specific place, the customer is located at a pick-up station. On the other hand, if goods are desired to be received at a specific place, the customer is a delivery station. In the VRPs, like in the classical Travelling Salesman problem, all customers need to be visited. However, this is not the case in the current thesis topic. There may be customers which are not able to be served within the given limit of tour length. This opens the aspects of an Orienteering Problem. Which customers should be selected for insertion into the route if the tour length is limited? The focus is to identify the most profitable customers in order to maximize the total profit.

I found these two groups of problems highly interesting because the question of the optimal selection and routing- and loading instruction is unified in one single problem. As mentioned before, the selection of the most profitable nodes play crucial role on the quality of the solution. The other focus is how to build the route with the selected nodes within the available limit of tour length. Finally, seeking the optimal loading instruction is inevitable to be able to satisfy the customer demands in an efficient way.

The TOPPD is motivated by its practical relevance. In real-life example, retailers face unbalances of their inventory level. In case of perishable products, like fruits, vegetables or bakery products, the imbalances between retailer partners can cause losses of revenue on a daily basis. In order to balance the surpluses and shortages, the transfer of items from one place to another can help to bring the inventory levels to a balanced level. Another recent application of the problem is the redistribution activity of bicycles

in the public bicycle sharing systems in order to optimize the supply of the rental stations. As the practice shows it is inevitable for the operators of such systems to invest in developing methods to improve the transportation in order to reach an optimal customer service.

The master thesis is structured as follows: the next chapter discusses the theoretical background of the problem. First, we overview the main classes of the vehicle routing problems which are related to the current topic. Subsequently, we discuss the methods applied to similar problems in the literature and provide a detailed description of the greedy adaptive search algorithm. Chapter 3 introduces the mathematical model of the TOPPD problem and presents the algorithms used for solving the problem. In Chapter 4 we discuss the computational experiments, among others we evaluate the solution algorithms and measure the effect of different parameters on the solution quality. Finally, the main findings of the thesis are summarized in the conclusion.

2. Theoretical Backgrounds

2.1 Vehicle Routing Problems

A wide variety of routing problems have been researched in the last decades. The oldest and most intensively studied problem in the operation research and computer science is the Travelling Salesman problem (TSP), in which a salesman starting from a certain city visits a number of other cities and returns to the origin city at the end of his tour. The objective of the problem is to find the path which minimises the total travel distances between the cities. There are two characteristics of the problem: every city must be visited in the tour and there are no values associated to the service, which means the only objective is the minimal distance. Another variant of the TSP is the m-traveling salesman problem (m-TSP), in which more than one salesman travel and visit n cities. Again, each city needs to be visited once and the goal is to minimise the total distance. (Lawler et al., 1985)

Many interesting routing problem have developed from the Traveling Salesman problem over the time. A variant of the TSPs are called TSPs with profits. In these tasks there is a given profit value associated with the cities. This means the selection of a certain city depends not only on the distance but also on the profit value which can be gained by a single visit. There are two opposite objectives in the TSPs with profit: one is to push the salesman to visit as many as cities as possible to collect the maximum total profit, the other one is to minimise the travel distance. Three types of TSPs with profits can be differentiated based on the way how these two opposite objectives are present in the objective functions. In the first category, both objectives are included in the objective function, the aim is find the tour which minimises the travel costs minus collected profit. In the second category, the travel cost is a constraint and the aim is to maximise the profit in such way that the distance does not exceed a certain value. In the third group of such problems, the profit objective is seen as a constraint and the aim is to find the route which minimises the travel distance under the condition that the profit value is not smaller than a preset value. (Feillet et al., 2005)

Table 2.1 Taxonomy of TSPs with profits

Profit	Distance	Problem
Objective	Objective	Profitable Tour Problem (PTP)
Objective	Constraint	Orienteering Problem (OP)
Constraint	Objective	Prize-Collecting TSP (PCTSP)

The second group of the problems are maximization problems which can be seen as a special group of the TSPs, because the distance limit appears to be a constraint and not an objective value. To transform the problem into TSP, the following question can be given: Can a tour be made through all the cities with a maximum length of tour c_{max}? After setting the parameters of c_{max} and the profit values, the OP can be solved. If the solution is a tour including all the cities, the answer is yes. Otherwise, c_{max} needs to be set to a higher value. In this sense, it is to notice that OP is equivalent to the TSP if the distance limit is relaxed to a level when all the nodes can be visited. OP has been also applied in different vehicle routing and inventory scheduling problems as we will see in the next examples. (Feillet et al., 2005)

The Truck Dispatching Problem, studied in 1959 by Dantzig and Ramsey, is known today as one of the most fundamental vehicle problems in the literature. This problem - known also as Capacitated Vehicle Routing problem (CVRP) - can be seen as generalisation of the TSP: seeking the shortest route through given stations under the condition that a certain delivery needs to be made at each visited station. In the classical problem there is a fleet of gasoline delivery trucks which need to supply a number of service stations from the terminal. The aim is minimising the total distances made by the trucks as well as satisfying all of the station demands (Dantzig and Ramser, 1959). After some years the same problem was studied also as an OP, in which the total score is maximized without breaking the distance limit Dmax. (Golden et. al, 1987)

In the classical VRP there is only one kind of demand, pick-up or delivery. A variant of the VRP is the Pick-up and Delivery Problem (PDP), in which the demand of customers may include a pick-up or delivery at the same time. In order to put the PDP into the class of VRPs, the VRPs will be discussed according to different classifications.

First, VRPs can be categorised based on the type of the demand. In the first group there are problems, where the demands require either pick-up or delivery service, but not the both. In the second group, the demand requires pick-up and delivery service, but the services are not associated or are called also as unpaired services. It means that the delivery points can be served from any pick-up point. Finally, problems belonging to the last group are cases where the demands are associated – with other expression they are paired pick-up and delivery services. In this case, the certain delivery point must be served by a given pick-up point. (Parragh et al., 2008, Montané and Galvao, 2002)

According to the flow of the product transfers, the PDPs can be divided into three subgroups. The first group - **many-to-many problems** - consists of the problems where the customers can serve as pickup and delivery points for any commodity. The swapping problem belongs to this group of problems. In this example, each node owns an object of a product type and has a demand for another type of product. A single capacitated vehicle needs to go through the nodes and swap the objects so that the demands of the nodes will be satisfied by the shortest possible path (Anily and Hassin, 1992). Another example is the Pickup and Delivery Traveling Salesman problem, in which one or more vehicle transfer a homogeneous product. Each unit picked up can be used to satisfy any delivery points' demand. (Hernandez-Perez and Salazar-Gonzalez, 2003)

Other class of problems are called **one-to-many-to-one** problems. In these problems, the available products at the depot are transported to the customers and the available products at the customers are transported to the depot. As an example, in the Vehicle Routing Problems with Backhauls (VRPB), which belongs to this group, all the deliveries must be served before the pick-up service can be done. VRBP have been studied as single and multi-vehicle PDP problem. (Montané and Galvao, 2002)

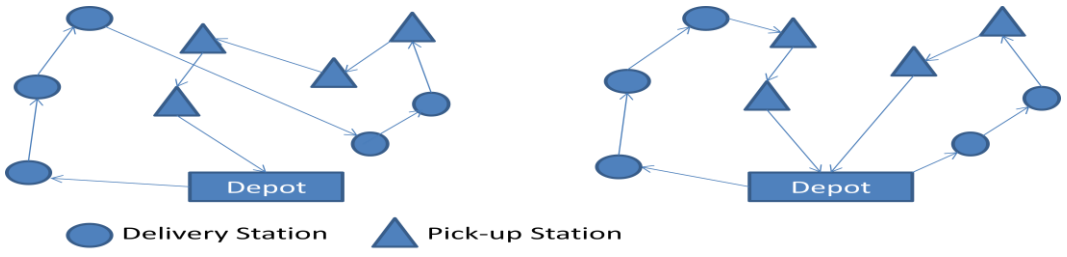


Figure 2.1 Single and Multiple Vehicle Routing Problems with Backhauls

Finally, the third group of problems are the **one-to-one problems**, in which the pick-up and delivery points are strictly connected, each transhipped product has a given origin and destination. (Berbeglia et al., 2010). A well-known example is the Dial-a-Ride problem, where a transportation request is related to one origin and one destination point, resulting paired pick-up and delivery points. The name of the problem refers to the real-life example, when people need to get transported urgently from one certain location to another one. The planning of the transportation of elderly or handicapped people in urban areas is a typical example of this type of problems. A variant of this problem considers the transportation of goods and each request is associated with one pick-up and one delivery station. Since, these service requests are paired, the pick-up station must precede the delivery station. Again, also this VRP has the variants of the problem with single and multiple vehicles.

Table 2.2 Category of pick-up and delivery problems (own representation based on Berbeglia et al, 2010)

	Many-to-many problems	**One-to-many-to-one problems**	**One-to-one problems**
Paired/unpaired	unpaired	unpaired	paired
Problems	Swapping problem Pick-up and delivery TSP TOPPD	Single-Vehicle PDP with single demands and Backhauls Multi-Vehicle PDP with single demands and Backhauls	Dial-A-Ride problem Single Vehicle Routing Problem Multi-Vehicle Routing Problem

The current thesis topic, TOPPD has a position among the many-to-many problems. Similarly to the Pick-up and Delivery TSP, there is one homogeneous product which needs to be transferred. It results that demands are unpaired, so each demand at any delivery station can be satisfied by products from any pick-up stations.

2.2 Taxonomy of VRPs by Bodin and Golden

As broad varieties of VRPs have been developed over the time, the researchers realised the need of a classification system. Applying the taxonomy developed by Bodin and Golden in 1981, the different variants of the VRPs can be categorised and the reader can put the current thesis topic into a clear framework of the VRPs.

In the next section the aspects of the characteristics of the Bodin and Golden taxonomy will be highlighted in details (Bodin and Golden, 1981):

1. time to service a particular node or arc : The question is, whether there is a specific time constraint for the route and/or schedule. There are three possible alternatives: Firstly, the time can be either specified or fixed in advance. In this case problem is purely a scheduling problem. Secondly, there can be time windows, which are certain time intervals, which influence the routing process. The problem is in this case both a scheduling and a routing problem. Thirdly, the time can be unspecified, which means there is no special need to focus on the timing of specific items during the routing process.

2. number of depots : If there is more than one terminal, the choice of the depot has also a significant role in the routing process.

3. *size of vehicle fleet available* : The fact, how many vehicle is available, one or more.

4. *type of fleet available*: The vehicles are either homogeneous - all the vehicles are identical with the same speed and capacity, etc. - or heterogeneous. In this case the vehicles have different features.

5. *nature of demands* : Demands can be deterministic - the exact values of demands are known in advance - , or stochastic - the demands occur with certain probabilities.

6. *location of demands:* The demands can be located on the nodes or on the arcs, or on both.

7. *underlying network :* The nodes can be part of an undirected, directed or mixed network. In case of a directed network the following order of certain nodes are strictly given.

8. *capacity constraints*: The capacity limit is related to the amount of load which one vehicle is able to transport. The limit is either the same for each vehicle or is different. In the third variant of problems, there is no constraint for capacity given.

9. *vehicle route times*: Route-time is to be understood as the maximum duration of time length that one vehicle uses for performing the route. This measure - usually called also as distance limit – is a variable cost factor which is dependent on the length of the tour. Concerning various variants of the problem, for each vehicle there can be the same route-time constraint defined, different route-time constraints defined and there is the case of unconstrained route-time.

10. *costs*: The focus is on the variable routing costs or on the fixed costs – for example, vehicle acquisition costs or other capital costs.

11. *operations*: The type of operations are determined by the characteristics of the nodes which are to be visited – whether these are pickup or delivery stations. In certain problems all the nodes are pick up stations, from where goods need to be delivered to somewhere else. In other problems there are only delivery stations, which are supplied from the depot. The problems, where the route needs to cover both pickup and delivery stations are the so-called pickup and delivery problems.

12. *objective*: The objective of the most vehicle routing problems are the minimization of the costs or size of the fleet. The focus can be put only on the routing costs or in other problems on the routing and fixed costs together. In the third category of the problems only the fixed costs are considered, the aim is e.g. the minimization of the number of vehicles satisfying a certain amount of demand.

Having described the main aspects of the taxonomy, the TOPPD can be defined according to the taxonomy as follows. The time is unspecified, because the goods are transported without any time constraints between the customer locations. In the model there is one depot, from where the trucks start the route and one depot, where the trucks arrive after the route. Since the starting and finishing points are predetermined in advance, there is no strategic decision regarding the optimal starting position. In the TOPPD there is more than one homogeneous vehicle which operates in order to supply the demands. In the current master thesis we go behind the question, what influence the number of vehicles has on the solution quality. The demands are deterministic and are

related to the specific nodes. In terms of costs, in the TOPPD there are two different cost dimensions: on the one hand we consider the variable costs related to the total distance taken in the route. On the other hand, there are costs which are related to the goods transferred. We consider that at the pickup stations the goods can be sold only for a salvage value to the customers. However, if we transfer goods from a location with overages to another station with shortages, the good can be possibly sold at higher prices. The operation of transferring goods from the pickup station to the delivery station means the loss of salvage value at the pickup station, this can be also seen as *cost* - , but at the same there is the realisable *profit* at the delivery station. The difference of these two components – the realised profit at the delivery station minus the lost salvage at the pickup station – provides us the net result of the whole operation. The objective of this specific problem is maximising the total profit by balancing the goods between the pickup and delivery stations.

To summarise, TOPPD put in the framework of Bolden and Golden is overviewed in the figure below:

Table 2.3 TOPPD in the taxonomy of Bodin and Golden (own representation)

	TOPPD
time to service	*unspecified, no time windows*
number of depots	*2 (start and finish)*
size of vehicle fleet	*Multiple*
type of fleet	*Homogeneous*
nature of demands	*Deterministic*
location of demands	*Nodes*
underlying network	*Undirected*
capacity constraint	*the same limit for each vehicle*
vehicle route times	*the same limit for each vehicle*
Costs	*variable and fix*
Operations	*pick-up and deliveries*
objective	*maximisation of profit*

2.3 Repositioning problem in Shared Mobility Systems as related problem to TOPPD

A real-life application of the TOPPD problem is present in the shared mobility systems. Such a system is the bike-sharing system which provides an effective alternative for public transport in the larger cities around the world. The bike-sharing systems including several stations where people can rent and return the bikes are aimed to be a self-operating system. However, the practice shows that it is difficult to meet the customers' demand, because the supply at the stations gets imbalanced due to the different demand of the stations. Thus, the operators need to deliver the bikes from the full stations (pick-up station) to the empty ones (delivery station) by a fleet of vehicles so that there are always a sufficient number of bikes available for the users at the stations. The aim is to find efficient vehicle routes for the transporting cars as well as optimise the load of bikes picked-up and delivered at the visited stations. The bike-sharing system problem is, in contrary to the TOPPD, a minimisation problem, in which the deviation of the targeted number of bicycles, the number of loading activities and time are minimised. According to the recent research high quality solutions have been found to the problem with the application of Greedy Construction Heuristics and Variable Neighborhood Search (Rainer-Harbach et al, 2013).

2.4 Heuristic and Metaheuristic Methods to TOPPD

2.4.1 Greedy Algorithm as Insertion Heuristic Method to TOPPD

Applying insertion heuristics, feasible solutions are constructed in a process, in which unvisited nodes are iteratively inserted into the existing routes. A route is considered as a finished route when there is no more unvisited node existing, which could be inserted into it. In general, there are two main decisions which need to be made at any insertion heuristics: the selection of the next insertion node and the selection of the insertion position.

In order to select the next insertion node to the tour, a criteria function is used. This criteria function can be based usually on the increase of distance, cost or profit. In

general, the different selection rules are usually combined with each other, which can result several variants of insertion heuristics.

In the literature, insertion heuristics are considered as an efficient way for constructing initial solutions for the pick-up and delivery problems. The reason is because they are fast and still can provide relatively good quality solutions. Also, they are commonly used in the commercial routing and scheduling software as a solution approach, which is seen as a good proof of their computational efficiency. (Lu and Dessouky, 2006)

In the classical insertion methods, the next node, which gives the minimal increase of distance or cost, is inserted into the tour, with respect to the capacity and distance limits. However, the degree of the feasibility is usually not considered. For example, how the level of the capacity will be after the insertion or what other nodes will be available to visit are ignored at this point in the insertion process. To reach higher quality solutions, it is recommended to take into account other feasibility constraints. In the greedy insertion heuristic implemented as solution method, the status of the route after the insertion was considered. It means, at the selection process it is not only examined whether the node to be inserted has a high profit value but also whether there are other nodes with high profit values in the surrounding of it which are to be visited after the insertion.

2.4.2 Neighbourhood Search as Improvement Heuristic Method to TOPPD

Improvement heuristics are applied to modify the tours which were previously built by the construction heuristics. These methods are characterised by the application of iterative changes done in the current tour. Specifically, the desired improvements are reached by a search process, in which some components of the solutions, such as nodes, edges, paths or other constructions are manipulated. Since the procedure takes place in the *neighbourhood* of the initial solution, this kind of methods are called neighbourhood searches. Briefly speaking, the procedure consists of two main phases, an initialization process and an iteration process. In the first phase, an initial solution x is generated and the objective value of the current solution is set. In the next phase, the solution is modified and the quality of the new solution will be compared to the original solution. If

there is an improvement, the modified solution will be the new current solution, where the new search can start from.

