universität wien

MASTERARBEIT

Titel der Masterarbeit

„A General Variable Neighbourhood Search Algorithm for the Multi-Vehicle Profitable Pickup and Delivery Problem“

Verfasst von

Murat Küçüktepe

angestrebter akademischer Grad

Master of Science (MSc)

Wien, 2014

Studienkennzahl lt. Studienblatt:	A 066 920
Studienrichtung lt. Studienblatt:	Masterstudium Quantitative Economics, Management and Finance
Betreuer:	O. Univ. Prof. Dipl.-Ing. Dr. Richard F. Hartl

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Die Arbeit wurde bisher weder in gleicher oder ähnlicher Form verfasst, noch einer anderen Prüfungsbehörde vorgelegt.

Innsbruck, September 2014

Murat Küçüktepe

Acknowledgement

I would like to thank the staff of the Chair of Production and Operations Management at the University of Vienna for their great assistance and support. In particular, I would like to express my sincere gratitude to my supervisor O. Univ. Prof. Dipl. - Ing. Dr. Richard F. Hartl for giving me opportunity to write my thesis at his department and leading my work, Dr. Margaretha Gansterer for her precious suggestions and critical reviews, Dr. Fabien Tricoire for introducing me to the Metaheuristics world, Dr. Günther Kiechle for giving me insights of the programming language C++.

Moreover, I am thankful to Univ. Prof. Dr. Walter Gutjahr for his aspiring guidance throughout my master study and Assist. Prof. Mag. Dr. Privatdoz. Stefan Haller for deepening my Algebra knowledge.

I would also like to thank three most valuable women in my life; my girlfriend Rukiye for her candid and continuous support during my study, my mother for motivating me and my sister for her loud and cheerful laughter.

Innsbruck, September 2014

Murat Küçüktepe

Contents

List of Tables

List of Figures

List of Abbreviations

BI	Backward Insertion
C1	Greedy Construction Heuristic
C2	Two-Stage Cheapest Insertion Heuristic
CFRS	Cluster-First-Route-Second
CPU	Central Processing Unit
CVRP	Capacitated Vehicle Routing Problem
DARP	Dial-A-Ride-Problem
DCVRP	Distance-Constraint Vehicle Routing Problem
FI	Forward Insertion
GCE	Gravity Centre Exchange
GPDP	General Pickup and Delivery Problem
GVNS	General Variable Neighbourhood Search
IDE	Inter Dummy Exchange
ITE	Inter Tour Exchange
ITI	Inter Tour Insert
KP	Knapsack Problem
M-M-Problems	Many-to-Many Problem
MVPPDP	Multi-Vehicle Profitable Pickup and Delivery Problem
OP	Orienteering Problem
PBE	Pairwise Backward Exchange
PCTSP	Prize-Collecting Traveling Salesman Problem
PDP	Pickup and Delivery Problem
PDTSP	Pickup and Delivery Traveling Salesman Problem
PDVRP	Pickup and Delivery Vehicle Routing Problem
PFE	Pairwise Forward Exchange
PTP	Profitable Tour Problem
PVRP	Profitable Vehicle Routing Problem
RFCS	Route-First-Cluster-Second
RP	Relocate Pairs
Sa-VND	Self-Adaptive Variable Neighbourhood Descent
SDARP	Single Vehicle Dial-A-Ride-Problem
Seq-VND	Sequential Variable Neighbourhood Descent
SPDP	Single Pickup and Delivery Problem
TSP	Traveling Salesman Problem
TSPP	Traveling Salesman Problem with Profits
VND	Variable Neighbourhood Descent
VNS	Variable Neighbourhood Search
VRP	Vehicle Routing Problem
VRPB	Vehicle Routing Problem with Backhauls
VRPBTW	Vehicle Routing Problem with Backhauls and Time Windows
VRPPD	Vehicle Routing Problem with Pickup and Deliveries
VRPPDTW	Vehicle Routing Problem with Pickup and Deliveries and Time Windows
VRPTW	Vehicle Routing Problem with Time Windows

Zusammenfassung

Da der Vertriebsmarkt unter starkem Konkurrenzdruck arbeitet und sehr schnell expandiert, sind kleinere Logistikunternehmen, die ein beschränktes Transportvolumen haben, gezwungen, nur diejenigen Kunden auszuwählen, die höhere Einnahmen bringen. Um die Transportkosten zu vermeiden, wird die Zusammenarbeit der Transportunternehmen zu einer Verpflichtung. Das oben beschriebene Problem wird in der Literatur als *Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)* bezeichnet. Dieses ist durch ein Hauptdepot bzw. Hauptlager charakterisiert in welchem ich mehrere Fahrzeuge befinden. Diese Fahrzeuge holen die Waren bei ausgewählten „Abholkunden“ bzw. Abholpunkten ab und befördern diese innerhalb der eingeplanten Fahrzeit an die entsprechenden Auslieferungskunden. Um dieses Problem zu lösen, wurde die Methode *General Variable Neighbourhood Search (GVNS)* angewendet. Während das *Shaking* aus drei verschiedenen Ansätzen besteht, werden Ausgangslösungen mit einer *Greedy- und Two-Stage Cheapest Insertion Heuristic* berechnet. Beim dem lokalen Suchprozess wird eine sequenzielle und eine selbstanpassungsfähige *Variable Neighbourhood Descent (VND)* mit elf Nachbarschaften vorgeschlagen. Der Algorithmus wird auf neu erstellte Daten, die sich von 20 - 1000 Kunden erstrecken, geprüft bzw. angewendet. Die Daten wurden erzeugt, um die Leistung des Algorithmus bei verschiedenen Einnahmearten, wie zum Beispiel bei konstanten, nachfrageproportionalen und zufälligen Einnahmen zu testen und zu analysieren. Da die Eingangsdaten vielseitig sind, ist ein gut diversifizierender Algorithmus erforderlich. Deshalb wird nach mehreren Tests das stärkste *Shaking* gewählt. Das empirische Experiment zeigt zufriedenstellende Ergebnisse. Daraus kann geschlossen werden, dass der Algorithmus fähig ist, auf praxisbezogene Probleme angewendet zu werden.

Schlagwörter: General Variable Neighbourhood Search, Profitable Pickup and Delivery Problem, Self-Adaptive Variable Neighbourhood Descent, Vehicle Routing Problem, Two-Stage Cheapest Insertion Heuristic

Abstract

As the distribution market expands rapidly in a severely competitive environment, small-scale logistic companies with insufficient volume of carriers are forced to choose the customers with higher revenues to serve. Hence, the collaboration of the carriers becomes an obligation in order to avoid the travel costs. The real-life problem described above can be formulated as a *Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)* that consists of a central depot hosting the multiple vehicles which transport the goods from chosen pickup customers to the corresponding delivery customers within the pre-determined travel time limit. To tackle with this problem, a sophisticated metaheuristic approach, the *General Variable Neighbourhood Search (GVNS)* is developed. Initial solutions are constructed with a greedy heuristic and a two-stage cheapest insertion heuristic while the shaking step is formed by three different components. For