For understanding the algorithm, it is necessary to clarify what is the meaning of *the neighbourhood of a solution x*. The neighbourhood of the solution $x \in X$ is defined as $N(x) \in X$. N is a function which maps a solution in a set of solutions. There is an initial solution x and during the run of the algorithm the cheapest solution x'' is searched in the surrounding of the solution x. In the next step, the cheapest found solution is compared with the original solution: if c (x') < c (x), the solution x will be updated to x'. The process will be repeated, and new better solution will be searched in the surrounding of the solution x'. The algorithm runs until no new improved solution can be found in the surrounding of solution x, which means *local optimum* is reached. The requirements for a local optimum is reached if $c(x) < c(x') \forall\, x' \in N(x)$. (Pisinger and Ropke, 2010)

Neighbourhood searches are categorised in three main classes in the literature. First, there are the *constructive neighbourhood methods*, in which the initial solution is changed by adding iteratively some new components to it and the other components of the solution remain the same as before. The *transition neighbourhood methods* belong to the second group. Their characteristic is that the solution is iteratively moving from one to another. This is the method, which is usually called local search, because the move only affects the pre-defined neighbourhood of the solution and no new component is added to the tour. The third class of neighbourhood searches are the *population-based neighbourhood methods* which give a generalised form of the previous two methods. The neighbourhoods of more than one solution are considered and one or more new solutions are constructed out of them. (Rego et. Glover, 2002).

Table 2.4 The types of improvement heuristics (based on Prosser et al., 1996)

Within a route	Between Routes
•2-opt •3-opt •k-opt	•Relocate •Exchange •Cross operators

In other system of classes the improvement heuristics are differentiated according to the number of tours which are affected through the improvement process. The procedure

can modify components within one tour or between different tours. An example of an one-tour improvement heuristic is the 2-opt procedure, in which the improvement is reached by reversing parts of the tour. In 2-opt exchanges two edges are eliminated in the tour and the tours are reconnected in a different way so that a new tour will be created. It is to be noted that there is only one way how to reconnect the parts to get a new tour. In the other group of improvement heuristics more than one tour will be modified through the procedure. In the application of a relocation algorithm a customer from a tour will be moved to another tour. The exchange operator swaps customers in different tours with each other. Finally, in the heuristics with cross operators parts of two tours are interchanged. (Prosser et al., 1996)

In the current thesis the proposed algorithms apply the 2-opt local search algorithm as improvement algorithm. The search procedure has been used in an adjusted form to the TOPPD due to the presence of distance and load capacity constraints at the same time. The specific details of the implemented 2-opt local search is presented in details in the section 3.3.

2.5 Greedy Randomized Adaptive Search Procedure (GRASP)

The GRASP metaheuristic was first introduced by Feo and Resende in 1995. This metaheuristic is an iterative procedure, in which the iteration consists of two parts: a constructive heuristic and a local search. In the construction phase a feasible solution is built, whose neighbourhood is investigated until a local optimum is reached. The best found solution is kept as result.

Table 2.5 The GRASP procedure (Feo et. Resende, 1995)

Procedure GRASP (Max_Iterations, Seed)	
1	**For** each seed (k= 1, …, n number of seeds)
2	Build initial solution with greedy randomized construction heuristic
3	Do local search
4	Update best solution (Solution, Best solution)
5	**End For**
6	**Return** Best solution

The pseudo-code in Table 2.5 illustrates the main steps of a general GRASP procedure. The GRASP iterations are performed until the stopping criterion is fulfilled.

Such a criteria can be met if the maximum number of iterations is reached or a desired solution is found. At every iteration – following the solution construction and improvement - it will be checked whether the new solution is better than the best found solution. If a better solution is found, the solution is updated in line 4. At the end of the procedure, the best found solution is returned as result.

In the next part one of the two main procedures of the GRASP will be discussed – the construction phase. A feasible solution is built in an iterative process. Each time the next element to be added needs to be determined. The basis for the selection is given by a candidate list, which contains the available elements to be added with respect to a greedy function. This list is called restricted candidate list. Each element is associated with a benefit value, which gives the measure for the selection. The list changes at every iteration since the benefit value of an element is affected every time a new element is inserted.

Table 2.6 Construction phase in GRASP (Feo et. Resende, 1995)

```
Procedure Construction of Greedy Randomized Solution (Solution)
1   Solution = {}
2   For Solution construction not done →
3                   Make Restricted Candidate List (RCL)
4                   s = Select Element At Random (RCL)
5                   Solution = Solution ∪ { s }
6                   Adapt Greedy Function ( s );
7   End For
```

First, the solution to be constructed is initialized. In the iterative part, four main steps are to be done: making the restricted candidate list of the elements to be added, selecting an element out of the restricted candidate list, adding the selected element to the solution and finally, updating the greedy function. The iterative process stops when the solution construction ends. It is usually the case when there is no more possible candidate which can be added to the solution.

After the GRASP solution is constructed, the improvement procedure takes place as next. Since the constructed solution is not locally optimised, the local search is a good attempt to improve the initial solution. We can see in Table 2.7 how the local search process operates as an iterative process. At the start the solution *s* is determined to the

optimisation problem P. The initial solution *s* will be compared to other solutions in the neighbourhood *N(s)*. If an improved solution *s'* is found, the solution *s* is updated to *s'* and the search continues in the neighbourhood of the updated solution *s*. The search for better solutions terminates when no better solution is found in the neighbourhood. The solution *s* has become locally optimal.

Table 2.7 GRASP local search phase (Feo et. Resende, 1995)

Procedure Local search (P, N(P),*s*)	
1	**For** *s* not locally optimal →
2	Find a better solution s' ϵ *N (s);*
3	Update *s* = *s'*;
4	**End For**
5	**Return** *s* as local optimal for P

Due to the exponential time duration of the local optimization processes, the quality of the initial solution can play a significant role in the efficiency of these procedures. The better the starting solution is, the more efficient the local search process will be. Thus, it can be assumed that the solutions constructed by the GRASP would require less time for the local search than any random starting solution.

GRASP can be seen as a repetitive sampling technique, which produces a sample solution from an unknown distribution of all obtainable results (Feo et. Resende, 1995, p.12-13). The mean and the variance of the solutions depend on the size of the restricted candidate list. If the restricted candidate list contains only one customer, then the GRASP is deterministic and returns the same solution in each iteration. This implies that the mean equals to the solution itself and the variance of solutions is always zero. On the other hand, if there is more than one customer in the candidate list, different solutions will be produced. The variance of the solutions is determined by the restriction level of the greedy function. If the greedy function becomes less restricted, weaker candidates get the possibility to be inserted in the solution, which results that the mean of the solutions decreases. However, the quality of best-found solution and the robustness are the indicators for the quality of the procedure. (Feo et. Resende, 1995).

2.6 Solution construction

In case of building more than one route, tours can be constructed in two different ways. One method is the *sequential* construction building method, which means that the routes are built after each other: when one route is built, the other route can be started to be constructed. The other option is the *parallel* construction method, in which the routes are built simultaneously. The implementation of these two methods will be described based on latest researches of the problem field.

2.6.1 The Sequential Construction Algorithm

Hosny and Mumford (2012) have applied a sequential construction heuristic in their research of the multiple vehicle problem pick-ups and deliveries with time windows. In their method pick-up and delivery pairs are inserted together in the route. Two phases of the heuristic can be differentiated. In the first phase, the pick-up and delivery pairs are initially inserted to the end of the tour. The process will be then followed in the second phase by an algorithm, which is to improve the given tour by swapping the locations and keeping the tour feasible. In case, the pair remains infeasible even after the improvement process, the pair will be removed from the tour. The advantage of this method is the fast speed of the insertion process: for choosing the insertion place there is no need to calculate the costs for all the insertion points in advance and choose the best one, because the pick-up and delivery pair is simultaneously inserted at the end of the route. After this step, the improvement heuristic determines whether a swap can be done in order to improve the quality of the solution. A swap is considered only in cases where the latter location has a deadline which precedes the earlier location. (Hosny and Mumford, 2012). Firstly, the position of the pick-up is determined and then the delivery customer is positioned (if needed). At this stage there is no need to screen all the insertion positions in the route, only the positions after the pick-up customer are considered. Similarly, in our model we apply this approach for the insertion of pair of customers, which contributes not only to the efficient usage of load but also to a faster routing process.

2.6.2 The Parallel Construction Algorithm

In parallel construction algorithms the routes are constructed simultaneously and it is assumed that more than one route will be built at the end of the process. In general, the number of vehicles determines how many routes are built in total.

In our example of TOPPD the number of vehicles has been set as a parameter. However, in the practice of the parallel construction algorithm the number of vehicles is not always known in advance and in such cases the number of vehicles needs to be estimated. In order to get an appropriate estimation, the parallel construction algorithm can be preceded by a sequential construction algorithm. In the practice, the number of routes constructed by the sequential algorithm is commonly used as an estimation for the number of routes for the parallel construction algorithm (Potvin and Rousseu, 1993).In another approach of Hosny and Mumford (2012) the number of vehicles is estimated based on the capacity of the vehicles and the total demand of the pick-up stations. According to their formula shown in Formula (1) the estimated number of vehicles M equals to the total demand of pick-up requests divided by the capacity of one vehicle. N+ represents the set of all the pick-up customers.

$$M = \left[\frac{\sum_{i \in N^+} q_i}{C}\right] \quad (1)$$

Given the estimated number of the vehicles, the next question is how to insert the selected node or nodes into the route. Hosny and Mumford (2012) proposed three different parallel construction algorithms. In the first one, the selected pickup and delivery pair is inserted into the first route in which the feasible insertion is possible. The advantage of this approach is the small run time of the heuristic, because there is no need to make comparisons to decide where the insertion would have the highest benefit. On the other hand, in case the constraints are not very strict, it can lead to similarities to the sequential algorithms. We can note that the nodes will be tried to be inserted into the first route as long as the feasibility rule does not break. In the second proposed algorithm, the selected pair is inserted into that tour which causes the least increase in the overall cost. In their third parallel heuristic algorithm, not only the best tour for the

insertion is selected but also the best unvisited pickup and delivery point pair – the pair which causes the least overall cost increase - will be chosen.

3. Team Orienteering Problem with Pick-up and Deliveries (TOPDP)

The Team Orienteering Problem with Pick-ups and deliveries can be seen as the combination of TO and the PDP. There are two set of points, the pick-up and delivery stations. The objective of the problem is to select the stations which will be visited and build tours to maximise the total profit collected. There are limited resources which has the consequence that there may be stations which may not be served and even if the visit occurs, the amount of goods transferred from or to a specific station covers only a part of the demand of that station.

In the next session two mathematical models will be discussed. First, considering the one-vehicle model as a basis for the multiple-vehicle case of TOPDP, we introduce the mathematical model of the one-vehicle case. Secondly, taking into account the number of vehicles, we overview the multiple-vehicle TOPPD model. This will be then followed by the explanation of the proposed methods implemented to the problem.

3.1 Mathematical Model

3.1.1 Description of the One-vehicle Model

The problem can be considered as follows. We have n retailers, each with a certain amount of demand balance. The demand balance of the retailer *i* is positive, when the retailer has a surplus of goods. We consider this point as a pick-up point with the surplus of *qi*. Without any interaction, the retailer *i* could sell its surplus of goods for a salvage value of p_i. The demand balance of another retailer j is negative, in case the retailer has a shortage of goods. These stations are seen as delivery points where the goods can be transported to. By selling each good that is transhipped to the delivery station, the revenue of p_j can be realised.

Assuming that q_j – the shortage of retailer j - is available at the retailer i. Transhipping q_j amount of goods from the retailer i to the retailer j may bring a benefit. The loss of the salvage value at the pick-up station will be compensated by the higher gain realised at the delivery station. The total gain of this transhipment equals to $p_i q_i$ -

$p_j q_i$, which equals to the gain at the delivery station minus the loss of salvage value at the pick-up station.

The unbalances of the retailers are to be reduced and managed by the operations of a number of homogeneous vehicles that serve the stations with the desired amount of goods. It is to notice that the operation can occur only within the available resources, namely within the maximum number of vehicles and capacity and distance limit of each vehicle.

Using one vehicle for the transhipments, the fix costs F are considered to make one tour. In these fix costs all those expenses are included which are independent from the length of the tour and the customers which are to be visited. As example, such costs can be the purchasing costs of the vehicle, the cost of the usage of motorways, etc.

The transportation costs are dependent on the tour length. All the costs are included in the cost matrix, in which c_{ij} stands for the cost which occurs when the vehicle makes a transhipment from the node *i* to the node *j*.

Concerning the decision variables, z_i is a binary variable and is used for the selection of the nodes which are to be visited. x_{ij} is the binary variable which is used to determine the tour connecting the selected nodes with each other. The variable x_{ij} equals to 1, when the tour goes from the node *i* to node *j*.

Taking into consideration all the costs and profit factors above, the objective function of the model with the constraints can be given as follows:

$$TP = max \ \sum_{i \in V} p_i y_i - \sum_{i,j \in V \cup \{0\}} c_{ij} x_{ij} - F_z \qquad (2)$$

Subject to

$$\sum_{j \in V \cup \{0\} \setminus \{i\}} x_{ij} = \sum_{j \in V \cup \{0\} \setminus \{i\}} x_{ji} = z_i \qquad \forall i \in V \cup \{0\} \ (3)$$

$$\sum_{i \in V} z_i \leq Mz \qquad (4)$$

$$\sum_{j \in V} x_{0j} = \sum_{j \in V} x_{j0} = z \qquad (5)$$

$$y_i \leq z_i \qquad \forall i \in V \ (6)$$

$$x_{ij} = 1 \Rightarrow s_i + d_{ij} \leq s_j \qquad \forall i, j \in V \cup \{0\} \ (7)$$

$$x_{ij} = 1 \Rightarrow Q_j = Q_i + q_i y_j \qquad \forall i, j \in V \cup \{0\} \ (8)$$

$$Q_0 = 0 \quad (9)$$

$$0 \leq Q_i \leq Q_{max} \quad \forall i \in V \cup \{0\} \quad (10)$$

$$\sum_{i,j \in V \cup \{0\}} d_{ij} x_{ij} \leq D_{max} \quad (11)$$

$$x_{ij}, z_i, z \in \{0, 1\} \quad \forall i, j \in V \cup \{0\} \quad (12)$$

$$s_i, Q_i \geq 0 \quad \forall i \in V \cup \{0\} \quad (13)$$

$$0 \leq y_i \leq 1 \quad \forall i \in V \quad (14)$$

In the constraint (3) the flow balance of each node can be seen. z_i is a binary variable that equals to 1, if the node i is included in the tour and equals to 0, if not included. If there is a connection between the node i and node j, both x_{ij} and x_{ji} must equal to 1. Otherwise, if they are not connected with each other, $x_{ij} = x_{ji} = z_i = 0$. In the constraint (4) and (5) the variable z is modelled. Constraint (4) assures that if the node i is included in the tour – in this case $z_i = 1$, then z must equal to 1, which means the tour is built. Constraint (5) says, if there are any nodes connected to the central depot ($x_{0} = 1$), the variable z must also equal to 1. Constraint (6) is related to the loaded quantity which is transhipped. The variable y_i stands for the portion of the quantity q_i transhipped and has a value between 0 and 1. It reassures that in case node i is not included in the tour – resulting that the value $z_i = 0$, y_i must necessarily equal to 0 as well. Constraint (7) guarantees that subtours are eliminated. If the route goes from the node i to the node j, then the total distance s_j of the route from the start to the node j must not be less than the total tour distance s_i calculated from the start depot to the node i plus the distance between the nodes i and j. Constraint (8), (9) and (10) are related to the loaded quantities. Constraint (8) defines that if x_{ij} has the value of 1, then the quantity on the load at the node j must equals to the sum of the quantity on the load at the node i plus the quantity loaded at the node j. The loaded quantity at the start is 0 and has to be always between 0 and the maximum capacity during the whole tour. The constraint (11) concerns the tour distance and defines the limit of the total tour length as D_{max}.(R. Hartl et. al, 2013)

3.1.2 Description of the TOPPD Model as Many-vehicle Model

In TOPPD there is more than one vehicle routed across all the pick-up and delivery customers. Each customer can be visited only once, which means that different vehicles cannot be routed to the same request even if the customer request is only partially satisfied (a request is partially satisfied in case $y_i < 1$).

Adjusting the model to the many-vehicle case, we introduce the set *K*, which includes number of available vehicles and z_k, which shows us whether the tour *k* is built (=1) or not (=0). We can consider this model as the extended version of the one-vehicle model. In this sense, the objective function and the constraints are modified into the following form:

$$TP = max \sum_{i \in V} p_i y_i - \sum_{i,j \in V \cup \{0\}, k \in K} c_{ij} x_{kij} - F \sum_{k \in K} z_k \quad (15)$$

Subject to

$$\sum_{j \in V \cup \{0\} \setminus \{i\}} x_{kij} = \sum_{j \in V \cup \{0\} \setminus \{i\}} x_{kji} = z_{ki} \qquad \forall k \in K,\ \forall i \in V \cup \{0\} \quad (16)$$

$$\sum_{i \in V\ k \in K} z_{ki} \leq M z_k \qquad \forall k \in K \quad (17)$$

$$\sum_{j \in V} x_{k0j} = \sum_{j \in V} x_{kj0} = z_k \qquad \forall k \in K \quad (18)$$

$$y_i \leq \sum_{k \in K} z_{ki} \leq 1 \qquad \forall i \in V \quad (19)$$

$$x_{kij} = 1 \Rightarrow s_{ki} + d_{ij} \leq s_{kj} \qquad \forall k \in K\ \ \forall i, j \in V \cup \{0\} \quad (20)$$

$$x_{kij} = 1 \Rightarrow Q_{kj} = Q_{ki} + q_{kj} y_j \qquad \forall k \in K\ \ \forall i, j \in V \cup \{0\} \quad (21)$$

$$Q_{k0} = 0 \qquad \forall k \in K \quad (22)$$

$$0 \leq Q_{ki} \leq Q_{max} \qquad \forall i \in V \cup \{0\} \quad (23)$$

$$\sum_{i,j \in V \cup \{0\}} d_{kij} x_{kij} \leq D_{max} \qquad \forall k \in K \quad (24)$$

$$x_{kij}, z_{ki}, z_k \in \{0, 1\} \qquad \forall k \in K\ \forall i, j \in V \cup \{0\} \quad (25)$$

$$s_{ki}, Q_{ki} \geq 0 \qquad \forall k \in K\ \forall i \in V \cup \{0\} \quad (26)$$

$$0 \leq y_i \leq 1 \qquad \forall i \in V \quad (27)$$

In comparison with the one-vehicle model, we note that the number of variables is exactly multiplied by $|K|$ except the decision variable y_i. The reason for that is the fact that the decision variable y_i is not related to any of the routes, but it represents the level of satisfaction of certain customer requests. All the other variables remain the same in content but are related to the route z_k .

There are significant changes to highlight in terms of the constraints: Constraint (19) reassures that in case the node i not included in any of the tours, the decision variable y_i is 0. Furthermore, the constraint guarantees that the maximum sum of the variables z_{ki} equals to one at any location i, which means that each request is visited maximum once during the routing procedure and cannot be included in more than one route.

3.1.3 Description of the TOPPD Model in the Experiments

In our experiments we implement a special version of the general TOPPD model described above. We set the limit on the number of tours built: we examine only the cases, in which the number of vehicles are $|K|$= 1, 2, 3. Considering that the set of K is determined in advance, the objective function and the constraints are modified as follows:

$$\boldsymbol{TP = max} \ \sum_{i \in V} \boldsymbol{p_i y_i} - \sum_{i,j \in V \cup \{0\}, k \in K} \boldsymbol{c_{ij} x_{kij}} - \ \boldsymbol{F|K|} \qquad \textbf{(28)}$$

Subject to

$$\sum_{j \in V \cup \{0\} \setminus \{i\}} \boldsymbol{x_{kij}} = \sum_{j \in V \cup \{0\} \setminus \{i\}} \boldsymbol{x_{kji}} = \boldsymbol{z_{ki}} \qquad \forall \boldsymbol{k} \in \boldsymbol{K}, \ \forall \boldsymbol{i} \in \boldsymbol{V} \cup \{\boldsymbol{0}\} \ \textbf{(29)}$$

$$\boldsymbol{y_i} \leq \sum_{k \in K} \boldsymbol{z_{ki}} \leq \boldsymbol{1} \qquad \forall \boldsymbol{i} \in \boldsymbol{V} \ \textbf{(30)}$$

$$\boldsymbol{x_{kij} = 1 \Rightarrow s_{ki} + d_{ij} \leq s_{kj}} \qquad \forall \boldsymbol{k} \in \boldsymbol{K} \ \forall \boldsymbol{i, j} \in \boldsymbol{V} \cup \{\boldsymbol{0}\} \ \textbf{(31)}$$

$$\boldsymbol{x_{kij} = 1 \Rightarrow Q_{kj} = Q_{ki} + q_{kj} y_j} \qquad \forall \boldsymbol{k} \in \boldsymbol{K} \ \forall \boldsymbol{i, j} \in \boldsymbol{V} \cup \{\boldsymbol{0}\} \ \textbf{(32)}$$

$$\boldsymbol{Q_{k0} = 0} \qquad \forall \boldsymbol{k} \in \boldsymbol{K} \ \textbf{(33)}$$

$$\boldsymbol{0 \leq Q_{ki} \leq Q_{max}} \qquad \forall \boldsymbol{i} \in \boldsymbol{V} \cup \{\boldsymbol{0}\} \ \textbf{(34)}$$

$$\sum_{i,j \in V \cup \{0\}} \boldsymbol{d_{kij} x_{kij} \leq D_{max}} \qquad \forall \boldsymbol{k} \in \boldsymbol{K} \ \textbf{(35)}$$

$$x_{kij}, z_{ki} \in \{0,1\} \qquad \forall k \in K \; \forall i,j \in V \cup \{0\} \; (36)$$

$$s_{ki}, Q_{ki} \geq 0 \qquad \forall k \in K \; \forall i \in V \cup \{0\} \; (37)$$

$$0 \leq y_i \leq 1 \qquad \forall i \in V \; (38)$$

We can see that certain constraints relating to the variable *z* do not appear in the same form as in the one-vehicle-model previously. The question whether the tour is started or not (the constraints (17) and (18) in the general many-vehicle model) is not relevant anymore, because the total number of tours is predetermined.

3.1.4 Special cases

The aim is to maximise the profit by balancing the surpluses and shortages between the nodes. However, there is the possibility that the tour does not contain all the unbalanced nodes, because of the following reasons:

- Small amount of deficit at node *j* : if the transportation cost c_{ij} is higher than the expected profit of $p_j y_j$ at the node *j*, the net gain of the insertion is negative:

$$c_{ij} + p_i y_i < 0 \qquad (39)$$

-Node *j* is located far from other surplus nodes: although the deficit is sufficiently high, the transportation cost c_{ij} exceeds the expected profit because of the large distance from node *i* to node *j*:

$$c_{ij} + p_i y_i + p_j y_j < 0 \qquad (40)$$

- Capacity cannot be higher than *Qmax*: certain nodes are not inserted, otherwise the capacity constraints is violated.

- Tour length has the maximum limit of *Dmax*: certain nodes are not inserted otherwise the distance constraint is violated.

- Case of an imbalanced problem: This is the case if all the nodes have deficits and any tour has no benefit.

3.2 Implementation of Construction Heuristics

Three different construction heuristics have been applied for building the initial tours:

1. Method S - Serial Insertion Heuristic with separate selection of pick-up and delivery stations
2. Method SS - Serial Insertion Heuristic with non-separate selection of pick-up and delivery stations
3. Method PP - Parallel Insertion Heuristic with non-separate selection of pick-up and delivery stations

The following four processes can be considered as crucial parts in the structure of all the three heuristics: the evaluation of the unvisited customers (scoring) (1), the selection procedure (2), the insertion procedure of the selected customer (3) and finally the determination of the quantity to be transhipped (4). These processes are described in the next part of the thesis in details – the first process shows unique cases for each heuristic, the other parts are common procedures that run in the same way in all the three heuristics.

3.2.1 Scoring the Unvisited Customers

3.2.1.1 General Evaluation of Pick-up and Delivery Customer Pairs

All the three proposed algorithms are greedy algorithms, in which always the best fitting customer or pair of customers are selected from the restricted candidate list and added to the tour. In order to avoid having an unbalanced number of pick-up and delivery customers in the tour, at every iteration - dependent on the type of algorithm - either the two type of request are added separately, the selection and insertion of a pick-up request is followed by the selection and insertion of a delivery request; or the pick-up-delivery customers are chosen and added to the tour in pair.

In order to perform the selection among the customers and every time choose the best fitting one to insert in the tour as next customer, the expected profits of the customers are calculated first. As it was previously discussed in the description of the mathematical model, each customer is associated with a profit value p_i which is positive or negative according to the type of request. However, it is not always possible to reach the maximal profit of the delivery customers. There are various cases when the delivery request is only partially satisfied, e.g there are no sufficient amount of goods on the load

to be delivered to that certain customer or there is another more beneficial delivery customer at a further location in the tour. Thus, it seems to be reasonable to consider another approach, in which pairs of pick-up and delivery customers are built into the tour together. The pairs are associated with a profit value which is realised if both customers are included in the tour. The values of all possible pick-up and delivery customer pairs are calculated and stored in a profit-matrix which has been applied as a basis for the evaluation of the customers in the selection process.

The set of pick-up customers *I* can be divided into two subsets of elements: the visited customers I_1 that are already in the tour included and the unvisited customers I_2, which have not been inserted in the tour yet. The sum of the two subsets equals to the set of all the pick-up customers. The set of the delivery customers is split up, respectively.

$$\boldsymbol{I = I_1 \cup I_2} \qquad \textbf{(41)}$$

$$\boldsymbol{J = J_1 \cup J_2} \qquad \textbf{(42)}$$

It is assumed if two unvisited customers – one pick-up and one delivery request - are chosen, then the minimum quantity of the two requests can be satisfied. According to this logic, if the pick-up customer *i* and the delivery customer *j* are selected for the insertion, the expected profit value $\mu_{i,j}$ is given as follows:

$$\boldsymbol{\mu_{i,j} = (p_i + p_j)\ \min\ (q_i, q_j) \qquad i \in I_2, j \in J_2} \qquad \textbf{(43)}$$

3.2.1.2 Algorithm S - Serial Insertion Heuristic with separate selection

In the first algorithm the evaluation of pick-up and delivery customers are based on the expected profit values of the unvisited pairs described previously. However, the selection and insertion of one pick-up and one delivery customers occurs separately, in two steps.

First, the selection of the pick-up customer is explained in details. If we take into account all the unvisited pick-up customers, the one with the highest expected profit weighted by the distance factor is selected. More precisely, the greedy value of an unvisited pick-up customer $i \in I_2$ can be given in mathematical formulation as follows.

$$\mu_{i=} \frac{max(\mu_{i,j})}{min(c(i,s))} \qquad i \in I_{2,}, j \in J_{2,}\ s \in I_1 \cup J_1 \qquad (44)$$

The maximum expected profit value is selected out of all the possible unvisited pick-up and delivery pair combination. The value $c\,(i,s)$ represents the distance between the pick-up request *i* and the closest already visited request in the tour. The consideration behind this step is the fact that any newly selected pick-up customer will not be added at the end of the current tour but at the position where the tour extension will be the smallest. Since the distance has an influence on the total profit as cost factor, we consider taking the distance factor into the greedy value as a good strategy.

In order to calculate this distance, the closest already visited request *s* in the current tour is searched as first. At start there is a special situation because no request has been inserted in the tour yet. In this case the distances between the potential request and the starting depot S, and potential request and finishing depot F are recorded and out of the two distances the smaller value is taken as distance weight factor.

$$\mu_{i=} \frac{max(\mu_{i,j})}{min(c(i,s))} \qquad i \in I_{2,}, j \in J_{2,}\ s \in \{S, F\} \qquad (45)$$

The expected profit of an unvisited delivery customer is calculated similarly. The greedy value of certain delivery customer is the maximum expected profit which can be reached by matching it with an unvisited pick-up customer and adding them to the tour. Again, the expected profit value is weighted by the distance weight presented previously. The greedy value μ of an unvisited delivery customer *j* is defined as follows:

$$\mu_{j=} \frac{\mathbf{max}(\mu_{i,j})}{\mathbf{min}(c(j,s))} \qquad i \in I_{2,}, j \in J_2, s \in \{S, F\} \qquad (46)$$

3.2.1.3 Algorithm SP - Serial Insertion Heuristic with pair insertion

In the second insertion heuristic pick-up and delivery customers are selected and inserted as a pair into the tour at the same time. Pairs are added to the tour until no more pairs can be added any more. If a tour has been built, the algorithm goes on building the next tour like in the other serial heuristics. The selection of the pair to be added is based on the expected profit which can be gained by the insertion.

The expected profit of a pair $\mu_{i,j}$ is calculated in the same way as in the previous method:

$$\boldsymbol{\mu_{i,j} = (p_i + p_j)\ \min\ (q_i, q_j) \qquad i \in I_2, j \in J_2} \qquad \textbf{(47)}$$

All the possible pick-up and delivery pair of customers are evaluated according to the Equation (47) and then the pairs are sorted in the candidate list in decreasing order of the expected profit values.

3.2.1.4 Algorithm PP - Parallel Insertion Heuristic with pair insertion

In the Parallel Insertion Heuristic pick-up and delivery customers are added to the tour as pairs. The basis for the selection is given by the same greedy values as at the Serial Insertion Heuristic discussed previously. The only difference between the two heuristics can be found in the building technique of the tours. The tours are being built parallel in the same period of time - after a pair of customers is selected to be added to one tour, the selection of a new pair of unvisited customers will be performed in order to add them to the second tour, etc. It goes on until all the tours have been screened one. When all the tours have been tried to be extended by adding the new customers, the selection process starts again by adding another pair of requests to the first tour.

3.2.2 Selection procedure: Roulette Wheel Selection

One important component of the greedy algorithm applied in the current thesis is the roulette wheel selection. It is a proportional selection process, in which the elements have not the same probability to be selected. Roulette wheel selection - used often as genetic operator in the genetic algorithms – applies a fitness function, which assigns a fitness value to each element. In our model the greedy values are the fitness values which are then used to associate a probability of selection with the customers.

First, the elements are sorted according to the greedy values, giving the list of best candidates. To choose the next customer to be inserted, I applied four versions of the roulette wheel selection:

1. The element with the highest greedy value – the first element in the best candidate list - is selected at every iteration.

2. The 3 best customers or customer pairs are taken from the candidate list and one element or pair is selected randomly out of it. The probabilities of selecting the element or pair *i* is:

$$p_i = \frac{\mu_i}{\sum_{j=1}^{3} \mu_j} \quad \textbf{(48)}$$

3. The 15 best customers or customer pairs are taken from the candidate list and one element or pair is chosen out of it. The probabilities of choosing the element or pair *i* is:

$$p_i = \frac{\mu_i}{\sum_{j=1}^{15} \mu_j} \quad \textbf{(49)}$$

4. One element is selected randomly out of all the elements found in the candidates list. The individuals in the population are all the N elements from the candidate list and the probability of an element to be chosen:

$$p_i = \frac{\mu_i}{\sum_{j=1}^{N} \mu_j} \quad \textbf{(50)}$$

Table 3.1 Representation of the randomised candidate selection in the roulette wheel selection. In the example one pair of customers is randomly chosen out of the 3 best pairs in the sorted candidate list. The probability calculation is based on the expected profit values. (own representation)

Candidate list	Elements	Exp. Profit	Probability p_i	p_i cumulated
1	{2, 4}	1200	p_1=1200/(1200+ 1000 +950)= 0,38	0,38
2	{2, 6}	1000	p_2=1000/(1200+ 100 +950) = 0,32	0,7
3	{3, 5}	950	p_3= 950/(1200+ 1000 +950) = 0,30	1
4	{3, 7}	…	…	…
5	{2, 6}			
6	{4, 5}			
…	….			

	100%		
p_i	38%	32%	30%
P (X ≤ x)	38%	70%	100%
	In case r =0, 5 :	0,5 < 0,7 ↓ Candidate 2 is selected	

During the selection process a random number *r* is generated between 0 and 1. In the following step the restricted candidate list is screened starting from the top and checked where the random number *r* is over the cumulated probability values of the

candidates. The implementation of the whole selection process is summarized in the pseudo-code in the Table 3.2.

Table 3.2 Selection process of the customer pair for insertion (own representation)

```
Procedure 1 Building the candidate list p where all the positive expected profits (μ(a, b ) of pick-up and delivery customers are stored
    Set I2 = set of unvisited pick-up customers, J2=unvisited delivery customers
    For each elements of I2
        Select element a (a ϵ I2 )
        For each elements of J2
            Select element b (b ϵ J2 )
            If μ (a, b) > 0
                Add μ(a, b) to candidate list p
            End If
        End For
    End For
Procedure 2 Sorting the candidate list into descending order according to the profit
    Sort elements of the candidate list p into descending order
Procedure 3 Reduce the candidate list to the size desired (n= 1, 3, 15 or all elements)
    For each element in list p (n number of elements of p, … , 1)
        If i> n
            Erase p(i) from the candidate list p
        End If
    End For
Procedure 4 Calculate the cumulated sums of the expected profit values
    Set the vector Sum and initialise Sum(0) =0
    For each element in candidate list p (i=1,… , n number of elements of p)
        Sum (i) = Sum (i-1) + p(i)
    End For
Procedure 5 Randomised selection procedure
    Generate a random number r:  0 < r <  sum (n)
    For each elements in vector Sum (i=1, …, n)
        If  r  < sum (i)
            Return i
        End If
    End For
```

3.2.3 Insertion of the Selected Customer at Best Position

When the most profitable customer or customers are selected, the aim is to insert them at the best position in the current tour. Therefore, a search will be performed in order to find the location where the insertion results the smallest tour extension. If it is found, it is checked whether the tour length after the insertion is not above the tour length limit D_{max}. If the tour length limit is not violated, the customer or customers are inserted at

the selected position. Otherwise, the selected customer is rejected and the search of a new potential customer will be started.

The pseudo-code of the insertion process of selected customers can be followed in the figure below:

Table 3.3 Pseudo-code of the insertion process of the selected candidate (own representation)

```
1   Given dmin = DBL_MAX as the smallest tour length extension found, the total length of the tour
    d and the insertion position p
2   For all possible positions in the current tour i = 1, ..., n-1
3           Calculate Δ D tour length extension if the customer x is inserted at the position i :
            Δ D = c (vi, x) + c (x, vi+1) - c (vi, vi+1)
4           If d + Δ D  < Dmax
5                   If Δ D < dmin
6                           dmin= Δ D and p = i
7                   End If
8           End If
9   End For
10  If d + dmin < Dmax
11          Insert point at the position p
12          Do 2-opt search
13  Else no insertion is possible
14  End If
```

3.2.4 Optimization of the Pick-up and Delivery Quantities

During the build of the tours, determining the quantities to be loaded is another important component in the insertion heuristic. This problem can be observed as a separate linear programming problem. We have a tour given and each point is associated with a maximum demand which can be satisfied. However, fulfilling the maximum demand would not result the maximum profit and in some cases it would cause infeasible solutions because the load capacity constraints would be violated. Therefore, it is reasonable to observe this problem separately from the route building problem and solve it as a linear problem, which can be defined in the following form:

$$\textbf{max} \sum_{i \in V} p_i y_i \qquad \textbf{(51)}$$

$$\textbf{s.t. } Q_{i+1} = Q_i + q_{i+1} y_{i+1} \qquad \forall i \in V \cup \{0\} \ \textbf{(52)}$$

$$Q_0 = 0 \qquad \textbf{(53)}$$

$$0 \leq Q_i \leq Q_{max} \qquad \forall i \in V \cup \{0\} \ \textbf{(54)}$$

$$0 \leq y_i \leq 1 \qquad \forall i \in V \cup \{0\} \ \textbf{(55)}$$

3.3 Implementation of Improvement Heuristics

The 2-opt local search has been applied in order to improve the current tour after the insertion of one or two customers. Following the insertion process, the 2-opt algorithm is performed at every iteration. In this algorithm two non-adjacent edges of the tour (v_i, v_{i+}) and (v_j, v_{j+}) are replaced by two other edges in a way that the tour can be reconnected again. By the delete of the two edges, a subpath is created which needs to be reversed to reconnect the subpath into one tour. The two new edges (v_i, v_j) and (v_{i+}, v_{j+}) are the only edges which can create a tour again if the original two edges are dropped. As an example, if the edges (v_i, v_j) and (v_{i+}, v_{j+}) are dropped, the subpath (v_{i+}, ..., v_j) is reversed. As a result, the original subpath (v_i, v_{i+}, ..., v_j, v_{j+}) is going to be replaced by (v_i, v_j, ..., v_{i+}, v_{j+}).

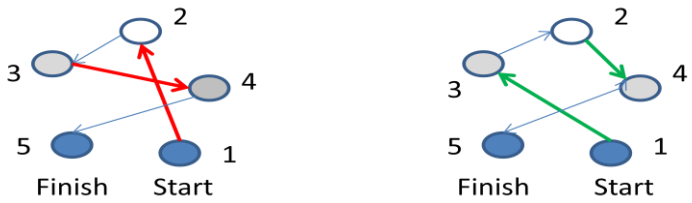


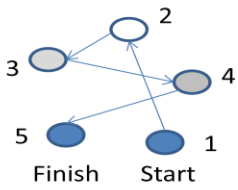
Figure 3.1 The 2-opt exchange of vertices 2 and 3 (own representation)

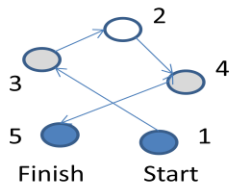
Such exchanges take place only if a decrease of the tour length can be reached. In order to test whether the tour length is going to be reduced by an exchange of customers, it is not necessary to concern the whole path of the tour. The tour is affected only between the locations where the edges are removed. Accordingly, if the vertices i and j are tested for an exchange, the exchange is seen as beneficial if the ΔD = c (v_i, v_j) + c (v_{i+}, v_{j+}) – c (v_i, v_{i+}) – c (v_j, v_{j+}) is smaller than zero.

As a replacement strategy *the first improvement* method has been applied. During testing the possible edge exchanges, as soon as an improvement is found, the 2-opt exchange is performed. After the exchange, the search starts again and lasts until a new possible exchange can be done. The process will be repeated until no more exchange could result a negative value of ΔD.

In the implementation of the search it was taken into account that the exchange may cause infeasibility. Considering the case when the exchange would reduce the tour

length but the load capacity constraint would be violated, the exchange is rejected. The Figure 3.2 below illustrates an example of an infeasible exchange.





Tour	1	2	3	4	5
∑ Load	0	+5	-2	-3	0
Load	0	5	3	0	0

Tour	1	3	2	4	5
∑ Load	0	-2	+5	-3	0
Load	0	-2	3	0	0

Figure 3.2 Infeasible 2-opt exchange (own representation)

If we replace the edges (v_1,v_2) and (v_3, v_4) by (v_1, v_3) and (v_2, v_4), the intersection between the edges is eliminated and the total tour length reduces. However, this 2-opt exchange would lead to infeasibility concerning the transhipped quantity of goods. Under the assumption that the quantities of transhipped goods Q_i remain the same as in the original tour, the load capacity constraint is violated, since $Q_3 < 0$.

In the implementation of the local search algorithm every time when a possible 2-opt exchange is found, it is checked whether the loaded quantities after the 2-opt exchange remains in the interval of 0 and the max load capacity. In case of infeasibility, the 2-opt exchange is rejected. The steps of the algorithm can be followed in the pseudo code in the Table 3.4 below.

Table 3.4 Two-opt local search implemented for the TOPPD (own representation)

	Given a tour $t = (v_1, v_2, \ldots, v_n)$, where the number of elements is n and the loaded quantities
1	$Q= (Q_1, Q_2, \ldots, Q_n)$
2	**Repeat**
3	no_improvement_found= TRUE
4	**For** $i = 1\ldots, n\text{-}2$
5	**For** $j = i+1, \ldots, n\text{-}1$
6	Let $Qnew = Q$
7	Select the edge (v_i, v_j) in the tour t
8	Calculate $\Delta D = c (v_i, v_j) + c (v_{i+1}, v_{j+1}) - c (v_i, v_{i+1}) - c (v_j, v_{j+1})$
9	**If** $\Delta D < 0$
10	Invert order of subsequence (v_{i+1} , v_j): $t_{new}=(v_1, v_2, \ldots, v_i, v_j, \ldots, v_{i+1}, v_{j+1}, \ldots, v_n)$
11	**For** $k = i+1, \ldots, n$
12	Calculate the new load quantity: $Qnew_k= Qnew_{k-1} + q_k y_k$
13	**If** $Qnew_k > Q_{max}$ or $Qnew_k < 0$
14	Break and go to line 19
15	**End If**
16	**End For**
17	Let $t = t_{new}$ *and* $Q=Qnew$
18	no_improvement_found= FALSE
19	**End If**
20	**End For**
21	**End For**
22	**Until** no_improvement_found = TRUE

4. Computational Experiments

4.1 Description of Experiments

The performance of the three methods presented in the previous section has been examined in computational experiments. The focus of the experiments is on the research of four issues: aiming to find the right sampling strategy (4.2.1), the measure and comparison of the quality performance of the different methods (4.2.2), the measure of the influence level of the parameters on the final results (4.2.3) and finally the measure of the improvement of results due to the local search heuristic (4.2.4).

In order to standardise and compare the results with each other, two reference values have been applied. First, the final results are compared to the best-known solutions for the one-vehicle OPPD problem according to the current research results of R. Hartl and M. Romauch (2013). Secondly, the best-found solution for each instance is used as a reference value to compare the solutions of different methods with each other.

The Formula 56 and 57 describe how to determine the best-found solution for given instance. The best-found solution $\boldsymbol{profit^{best}}$ for given instance is the solution with the maximum total profit in the set of solutions including all the results produced by the GRASP algorithms. Besides the best-found solution, the average of the solutions is seen as another important measure to evaluate the methods. This measure is calculated by the sum of all solutions divided by the number of solutions (seeds).

$$\boldsymbol{profit^{best}}\,(\boldsymbol{instance}) = \boldsymbol{max_{seed,method}}\,(\,\boldsymbol{profit}\,(\boldsymbol{seed}, \boldsymbol{instance}, \boldsymbol{method})) \quad \textbf{(56)}$$

$$\boldsymbol{profit^{average}}(\boldsymbol{instance}) = \frac{1}{|\boldsymbol{SEEDS}|} \sum_{\boldsymbol{seed}\in\boldsymbol{SEEDS}}\,(\boldsymbol{profit}\,(\boldsymbol{seed}, \boldsymbol{instance}, \boldsymbol{method})) \quad \textbf{(57)}$$

In order to compare the results when the solutions are given for various instances, the two measures above are standardised: the profit results are referenced to the best-found solution for each instance. Following this step, we sum up the values and the average result of them is calculated. In the formula the expression GROUP represents the set of instances which are included in the same category - providing the basis for the average calculation. The best solution for a group of instances is defined as the average of the total maximum results for instances in the group (Formula 58). Similarly, the average solution for various instances is the average of the averages for each single instance which are in the group (Formula 59).

$$profit^{best}(GROUP) = \frac{1}{|GROUP|} \sum_{instance \in GROUP} max_{seed,method}\left(profit\,(seed, instance, method)\right) \quad \textbf{(58)}$$

$$profit^{average}(GROUP) = \frac{1}{|GROUP|} \sum_{instance \in GROUP} \frac{1}{|SEEDS|} \sum_{seed \in SEEDS} \left(profit\,(seed, instance, method)\right) \quad \textbf{(59)}$$

Apart from the assessment of the maximum and average results another interesting question is how robust the method is, with other words, how large size the range of solutions has. The robustness is measured by the size of the box-plot which represents the distribution of all the solutions that a method produces.

Concerning the maximum number of iterations the methods run, there are two different cases to differentiate. The first case is when the selection process of the customers does not include any randomisation procedure because the customer with the highest greedy value is selected every time (e.g. in case of the greedy nearest neighbour algorithm with best insertion). In this case, it is sufficient to perform one single run of iteration, since the multiple runs of the algorithm do not provide the generation of new results. In all other cases, the methods are run 100 times with the same parameter settings to capture the different outputs of results.

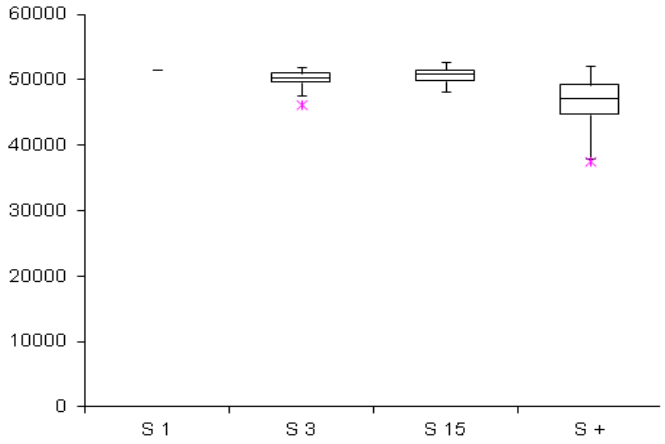


Figure 4.1 Representation of the distribution of solutions due to four different sampling strategies. In every iteration of the method S the next candidate to be inserted is selected out of 1, 3, 15 or all the available not-yet-visited candidates. The result is different outcome of total profits (y-axis). In the example, each method is to build 3 routes in the Chao instance with the parameters Lmax=300 and Dmax=45.

Figure 4.1 serves as an example to show that the size of the solution range can be different due to the randomisation process. In the first sampling strategy there is no randomisation in the selection process. The highest greedy value is selected at every new insertion, thus the set of solutions contains only one unique solution. In the sampling strategy 2, 3, and 4 the randomisation process leads to different outputs of results. The size of the ranges of solutions can reflect how robust the applied method is. As an example, the sampling strategy 4 has reached higher maximum result than the sampling strategy 2. However, the box-plot of the solutions has relatively larger size than the box-plot in the method 2. The large distribution of solutions – including a large amount of low objective values - can be interpreted as an indicator of lower robustness of the given method. The best-found solution was reached by the sampling strategy 3, which is the method S15 in this case.

In the computational analysis we expect to find the answer to the question whether the size of candidates list has a significant influence on the results and whether a certain sampling strategy is preferable in case of TOPPD: The experiments have been run in two well-known benchmark instances with various parameter settings. In the next part, we describe the characteristics of the instances and parameters applied.

4.1.1 Instances

The Tsiligirides and Chao benchmark instances adjusted to the TOPPD were applied to assess the quality of the proposed methods. In both problem families the instances have a certain number of customers (32 customers in the Tsiligirides Set1 and 64 customers in the Chao Set64 instances) allocated in the space categorised according to two constraints: the maximum distance level and the maximum load capacity level. Each instance includes the following information: (a) coordinates of the pick-up and delivery customers (b) coordinates of the depot (start and final location of the tours), (c) the profit values of the customers, (d) the available quantity of the shortages and overages, (e) the capacity of the vehicles, (f) the maximum distance for a vehicle.

4.1.2 Parameter Settings in the Computational Tests

To identify and assess which parameters influence significantly the quality performance of the methods, the computational tests have been run in different parameter settings according to (a) the number of vehicles, (b) the maximum capacity of the loads and (c) the maximum total distance for one vehicle.

Table 4.1 Overview of the parameters (own representation)

Load capacity	Distance limit	Number of vehicles	Instances
Lmax= 60, 120, 300	Dmax= 10, 15, 20, … , 85 *	k=1, 2, 3	Chao and Tsiligirides

* In Chao Set64 instance set Dmax is between 10 and 80

The three methods – method S, SP and PP - have been compared for the Tsiligirides and Chao instance sets with the load capacities of 60, 120 and 300. Within all the three capacity level the methods have been run for 16 different distances limits (14 distance level at Chao instances) and 3 different fleet sizes (k = 1, 2, 3). The data set includes the results of 144 cases for the Tsiligirides instance set – 3 capacity levels, 3 vehicle levels and 16 distance level – and 126 cases for the Chao instance set.

Table 4.2 Number of test cases (own representation)

Instance sets	Lmax	Dmax	k	Total cases
Tsiligirides	3	16	3	144
Chao	3	14	3	126
Total				**270**

In each test case the best-found total profit of the three heuristic methods was taken as reference value in order to compare the results with each other.

4.2 Analysis of the Computational Results

4.2.1 Candidate List in the Randomised Selection Algorithm

At each selection of new customers for insertion, the list of candidates is screened. This list includes 1, 3, 15 or all the elements in decreasing order according to the expected profit values. We examine whether the size of the candidate list plays a significant role on the quality of the generated solutions. First, the maximum and average profits reached

by the three different heuristic methods have been compared to the best-known objective values for the one-vehicle problem.

Table 4.3 The comparison of the best-found and average profit results for the adapted Tsiligirides and Chao instances according to the number of elements which have been randomized in the selection process. The results have been referenced to the best-known results for the 1-vehicle problem.

Method S

	S 1		S 3		S 15		S +	
Instances	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**
Chao	0,89	0,89	1,19	0,94	1,2	0,91	1,15	0,85
Tsiligirides	1,02	1,02	1,24	1,02	1,25	0,98	1,23	0,96
Total Average Score	**0,96**	**0,96**	**1,21**	**0,98**	**1,23**	**0,95**	**1,2**	**0,91**
Average / Max	**1,00**		**0,81**		**0,77**		**0,76**	

Method SP

	SP 1		SP 3		SP 15		SP +	
Instances	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**
Chao	1,12	1,12	1,25	1,13	1,29	1,12	1,22	0,96
Tsiligirides	1,19	1,19	1,26	1,19	1,28	1,15	1,27	1,04
Total Average Score	**1,16**	**1,16**	**1,26**	**1,16**	**1,28**	**1,14**	**1,24**	**1,01**
Average / Max	**1,00**		**0,92**		**0,89**		**0,81**	

Method PP

	PP 1		PP 3		PP 15		PP +	
Instances	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**
Chao	1,14	1,14	1,27	1,13	1,28	1,13	1,20	0,95
Tsiligirides	1,16	1,16	1,24	1,16	1,27	1,13	1,24	1,01
Total Average Score	**1,15**	**1,15**	**1,26**	**1,15**	**1,28**	**1,13**	**1,22**	**0,98**
Average / Max	**1,00**		**0,92**		**0,89**		**0,81**	

It is to note that the maximum objective values are reached when 15 elements are taken into the restricted candidate list for randomisation. The best-found results are improved, when the number of randomised elements increased from 1 element to 15 elements. In case all the elements are involved in the randomisation, the best results were worse on average compared to the cases when the sampling strategy involves the top 15-elements. The other interesting finding is that the quality of the average solutions compared to the best solution falls subsequently when the number of elements increases. In case of the randomisation of 15 or more elements the average solution quality is lower than the average results of a method with a candidate list of 1 element.

The spread of solutions is illustrated in the Figure 4.2 for the adapted Chao instances. The method S had the worst overall performance. Apart from the lowest objective values, the range of solutions is the largest among the three methods. It can be seen, the interval of the 50% of the solutions grows as the number of elements randomised increases and the median of the solutions decreases. The methods SP and PP show similarities, however, the solutions of the method SP are more robust.

To sum up, the randomisation of 15 elements has the best performance in terms of the best-found solutions but we noted that the average solutions had slightly dropped compared to the average solutions of methods applying the randomisation of 3 elements.

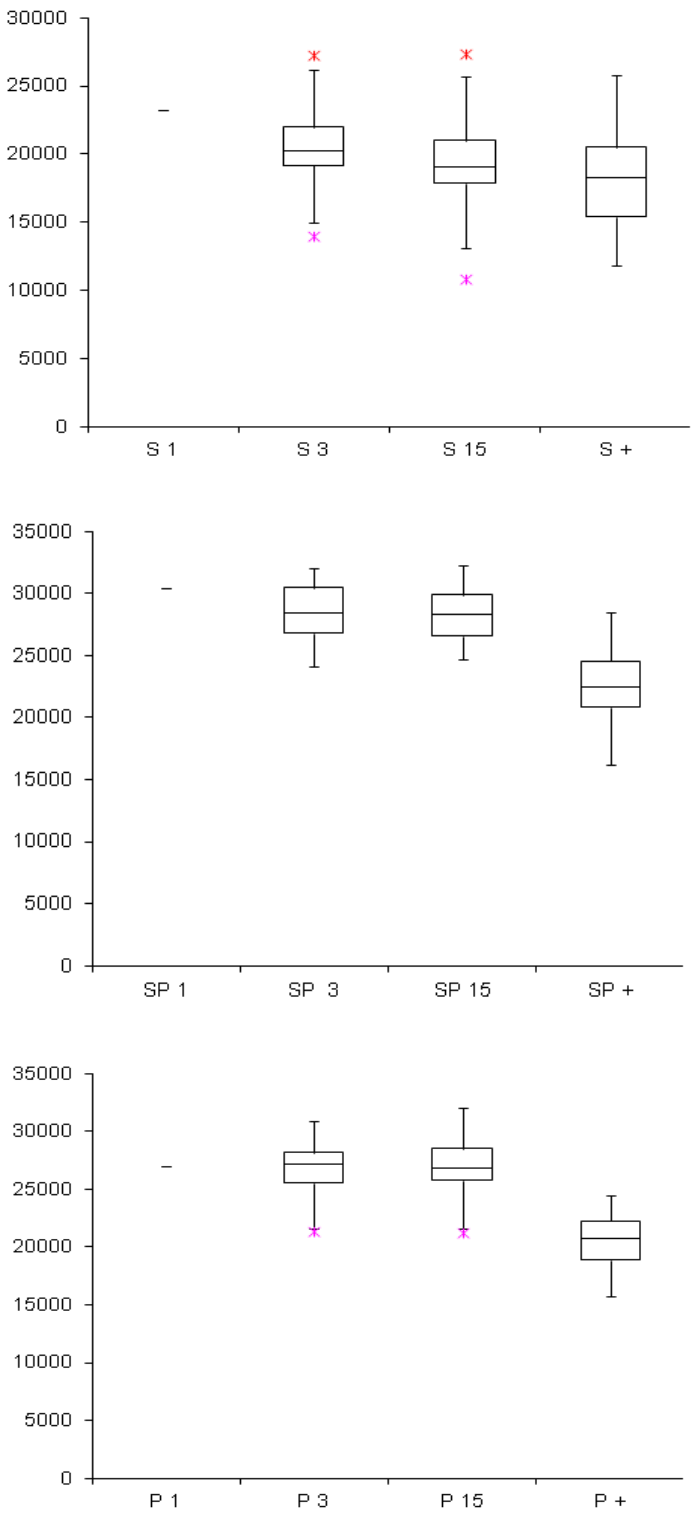


Figure 4.2 The solutions of the methods S, SP and PP with a candidate list of 1, 3, 15 and all unvisited customers for an adapted Chao instance (k=3, Dmax=25, Lmax=120)

4.2.2 Overall Results

The comparison of the averages of the best-found and average objective values can be seen in the Table 4.3. We can see that the method SP and PP outperform the method S in all the fleet-size cases of k=1, 2, and 3.

Table 4.4 The averages of the best-found and average results compared to the best-found results for the adapted Chao and Tsiligirides instances under different fleet-sizes (k=1, 2, 3). In the selection process 15 elements are randomized in each method. (own results)

	S15		SP 15		PP 15	
Instances	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**
Chao	**0.92**	**0.70**	**0.97**	**0.85**	**0.97**	**0.85**
K=1	0.90	0.64	0.96	0.82	0.96	0.82
K=2	0.91	0.71	0.97	0.85	0.97	0.85
K=3	0.93	0.75	0.97	0.87	0.98	0.89
Tsiligirides	**0.95**	**0.76**	**0.97**	**0.88**	**0.97**	**0.87**
K=1	0.93	0.64	0.94	0.81	0.94	0.81
K=2	0.97	0.76	0.98	0.88	0.98	0.86
K=3	0.97	0.87	0.99	0.94	0.98	0.93
Total Average	**0.94**	**0.73**	**0.97**	**0.86**	**0.97**	**0.86**

Concerning the larger sized Chao instance set, the method SP and PP outperform the method S in each category. In terms of the best-found solutions, the two methods scored the same results on average. However, when we look at the average solutions, we can see that the method PP15 performed the best results in the 3-vehicle case. In case of the smaller Tsiligirides instances, the SP 15 is the best-performing method with the best average solutions out of the three methods.

Table 4.5 The averages of the best-found and average objective values in different distance limit categories compared to best-known solutions for the 1-vehicle problem for the adapted Tsiligirides (Dmax=10, … 85) and Chao instances (Dmax=15, …, 80). The load capacity is Lmax=300.

		S15		SP15		PP15	
Instances	**Dmax**	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**
Chao		**1.16**	**0.87**	**1.24**	**1.08**	**1.23**	**1.08**
Set 64	15	**1.38**	0.73	1.34	1.18	1.27	1.14
	20	1.54	0.98	**1.76**	1.62	1.75	1.59
	25	1.37	0.90	**1.58**	1.30	1.55	1.25
	30	1.21	0.81	**1.54**	1.19	1.43	1.12
	35	1.29	0.87	**1.42**	1.18	1.40	1.13
	40	1.15	0.85	1.27	1.12	**1.30**	1.10
	45	1.13	0.92	**1.25**	1.10	1.22	1.06
	50	1.12	0.91	**1.17**	1.03	1.16	1.04
	55	1.06	0.88	1.07	0.94	**1.10**	0.98
	60	1.01	0.91	1.00	0.91	**1.04**	0.96
	65	1.04	0.90	1.06	0.97	**1.07**	0.99
	70	0.97	0.85	1.00	0.90	**1.02**	0.94
	75	0.99	0.89	0.99	0.89	**1.01**	0.94
	80	0.93	0.79	0.89	0.81	**0.97**	0.90
Tsi		**1.24**	**0.98**	**1.28**	**1.15**	**1.27**	**1.13**
Set 1	10	1.55	1.14	**1.56**	1.55	1.56	1.51
	15	1.72	1.21	**1.83**	1.78	1.83	1.79
	20	1.70	1.25	**1.68**	1.43	1.67	1.39
	25	1.38	0.95	**1.61**	1.24	1.61	1.22
	30	1.06	0.75	**1.06**	0.90	1.06	0.90
	35	1.38	0.98	**1.60**	1.31	1.53	1.20
	40	**1.32**	0.99	1.28	1.15	1.26	1.13
	46	1.26	0.99	1.34	1.15	**1.35**	1.11
	50	1.24	0.94	1.20	1.08	1.19	1.07
	55	1.19	0.96	**1.17**	1.08	1.16	1.08
	60	1.08	0.95	**1.08**	1.00	1.07	0.98
	65	1.06	0.93	**1.05**	1.00	1.04	0.99
	70	0.99	0.90	0.99	0.94	**1.00**	0.94
	75	0.99	0.89	0.99	0.93	**0.99**	0.93
	80	1.00	0.87	0.98	0.93	**0.98**	0.92
	85	1.00	0.98	1.00	0.98	**0.99**	0.97
Total Average		**1.20**	**0.93**	**1.26**	**1.12**	**1.25**	**1.11**

In the Table 4.5 the maximum and average total profits referenced to the best-known results for 1-vehicle problems are averaged in each distance limit category. Again, concerning the maximum profits as well as the average profits the method SP 15 outperforms the two other methods by reaching a total average score of 1,26. However, we note that the parallel method performed better than any other methods when there was a medium or large distance limit set in the computational run. The differences in the larger Chao instances are especially remarkable.

In further analysis it was examined how the methods performed when the distance limit as well as the fleet-size vary. In Table 4.6 the minimum gaps to the best-found results can be seen under large load capacity. The SP 15 provides the best results on average for the adapted instance sets. We note that the S15 has poor results for small distance limits, but perform outstandingly well for the large distance set in case 2 and 3 tours were built. When the capacity is limited to the small size of Lmax=60 (see Table 4.6), we can see that the best-performing method for small distances was SP15 and the best one for large distances was PP15. We note that the method SP15 and PP15 performed outstandingly bad under the Dmax=15 and fleet size of k=1. The reason for the worse performance (the size of the minimum gap is 44% here) can be explained by the fact that the single insertion is more beneficial for the very small distances. Comparing the Figure 4.3 and Figure 4.4 we note that due to the separate pair insertion some additional candidates can be inserted in the tour. This solution was not possible in the application of the method SP which result the loss of profits.

Table 4.6 Minimum relative gaps to the best-found solutions of given fleet-size k under large capacity limit of Lmax=300

		S15			SP15			PP15		
Chao	**Dmax**	**k=1**	**2**	**3**	**k=1**	**2**	**3**	**k=1**	**2**	**3**
Set 64	15	**0.0%**	**0.0%**	11.4%	44.1%	**1.0%**	**0.0%**	44.1%	9.8%	4.9%
	20	**1.7%**	18.9%	19.2%	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	2.7%
	25	7.4%	27.7%	13.5%	**0.0%**	**0.0%**	**0.0%**	8.4%	**1.3%**	2.5%
	30	25.6%	25.1%	29.4%	**0.0%**	**0.0%**	**0.0%**	4.1%	2.0%	11.2%
	35	3.2%	21.0%	10.7%	5.2%	**0.0%**	**0.0%**	**0.0%**	2.4%	3.7%
	40	22.5%	25.3%	11.8%	**0.8%**	**0.0%**	**0.0%**	**0.0%**	4.1%	4.2%
	45	29.0%	8.2%	2.8%	**0.0%**	**0.3%**	**0.0%**	**2.0%**	**0.0%**	2.0%
	50	14.1%	9.7%	**1.8%**	**0.0%**	**0.0%**	**0.0%**	**1.3%**	7.2%	**0.1%**
	55	10.1%	**0.0%**	**0.0%**	**0.5%**	**0.8%**	2.7%	**0.0%**	2.3%	**1.8%**
	60	11.5%	**0.0%**	**0.0%**	**0.1%**	**0.0%**	**0.0%**	**0.0%**	**0.1%**	**0.1%**
	65	13.5%	**0.0%**	**0.0%**	**0.0%**	2.0%	2.0%	**1.3%**	2.0%	2.3%
	70	14.6%	**0.0%**	**0.0%**	**0.0%**	**1.8%**	**1.8%**	**2.0%**	**2.0%**	2.1%
	75	8.1%	**0.0%**	**0.0%**	**0.0%**	2.3%	2.3%	**1.6%**	**1.4%**	**1.8%**
	80	5.0%	**0.1%**	**0.1%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.2%**
Tsi	**Dmax**	**k=1**	**2**	**3**	**k=1**	**2**	**3**	**k=1**	**2**	**3**
Set 1	10	**0.0%**	**0.0%**	**0.9%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
	15	**0.0%**	**0.0%**	14.6%	**1.4%**	**0.7%**	**0.0%**	**1.4%**	**0.7%**	**0.0%**
	20	3.4%	**0.0%**	2.4%	**0.0%**	8.0%	**0.0%**	**0.1%**	8.1%	**0.6%**
	25	48.0%	7.8%	11.3%	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
	30	**0.8%**	**1.3%**	**0.0%**	**0.0%**	**0.0%**	**0.6%**	**0.0%**	**0.7%**	**0.2%**
	35	2.4%	21.2%	18.5%	**0.0%**	**0.0%**	**0.0%**	**0.0%**	2.3%	7.8%
	40	**0.0%**	**0.0%**	**0.0%**	3.8%	**1.5%**	3.8%	5.8%	6.2%	4.0%
	46	6.5%	17.4%	**1.6%**	**0.0%**	4.1%	**0.0%**	**0.0%**	**0.0%**	2.3%
	50	**0.0%**	**0.7%**	**0.0%**	14.9%	**0.0%**	**1.0%**	11.9%	3.5%	**1.2%**
	55	**0.0%**	**1.5%**	**0.1%**	7.5%	**0.0%**	**0.0%**	7.5%	**1.8%**	**0.5%**
	60	2.0%	**0.0%**	**0.0%**	**0.0%**	**0.5%**	**0.5%**	5.1%	**0.7%**	**1.1%**
	65	**0.0%**	**0.0%**	**0.0%**	**0.7%**	**0.0%**	**0.0%**	3.0%	**0.2%**	**0.9%**
	70	4.4%	**0.2%**	**0.2%**	3.1%	**0.0%**	**0.0%**	**0.0%**	**0.7%**	**0.6%**
	75	3.4%	**0.0%**	**0.0%**	**0.0%**	**0.2%**	**0.2%**	**0.0%**	**0.3%**	**0.8%**
	80	**0.0%**	**0.0%**	**0.0%**	**0.7%**	**1.9%**	**1.9%**	**0.7%**	2.0%	2.6%
	85	**0.1%**	**0.0%**	**0.0%**	**0.1%**	**0.0%**	**0.0%**	**0.0%**	**0.4%**	**1.4%**

Table 4.7 Minimum relative gaps to the best-found solutions of given fleet-size k under small capacity limit of Lmax=60

		S15			SP15			PP15		
Chao	**Dmax**	**k=1**	**2**	**3**	**k=1**	**2**	**3**	**k=1**	**2**	**3**
Set 64	15	**0.0%**	**0.0%**	11.4%	44.1%	**1.0%**	**0.0%**	44.1%	9.8%	4.9%
	20	**1.7%**	16.1%	14.0%	**0.0%**	**0.0%**	**0.2%**	**0.0%**	**0.0%**	**0.0%**
	25	**0.0%**	29.2%	13.3%	8.8%	**0.0%**	**0.0%**	8.8%	2.1%	**0.7%**
	30	32.0%	30.6%	22.1%	**0.0%**	**0.0%**	**0.0%**	10.9%	18.0%	5.0%
	35	**0.7%**	18.1%	4.7%	**0.7%**	**0.0%**	**0.0%**	**0.0%**	**0.2%**	2.5%
	40	2.6%	16.9%	11.3%	6.4%	9.6%	7.3%	**0.0%**	**0.0%**	**0.0%**
	45	23.5%	7.7%	6.4%	**0.0%**	**1.0%**	**0.0%**	7.1%	**0.0%**	**1.8%**
	50	6.5%	4.4%	7.5%	3.9%	**0.0%**	8.4%	**0.0%**	4.0%	**0.0%**
	55	9.2%	7.2%	8.8%	**0.0%**	5.8%	14.8%	**1.3%**	**0.0%**	**0.0%**
	60	3.1%	**0.0%**	**1.4%**	4.8%	9.8%	11.4%	**0.0%**	**1.2%**	**0.0%**
	65	12.4%	**0.0%**	**0.0%**	**1.1%**	2.2%	2.2%	**0.0%**	**1.5%**	**0.8%**
	70	18.5%	4.0%	7.6%	4.2%	5.0%	8.7%	**0.0%**	**0.0%**	**0.0%**
	75	9.1%	**0.3%**	4.8%	**0.0%**	8.3%	13.2%	**1.9%**	**0.0%**	**0.0%**
	80	8.5%	4.0%	7.4%	10.0%	16.7%	20.6%	**0.0%**	**0.0%**	**0.0%**
Tsi	**Dmax**	**k=1**	**2**	**3**	**k=1**	**2**	**3**	**k=1**	**2**	**3**
Set 1	10	**0.0%**	**0.0%**	**0.9%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
	15	**0.0%**	**0.0%**	14.6%	**1.4%**	**0.7%**	**0.0%**	**1.4%**	**0.7%**	**0.0%**
	20	3.4%	**0.0%**	2.4%	**0.0%**	8.0%	**0.0%**	**0.1%**	8.1%	**0.6%**
	25	48.0%	7.8%	11.3%	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
	30	**0.8%**	**1.3%**	**0.0%**	**0.0%**	**0.0%**	**0.6%**	**0.0%**	**0.7%**	**0.2%**
	35	2.4%	21.2%	18.5%	**0.0%**	**0.0%**	**0.0%**	**0.0%**	2.3%	7.8%
	40	**0.0%**	**0.0%**	**0.0%**	3.8%	**1.5%**	3.8%	5.8%	6.2%	4.0%
	46	6.5%	17.4%	**1.6%**	**0.0%**	4.1%	**0.0%**	**0.0%**	**0.0%**	2.3%
	50	**0.0%**	**0.7%**	**0.0%**	14.9%	**0.0%**	**1.0%**	11.9%	3.5%	**1.2%**
	55	**0.0%**	**1.5%**	**0.1%**	8.0%	**0.0%**	**0.0%**	7.6%	2.1%	**0.5%**
	60	**0.0%**	**0.0%**	**0.0%**	**1.2%**	**0.5%**	**0.5%**	3.0%	**0.6%**	**1.1%**
	65	**0.0%**	**0.0%**	**0.0%**	**0.7%**	**0.0%**	**0.0%**	3.0%	**0.2%**	**0.9%**
	70	4.4%	**0.2%**	**0.2%**	3.1%	**0.0%**	**0.0%**	**0.0%**	**0.7%**	**0.6%**
	75	3.0%	**0.0%**	**0.0%**	2.8%	**0.5%**	**0.5%**	**0.0%**	**0.3%**	**0.8%**
	80	**0.0%**	**0.0%**	**0.0%**	2.6%	**2.0%**	**2.0%**	**0.7%**	2.0%	2.6%
	85	**0.1%**	**0.0%**	**0.0%**	**0.1%**	**0.0%**	**0.0%**	**0.0%**	**0.4%**	**1.4%**

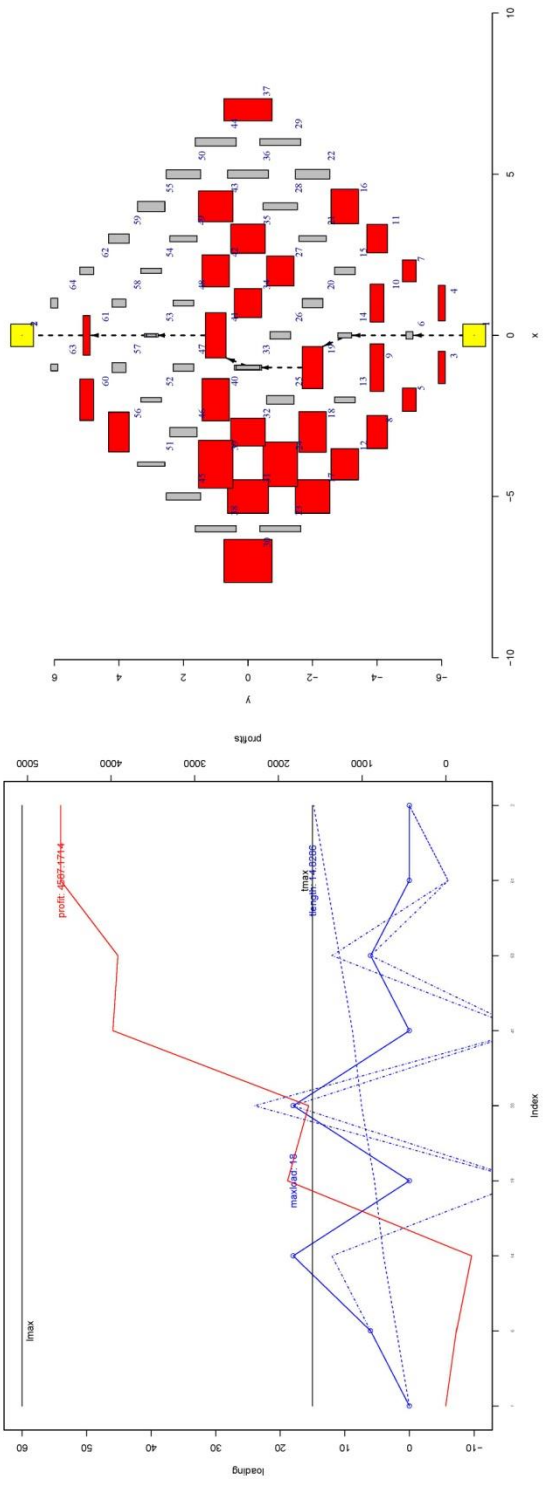


Figure 4.3 Representation of the best-found solution for the smallest distance limit of Dmax=15 by applying the method S 15. In this method one candidate is added to the tour in every iteration.

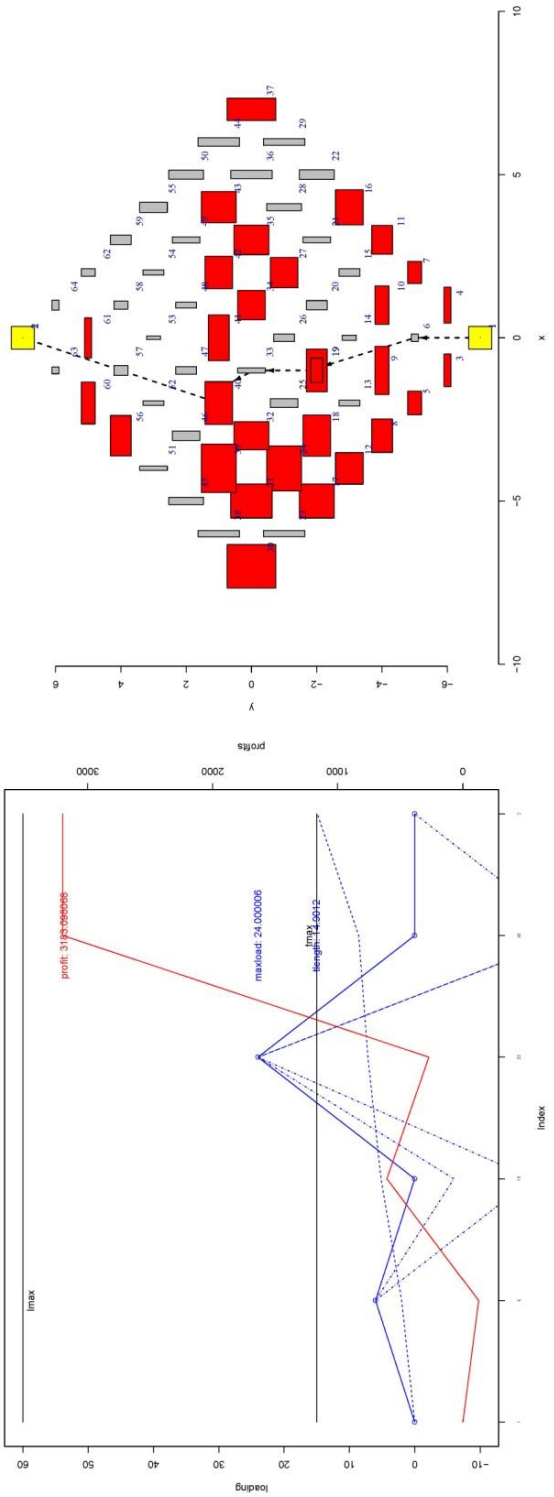


Figure 4.4 Representation of the best-found solution for the smallest distance categor of Dmax=15 by applying the method SP 15. In this method candidates are added in pairs – one pick-up and one delivery customer- to the tour.

Table 4.8 The averages of the best-found and average solutions compared to the best-found solutions with randomisation of 15 element under capacity constraints Lmax= 60, 120 and 300

	S15		SP 15		PP 15	
Instances	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**
Chao	**0.92**	**0.70**	**0.97**	**0.85**	**0.97**	**0.85**
Lmax=60	0.91	0.68	0.94	0.79	0.97	0.81
Lmax=120	0.92	0.70	0.97	0.85	0.97	0.87
Lmax=300	0.92	0.71	0.98	0.90	0.97	0.88
Tsiligirides	**0.95**	**0.76**	**0.97**	**0.88**	**0.97**	**0.87**
Lmax=60	0.95	0.74	0.97	0.87	0.97	0.86
Lmax=120	0.95	0.74	0.97	0.87	0.97	0.86
Lmax=300	0.95	0.79	0.97	0.89	0.97	0.88
Total Average	**0.94**	**0.73**	**0.97**	**0.86**	**0.97**	**0.86**

The performance of the three heuristics has been examined in different load capacity settings. Based on the overall results, the SP15 and the PP15 are found to be the best performing two methods. We note that the PP15 has better performance than the SP15 when the load capacity is on low level.

Table 4.9 The averages of the best-found and average solutions compared to the best-found solutions with randomisation of 15 element under capacity constraints Lmax= 60, 120 and 300 for the Chao instances. Dmax varies between 15-80 for the Chao instance set.

	S15		SP15		PP15	
Chao Set 1	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**
Lmax=60	**0.91**	**0.68**	**0.94**	**0.79**	**0.97**	**0.81**
k=1	0.92	0.63	0.95	0.77	0.95	0.78
k=2	0.90	0.69	0.95	0.79	0.96	0.81
k=3	0.92	0.72	0.94	0.80	0.99	0.85
Lmax=120	**0.92**	**0.70**	**0.97**	**0.85**	**0.97**	**0.87**
k=1	0.90	0.64	0.96	0.82	0.96	0.83
k=2	0.92	0.72	0.98	0.86	0.98	0.87
k=3	0.93	0.75	0.98	0.87	0.97	0.90
Lmax=300	**0.92**	**0.71**	**0.98**	**0.90**	**0.97**	**0.88**
k=1	0.90	0.64	0.97	0.86	0.96	0.85
k=2	0.92	0.73	0.99	0.91	0.97	0.88
k=3	0.94	0.77	0.99	0.92	0.97	0.90
Total Average	**0.92**	**0.70**	**0.97**	**0.85**	**0.97**	**0.85**

Again, the Table 4.8 highlights that the method PP15 performs as best method when the load capacity is low and on the other hand the serial SP15 is the best method when the load capacity is large. When the load capacity is on the moderate level of Lmax=120, the two methods had nearly the same performance on average.

Besides the total profits the run time is the other important performance measure in the evaluation of the proposed methods. Under run time we understand the computational time of one solution. Concerning the overall run times, the method S is the best-performing method with the fastest average run time of 1,30. In case of the methods S and SP, we can see that the parameter *k* – the number of tours - has no significant influence on the run time. Surprisingly, observing the run times of the parallel method, there is a significant improvement of run times when the number of tours increases from 1 to 3. The highest efficiency of the parallel method is reached when 3 tours are built – in this case the calculation time of the solutions is under the level of method SP on average. These results reassure our previous findings – the method PP is the best-performing method when at least 3 tours are built.

Table 4.10 Comparison of the run times under different distance limits from 15 to 80 for the adapted Chao instances under fleet-size k=1, 2, and 3.

		S15			SP15			PP15		
		k=1	2	3	k=1	2	3	k=1	2	3
Dmax	15	**1.23**	**1.24**	**1.24**	3.87	3.87	3.88	5.63	8.13	7.08
	20	**1.23**	**1.23**	**1.23**	5.11	5.12	5.12	8.94	10.21	7.87
	25	**1.28**	**1.28**	**1.28**	5.84	5.84	5.84	10.53	10.89	8.09
	30	**1.21**	**1.21**	**1.22**	6.38	6.38	6.38	12.01	10.90	8.18
	35	**1.27**	**1.28**	**1.28**	8.08	8.08	8.08	15.45	12.45	8.26
	40	**1.22**	**1.22**	**1.23**	9.45	9.45	9.45	20.69	13.70	**8.05**
	45	**1.27**	**1.27**	**1.27**	9.78	9.78	9.78	24.60	14.04	**6.97**
	50	**1.25**	**1.25**	**1.25**	10.40	10.40	10.40	28.42	11.62	**5.85**
	55	**1.38**	**1.38**	**1.38**	11.30	11.30	11.30	31.99	**8.64**	**5.27**
	60	**1.26**	**1.26**	**1.27**	11.06	11.06	11.06	34.24	**6.89**	**5.28**
	65	**1.31**	**1.31**	**1.31**	12.66	12.66	12.67	37.44	**4.59**	**5.47**
	70	**1.34**	**1.34**	**1.35**	12.21	12.21	12.21	39.96	**4.57**	**5.60**
	75	**1.45**	**1.46**	**1.46**	11.35	11.35	11.36	32.80	**4.37**	**5.33**
	80	**1.43**	**1.43**	**1.43**	7.90	7.90	7.90	24.97	**4.83**	**5.76**
Average of Run time		**1.30**	**1.30**	**1.30**	**8.96**	**8.96**	**8.96**	**23.43**	**8.99**	**6.65**

To sum up, we examined the efficiency of the three methods under different settings of parameters, such as fleet-size, distance limit and load capacity. Based on the overall average profit results, the serial heuristic SP15 has been approved to be the best performing method out of the three heuristics. Concerning the other serial heuristic, the method S had poor results for small distance limits. However, this method had an outstanding performance for large distance sets. Furthermore, regarding the computational time this method is significantly better than the other two methods.

On the other hand, we find some special settings when the parallel method PP and higher quality solutions than the method SP and S. Explicitly, the parallel method is the best-performing method in multiple-vehicle cases (k=3) when many customer requests (large-sized Chao instance) need to be satisfied when the load capacity is small (Lmax=60) and the distance limit is large or unconstrained (Dmax $\geq$ 55). Under these settings the best results of PP15 were 1,10 / 1,05= 4,76% better on average than the best-found results of SP15.

Table 4.11 The best-found and average results compared to the best-known solutions for 1 vehicle problem under the settings k=3, Lmax=60 and Dmax is larger than 55

	S15		SP 15		PP 15	
Chao Instances	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**
k=3, Lmax=60, Dmax>=55	**1,09**	**0,98**	**1,05**	**0,97**	**1,10**	**1,05**

4.2.3 Influence of Parameters on the Solution Quality

As next we answer the question whether there are certain parameters which significantly influence the solution quality. The parameters which are studied in the next chapter: the number of vehicles which tranship the goods (4.2.3.1), the maximum load capacity of the vehicles (4.2.3.2) and the maximum available distance per vehicles (4.2.3.3).

4.2.3.1 Number of Vehicles

The number of vehicles as parameter was set to the values k=1, 2 and 3 in the computational runs. First, it is examined how the solution quality changes if the number

of vehicles increases. As reference value the best solution for the one-vehicle problem case is applied.

Table 4.12 Influence of the number of vehicles on the solution quality – The averages of maximal objective values compared to the best-known solutions for the 1-vehicle OPPD for the Chao and Tsiligirides instance sets under different fleet-sizes.

	S		SP		PP	
K	**AVG**	**+ / - Δ**	**AVG**	**+ / - Δ**	**AVG**	**+ / - Δ**
1	0,63		0,71		0,71	
2	1,00	+58%	1,11	+56%	1,07	+52%
3	1,25	+25%	1,35	+21%	1,31	+22%

In all three methods the solution quality has been improved due to the increase of the number of vehicles. The profit improvement is above 50% when the fleet size changes from 1 vehicle to 2 vehicles. Also, a remarkably large increase occurs when the size of vehicles changes from the level of 2 to 3. The positive effect can be observed nearly equally at each method. The highest benefit of the fleet-size increase is gained by the method S. However, the method S remains the worst performing heuristic on average in all the three cases of fleet-size.

4.2.3.2 Maximum Load Capacity

The maximum load capacity was set to the values of 60, 120 and 300. According to the computational results, the Chao and the Tsiligirides instances reacted differently on the changes of the load capacity. Due to an increase of the load capacity from 60 to 300, the average improvement of the best-found results was between 11%-16% in the Chao instance set. On the other hand, the same measure in the Tsiligirides instances at each method was zero. Concerning the solution averages, the improvements were remarkable in the Chao instance set. The average improvement of the solution averages are between 17%-29% with the highest increase recorded at the Serial Heuristic with Pair insertion. The same measure value in the Tsiligirides instances has changed by 4-7%.

Table 4.13 Results with load capacities 60, 120 and 300 compared to the best-found profits under load capacity Lmax=60

15 elements randomized	S 15		SP 15		PP 15	
L max	MAX	AVERAGE	MAX	AVERAGE	MAX	AVERAGE
Chao						
60	1,00	0,79	1,00	0,83	1,00	0,85
120	1,09	0,89	1,12	0,99	1,09	0,98
300	1,12	0,93	1,16	1,08	1,11	1,02
Total Δ (300/60)	**1,12**	**1,17**	**1,16**	**1,29**	**1,11**	**1,21**
Tsiligirides						
60	1,00	0,80	1,00	0,89	1,00	0,88
120	1,00	0,80	1,00	0,90	1,00	0,89
300	1,00	0,86	1,00	0,93	1,00	0,92
Total Δ (300/60)	**1,00**	**1,07**	**1,00**	**1,04**	**1,00**	**1,04**

4.2.3.3 Maximum Available Distance per Vehicle

The influence of the maximum available distance per vehicle is discussed as next. The distances were given as parameters from the unit of size 10 to 85 in the Tsiligirides instances, and from the unit of size 15 to 80 in the Chao instances.

Table 4.14 highlights the effect of the distance limit parameter on the solution quality: the average objective values improve in each of the methods as the distance parameter increases. According to the averages of the best-found and average results, the methods SP and PP outperform the method S in the Chao instances. We note that there is only one category – the smallest distance limit Dmax 15- where the method S is the best performing heuristic. Examining the quality performances of the solutions under different level of distance limits we can see that the method SP performs better for small distance limits. On the other hand the method PP performs outstandingly in the larger distance limits (see the solution results at the Dmax of 45 and above). Furthermore, we can see the averages of the results sink between at the Dmax of 20 and 30 in each method in the Chao instances. The same negative improvement can be found in the Tsiligirides instances between the Dmax 15 and 35. It indicates that the average quality of the results is relatively lower for the small distance limits – resulting lower robustness of methods - than the average results for the large distance limits.

.

Table 4.14 The averages of the best-found and average objective values for the adapted Chao and Tsiligirides instances compared to the best-found solutions (Tsiligirides Dmax=10, Chao Dmax=15)

	S15		SP15		PP15	
	MAX	**AVERAGE**	**MAX**	**AVERAGE**	**MAX**	**AVERAGE**
Chao	**0.92**	**0.70**	**0.97**	**0.85**	**0.97**	**0.85**
15	0.96	0.51	0.89	0.80	0.85	0.77
20	0.89	0.56	1.00	0.93	0.99	0.92
25	0.87	0.56	0.99	0.80	0.96	0.78
30	0.78	0.52	0.99	0.76	0.93	0.72
35	0.91	0.60	0.99	0.82	0.98	0.79
40	0.87	0.64	0.96	0.84	0.98	0.83
45	0.88	0.70	0.99	0.85	0.96	0.83
50	0.94	0.75	0.98	0.86	0.98	0.87
55	0.94	0.77	0.96	0.84	0.98	0.87
60	0.96	0.87	0.96	0.87	0.99	0.92
65	0.95	0.82	0.98	0.89	0.98	0.91
70	0.93	0.81	0.97	0.87	0.98	0.90
75	0.96	0.86	0.96	0.87	0.99	0.92
80	0.95	0.81	0.92	0.84	1.00	0.92
Tsiligirides	**0.95**	**0.76**	**0.97**	**0.88**	**0.97**	**0.87**
10	1.00	0.70	1.00	1.00	1.00	0.98
15	0.96	0.63	0.99	0.97	0.99	0.98
20	0.95	0.70	0.94	0.80	0.94	0.77
25	0.83	0.56	0.99	0.74	0.99	0.73
30	0.90	0.64	0.91	0.76	0.91	0.76
35	0.84	0.58	0.96	0.78	0.93	0.73
40	0.98	0.72	0.96	0.85	0.94	0.84
46	0.91	0.70	0.97	0.82	0.98	0.80
50	0.98	0.72	0.93	0.84	0.93	0.83
55	0.99	0.79	0.97	0.89	0.97	0.89
60	0.98	0.85	0.98	0.89	0.96	0.88
65	1.00	0.87	1.00	0.94	0.99	0.93
70	0.98	0.90	0.99	0.94	1.00	0.94
75	0.99	0.89	0.99	0.93	1.00	0.93
80	0.99	0.86	0.97	0.92	0.97	0.91
85	1.00	0.98	1.00	0.98	0.99	0.97

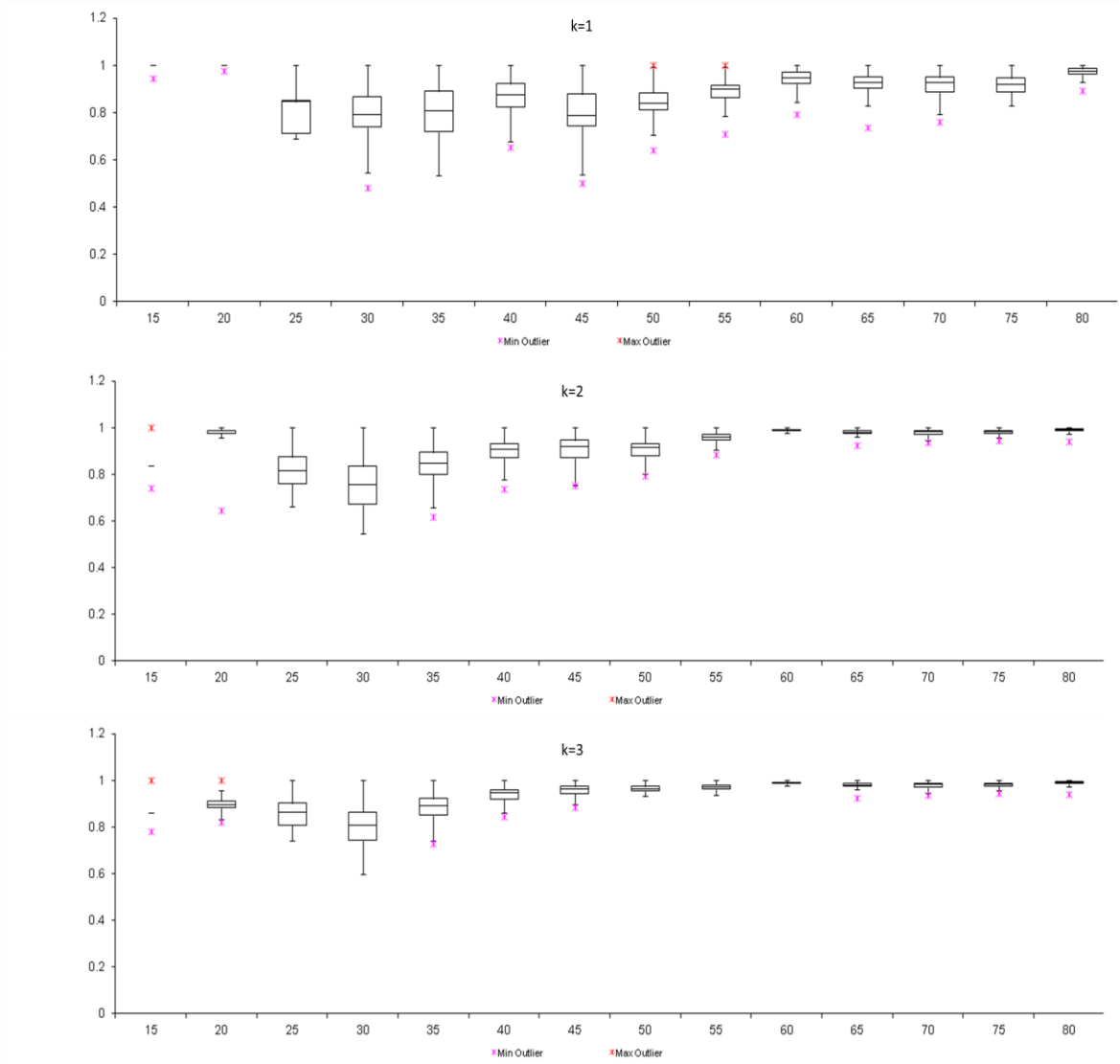


Figure 4.5 The distribution of solutions of the serial method SP15 examined under different distance limits (Dmax= 15, 20, …, 80) and fleet size (k=1, 2, 3) for the adapted instance set of Chao. Load capacity is set to the largest level (Lmax=300).

The influence of distance limit was measured by box plots which present us the distribution of solutions under different distance levels. In the Figure 4.5 we can see how the ranges of solutions decrease due to the fleet size and distance limit. We can recognize a U-formed shape of the box-plots in all three cases of k. In the first stage, the range of results increases due to the increase of distances. In the second stage, starting from the middle-sized Dmax=30, the range of solutions decreases continuously. The decrease is even more remarkable as the fleet of size increases.

Summarizing the findings we state that the distance limit parameter has a significant influence on the performance of the methods. The average of the results increases significantly if the distance limit is extended. Furthermore, the tour length can determine which tour building technique is preferable: the parallel method is performing better if the tour length is large, on the other hand, the serial SP15 is the better one if the tour length is small.

4.2.4 Measure of Improvement due to the Local Search (LS) Procedure

One important component of the GRASP algorithms is the continuous local search procedure after every insertion of a newly inserted customer or customer pair. The aim of the local search procedure is to shorten the tour without causing any loss of profits and infeasibility by the inversion of subtours in the route (e.g infeasibility occurs when the load capacity constraint is violated due to the subtour reversion). As result of the reduction of tour length we expect that the objective values increase. However, the benefit of the continuous local search is not guaranteed – we find cases, in which the overall total profit is lower when the insertion procedure is followed by the local search procedure. It is an interesting question to study how much influence the LS has on the quality performance of the proposed methods. Thus, we made computational experiments, in which the proposed methods have been run with and without applying the local search procedure.

The Figure 4.6 presents how the methods with LS compared to the methods without LS perform. We can see that the improvement of the results due to LS is not significant on the results of the smaller-sized Tsiligirides instance set. The parallel P15 is the only method, which has a one-time increase of 10% of the total profits when LS was applied. On the other hand, the results on the Chao instance set show the LS has a positive impact on the solution quality. The improvement is significant when the distance constraint is small or middle-sized.

To sum up, the practice shows that the application of LS does not provide better results in each single case. However, we note that if the number of customers is high – like in the large-sized Chao instance sets - and the distance limit of vehicles is strictly

constrained, the LS has a significant influence on the quality results, reaching an improvement of up to 12%.

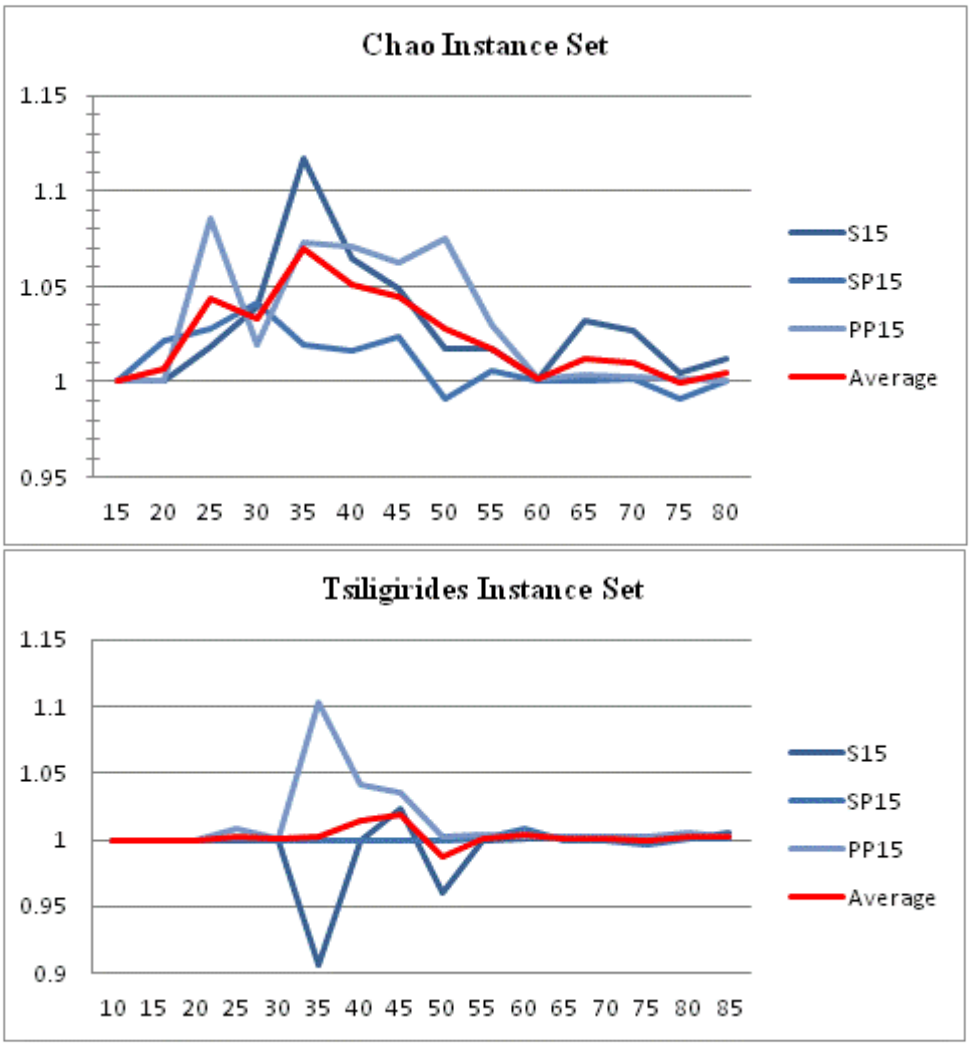
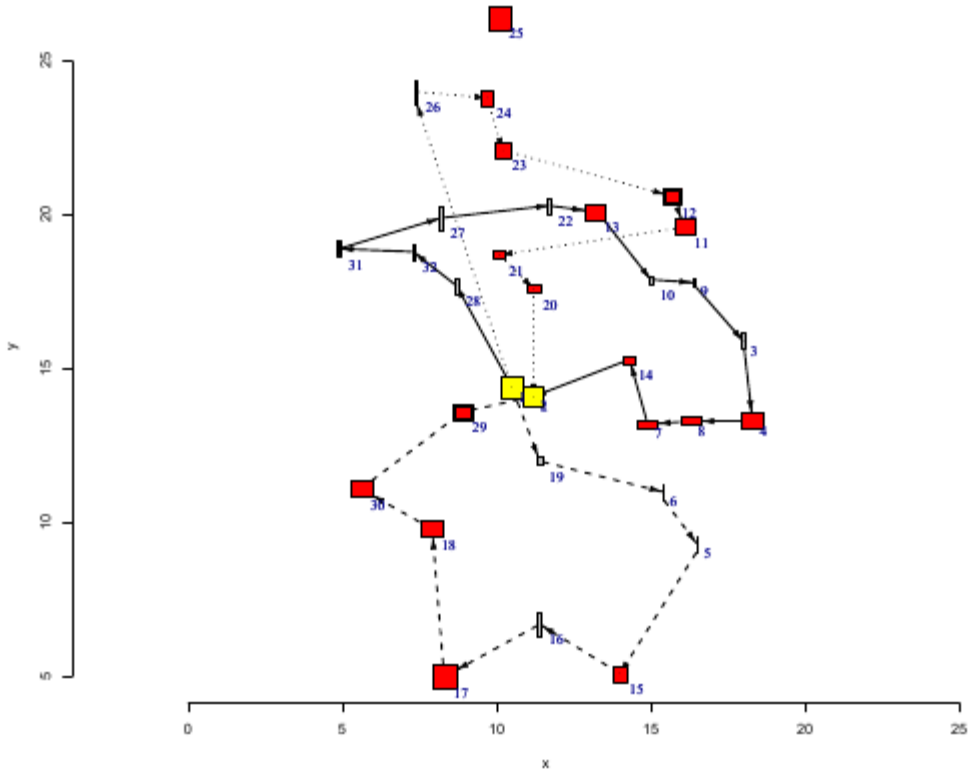
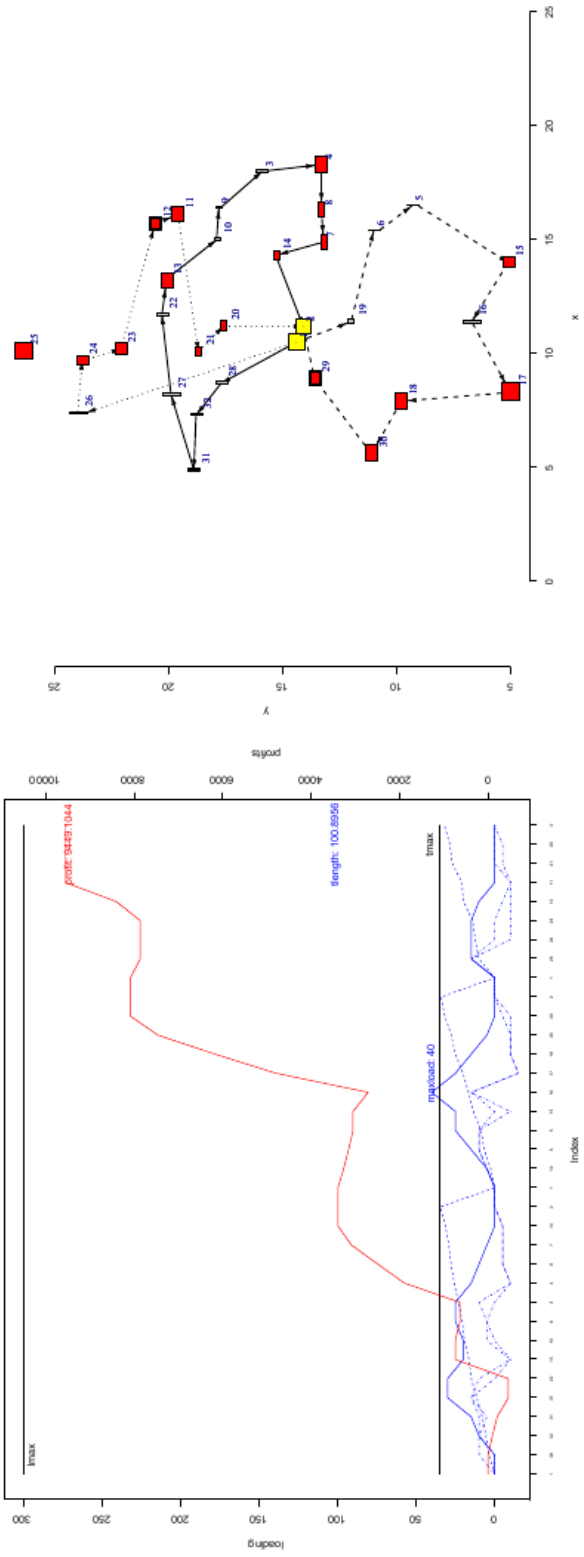


Figure 4.6 The figures illustrate the change of results due to the Local Search (LS) improvement heuristic. Solutions of methods applying LS are compared to the solutions of methods without LS. The tests were run in the Tsiligirides and Chao instance sets under large capacity constraint (Lmax=300) and vehicle size of k=3.

We notice how large the difference between the solution quality when we apply the heuristic S15 for the Tsiligirides instance under the distance limit of 35. In the graph we can see that the profit level at the best-found solution without LS is 10% higher than the best-found solution with LS: The result is surprising because we expect that the LS has a positive effect on the profit level because the improvement of the tour length due to the continuous 2-opt exchanges can enable us to satisfy more customer requests.

However, we find that in some cases the benefit of the LS to the first tour means a disadvantage to the other tours at the same time. The example in Figure 7 shows that the first tour has been positively improved due to the 2-opt LS procedure. Concerning the second tour, the LS has no effect, the tour and the profit level are the same in both cases, with and without LS. However, we notice that the last tour built by the S15 with LS reaches only a profit of 1437 compared to the tour built by the heuristic without applying LS, in which the profit level was at 4010. According to our findings, due to the LS some additional pick-up stations can added to the first tour, resulting higher profits in total. We can see that certain pick-up stations are not included in the third tour, which results a lower level of capacity utilization in the end. In the graph it can be seen that the load capacity peaks at a lower level when the 2-opt LS is applied. To sum up, this example shows that applying the LS there is the possibility to decrease the tour length and as a positive result we might add further customer requests to the tour. However, it does not guarantee the better balance of the pick-up and delivery requests, which can cause that the heuristic method might have a worse overall performance when there is more than one tour built.


25
26
24
23
12
11
22
13
27
31
32
21
20
28
10
9
3
14
29
19
30
6
18
5
16
17
15
0
5
10
15
20
25
x
y


profit:94491044
length: 100.8956
tmax
maxload: 40
loading
Index
profits
x
y

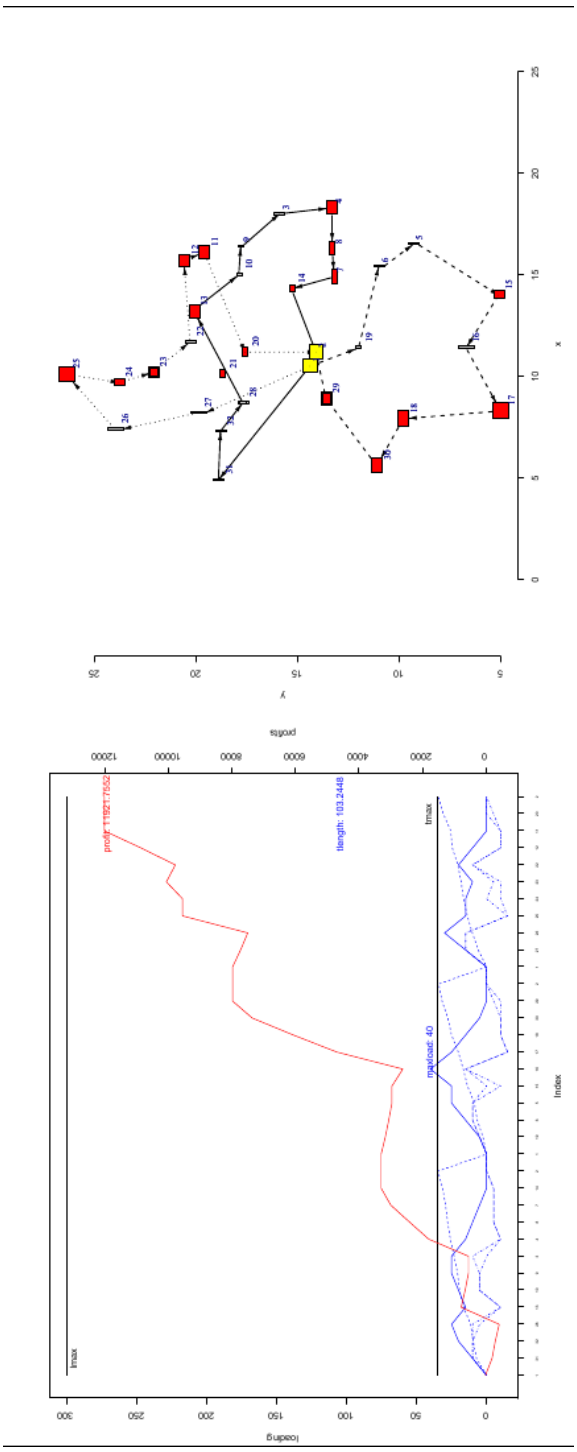


Figure 4.7 The heuristic S15 with and without LS were applied and the solution qualities of the two methods were compared. First, the best-found solution of S15 without LS can be seen under the parameters of k=3, Dmax=35 and Lmax=300. Secondly, the same solution without LS can be seen. The example shows that the improvement of LS made in the first tour had a negative impact on the profit of the third tour.

5. Conclusion

In the current thesis the model of TOPPD was presented and classified among the Vehicle Routing Problems. We proposed three different heuristics as solution methods to the problem. The main focus was to create algorithms which provide high quality solutions – maximising the total profits by satisfying the pick-up and delivery customer requests and minimising the costs which are associated with the transport.

All proposed methods apply a greedy randomised algorithm as insertion heuristic, which was followed by a 2-opt-local search procedure as improvement heuristic. The solution algorithms differ from each other in two important aspects: firstly, according to the tour building techniques, serial or parallel sequences have been applied; secondly, the number of inserted customers is variable in the methods– in the method S one single customer, in the methods SP and PP one pair of customers is inserted into the tours during the iterations.

The quality performance of the algorithms was measured by computational experiments. The results provided evidence on the efficiency of methods applying the insertion heuristic of paired-customers (pick-up and delivery customers). We found that the serial building technique has better performance than the parallel one on average. However, we noted that the parallel method has an outstanding performance for many-tour cases under small load capacity. Another interesting finding is that in the unconstrained case – if there is no load capacity and distance limit given - the serial method with single customer insertion guarantees the best solutions. In terms of computational times we showed that the insertion of pair of customers into the route is more time-demanding than methods applying single customer insertion.

Having studied the effect of various parameters on the solution quality it was found that the distance limit and the fleet-size are the main factors which have significant impact on the quality of the average solutions. Furthermore, under large distance limit and fleet-size the results had higher level of robustness. Finally, we examined the influence of the local search procedure. We noted that the efficiency of the search procedure was influenced by the instance size: in the large-sized instances the local

search showed significant positive impact on the results, in the smaller-sized instances this effect was not recognisable.

The proposed methods provided an efficient approach to the TOPPD problem. An interesting aspect for further research is the extension of the proposed methods, in which we allow that customers are visited more than once within the same route or different routes. For small capacity limits we think there is the possibility it would lead to higher profit results.

The TOPPD has several extensions for future research. We consider a new variant of problem, in which the customers with overages are not anymore associated as pick-up stations and the customers with shortages as delivery stations. In such case the delivery of goods between customers with overages is also profitable since the overall cost reduction in the supply chain is reached due to the reduced inventory costs. As result it would open new aspects in the optimization of loading instructions.

LITERATURE

B.L. Golden, L. Levy, R. Vohra. (1987). The orienteering problem. *Naval Research Logistics, 34,* 307-318.

C. Rego, F. Glover. (2002). *Local Search and Metaheuristics.* In Guten and Punnen.

D. Feillet, P. Dejax, M. Gendreau. (2005). Traveling Salesman Problems with Profits. *Transportation Science, 39*(2), 188-205.

D. Pisinger, S. Ropke. (2010). *Handbook of Metaheuristics.* Springer US.

E. L. Lawler, J.K. Lenstra, A.H. Kan, D. B. Shmoy . (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* Wiley.

F. A. T. Montané, R. D. Galvao. (2002). Vehicle Routing Problems with Simultaneous Pick-up and Delivery Service. *OPSEARCH, 39*(1), 19-32.

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

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

H. Hernandez-Perez, J-J Salazar-Gonzalez. (2003). Heuristics for the One-Commodity Pickup-and-Delivery Traveling Salesmen Problem. *Transportation Science, 38(2),* 245-255

J. Y. Potvin, J.M. Rousseau. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research, 66*(3), 331-340.

L. Bodin, B. Golden. (1981). Classification in vehicle routing and scheduling. *Journal Networks, 11*(2), 87-108.

M. I. Hosny, C. L. Mumford. (2012). Constructing initial solutions for the multiple vehicle pickup and delivery problem with time windows. *Journal of King Saud University - Computer and Information Sciences, 24*, 59-69.

M. Rainer-Harbach, P. Papazek, B. Hu,G. R. Raidl. (2013). Balancing Bicycle Sharing Systems: A Variable Neighborhood Search Approach. In *Evolutionary Computation in Combinatorial Optimization* (old.: 121-132). Springer.

P. Prosser, P. Shaw. (1996). Study of Greedy Search with Multiple Improvement Heuristics. *Department of Computer Science Research Report 96/201*.

Q. Lu, M.M. Dessouky. (2006). A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research, 175*(2), 672-687.

R. Hartl, M. Romauch. (2013). The Influence of Routing on Lateral Transshipment. *Computer Aided Systems Theory,* 267-275. Springer.

S. Anily, R. Hassin. (1992). The Swapping Problem. *Networks, 22*, 419-433

S.N. Parragh, K.F. Doerner, R. F. Hartl. (2008). *Journal für Betriebswirtschaft, 58*(1), 21-51.

T. A. Feo, M. G.C. Resende. (1995). Greedy Randomised Adaptive Search Procedure. *Journal of Global Optimization, 6*, 109-134.

APPENDIX

Appendix 1 Best-known solutions for the 1-vehicle OPPD applied as reference values

		Objective values		
Chao	**Dmax**	**Lmax=60**	**120**	**300**
Set 64	15	4587.17	4587.17	4587.17
	20	11326.20	11326.20	11326.20
	25	14987.00	15167.40	15167.40
	30	17982.10	18294.10	18294.10
	35	21889.00	23833.10	23833.10
	40	26300.10	27584.10	27956.40
	45	32901.20	34863.60	35151.60
	50	32458.10	34864.20	35494.10
	55	36371.10	39113.10	39227.30
	60	40554.20	42876.30	43590.30
	65	48859.10	50293.40	50797.80
	70	51326.30	53288.10	54002.10
	75	46989.80	49869.30	50487.30
	80	41812.00	44956.00	45768.10
Tsiligirides	**Dmax**	**Lmax=60**	**120**	**300**
Set 1	10	560.24	560.24	560.24
	15	1205.98	1205.98	1205.98
	20	1430.73	1430.73	1430.73
	25	4195.79	4195.79	4195.79
	30	4820.44	4820.44	4820.44
	35	5875.32	5875.32	5875.32
	40	5866.66	5866.66	5866.66
	46	6274.14	6274.14	6274.14
	50	7740.26	7740.26	7740.26
	55	8085.02	8245.12	8245.12
	60	8260.02	8260.02	8260.02
	65	8000.02	8000.02	8000.02
	70	7685.39	7685.39	7685.39
	73	10007.60	10007.60	10007.60
	75	10245.90	10245.90	10245.90
	80	10090.80	10090.80	10090.80
	85	7886.46	7886.46	7886.46

Appendix 2 Best-found solutions of the methods S15, SP15 and PP15 for the adapted Chao and Tsiligirides instances under different distance and vehicle-size settings for large load capacity

Lmax=300

		S15			SP15			PP15		
Chao	**Dmax**	**k=1**	**2**	**3**	**k=1**	**2**	**3**	**k=1**	**2**	**3**
Set 64	15	4587.17	7062.18	7407.38	3183.1	6990.27	8253.47	3183.1	6432.36	7869.42
	20	10306.1	17642.31	24354.65	10486.2	20984.3	29040.51	10486.2	20984.3	28272.42
	25	13049	20392.36	28965.03	14015	26038.8	32889.46	12929	25714.3	32091.28
	30	13218	24612.7	29718.22	16596.5	30792.5	38466.99	15936.2	30174.2	34602.9
	35	21955	30177.13	40923.3	21547	36506.6	45297.06	22663	35654.6	43677.4
	40	22382.2	33670.2	43350.72	27200.1	42179.4	48462.68	27428.1	40504.3	46525.1
	45	26409.1	43650.1	51375.3	34059.2	47076.3	52810.3	33399.2	47209.4	51772.6
	50	29560.1	44012.4	48206.4	33730.1	48284.9	49090.73	33304.7	45051.4	49030.4
	55	33005.2	48424.6	49309.93	36155.6	48028.4	48014.4	36347.6	47350.8	48425.9
	60	36324	47321.9	47307.9	40452.1	47314.12	47300.12	40506.2	47291.9	47241.5
	65	43381.3	58569.9	58555.9	49219.1	57420.79	57406.79	48607.4	57402.7	57233.4
	70	45614.2	58764.3	58750.3	52262.3	57701.92	57687.92	51242.3	57637.3	57562
	75	46401.8	53546.5	53532.5	50181.3	52334.38	52320.38	49372.1	52816.2	52588.6
	80	42688.9	45674.26	45660.26	44830	45713.67	45699.67	44818.1	45704.1	45623.4
Tsi	**Dmax**	**k=1**	**2**	**3**	**k=1**	**2**	**3**	**k=1**	**2**	**3**
Set 1	10	560.239	1027.354	1017.457	560.239	1027.354	1026.592	560.239	1027.354	1026.592
	15	1205.98	2232.01	2798.282	1189.72	2215.75	3206.931	1189.72	2215.75	3206.931
	20	1245.52	2655.92	3377.386	1287.97	2458.88	3460.1	1287.03	2457.94	3440.181
	25	2835.02	6455.41	8137.73	4195.79	6956.1	9055.8	4195.79	6956.1	9052.84
	30	3520.61	5595.91	6186.14	3550.11	5666.9	6150.458	3550.11	5627.92	6171.81
	35	4935.81	8551.42	10801.85	5051.92	10368.13	12797.12	5051.92	10131.11	11867.66
	40	5505.45	8190.19	9566.49	5305.06	8071.29	9212	5201.54	7715.56	9195.1
	46	5484.17	8132.36	10088.21	5840.2	9164.07	10245.74	5839.9	9543.39	10011.08
	50	7565.44	10423.07	10866.61	6585.05	10495.68	10755.54	6760.2	10141.57	10742.18
	55	7575.72	10721.83	10921.52	7050.23	10881.71	10933.25	7050.23	10686.11	10881.18
	60	7590.38	9629.86	9629.098	7745.19	9579.44	9578.678	7370.24	9564.25	9521.4
	65	7741.47	8817.53	8816.768	7686.16	8815.55	8814.788	7515.45	8803.31	8734.34
	70	7330.84	7729.95	7729.188	7425.88	7742.257	7741.495	7655.06	7691.15	7693.25
	75	9705.15	10317.55	10316.79	10030.4	10292.16	10291.4	10030.9	10287.06	10237.47
	80	9665.28	10252.14	10251.38	9601.33	10061.01	10060.25	9595.93	10049.69	9993.12
	85	7870.96	7867.458	7866.697	7871.45	7870.688	7869.927	7881.57	7842.05	7759.45

Appendix 3 Best-found solutions of the methods S15, SP15 and PP15 for the adapted Chao and Tsiligirides instances under different distance and vehicle-size settings for small load capacity

Lmax=60

		S15			SP15			PP15		
Chao	**Dmax**	**k=1**	**2**	**3**	**k=1**	**2**	**3**	**k=1**	**2**	**3**
Set 64	15	4587.17	7062.18	7407.38	3183.10	6990.27	8253.47	3183.10	6432.36	7869.42
	20	10306.10	17642.31	24354.65	10486.20	20480.41	27708.43	10486.20	20480.41	27768.42
	25	13049.00	19516.36	28533.03	11993.40	25216.80	32319.46	11993.40	24694.00	32079.00
	30	12576.00	21918.05	29154.22	16596.50	28627.00	35587.09	14970.20	24264.55	33876.78
	35	18289.00	26307.13	38296.00	18289.20	31076.10	40113.16	18421.00	31016.20	39135.40
	40	18794.00	29674.30	37020.56	18134.10	31666.50	38390.25	19286.50	34702.20	41196.70
	45	24549.10	38106.20	45911.25	30321.20	40596.20	48838.02	28299.10	41022.10	47955.20
	50	25222.20	37587.40	42043.63	25840.10	39236.60	41689.52	26854.10	37725.00	45181.80
	55	25709.20	39862.10	42623.54	28067.10	40391.80	40410.92	27719.10	42718.90	46388.90
	60	32322.40	45912.20	45895.77	31782.30	41809.20	41795.20	33318.10	45358.90	46540.10
	65	39745.00	55524.10	55510.10	44191.30	54319.30	54305.30	44689.20	54730.10	55094.90
	70	38372.50	51427.10	51413.10	43610.40	50936.34	50922.34	45452.50	53481.00	55339.40
	75	37485.00	48661.60	48647.60	40911.70	45071.31	45057.31	40146.60	48801.70	50990.10
	80	32944.10	39397.57	39383.57	32506.30	35111.81	35097.81	35746.30	40972.30	42313.70
Tsi	**Dmax**	**k=1**	**2.00**	**3.00**	**k=1**	**2.00**	**3.00**	**k=1**	**2.00**	**3.00**
Set 1	10	560.24	1027.35	1017.46	560.24	1027.35	1026.59	560.24	1027.35	1026.59
	15	1205.98	2232.01	2798.28	1189.72	2215.75	3206.93	1189.72	2215.75	3206.93
	20	1245.52	2655.92	3377.39	1287.97	2458.88	3460.10	1287.03	2457.94	3440.18
	25	2835.02	6455.41	8137.73	4195.79	6956.10	9055.80	4195.79	6956.10	9052.84
	30	3520.61	5595.91	6186.14	3550.11	5666.90	6150.46	3550.11	5627.92	6171.81
	35	4935.81	8551.42	10801.85	5051.92	10368.13	12797.12	5051.92	10131.11	11867.66
	40	5505.45	8190.19	9566.49	5305.06	8071.29	9212.00	5201.54	7715.56	9195.10
	46	5484.17	8132.36	10088.21	5840.20	9164.07	10245.74	5839.90	9543.39	10011.08
	50	7565.44	10423.07	10866.61	6585.05	10495.68	10755.54	6760.20	10141.57	10742.18
	55	7420.72	10721.83	10921.52	6870.19	10880.22	10933.25	6895.31	10656.30	10881.18
	60	7590.38	9625.75	9624.99	7500.19	9579.44	9578.68	7370.24	9564.25	9521.40
	65	7741.47	8817.53	8816.77	7686.16	8815.55	8814.79	7515.45	8803.31	8734.34
	70	7330.84	7729.95	7729.19	7425.88	7742.26	7741.50	7655.06	7691.15	7693.25
	75	9705.15	10317.55	10316.79	9731.35	10270.58	10269.82	10000.20	10287.06	10237.47
	80	9665.28	10252.14	10251.38	9416.33	10054.52	10053.76	9595.93	10049.69	9993.12
	85	7870.96	7867.46	7866.70	7871.45	7870.69	7869.93	7881.57	7842.05	7759.45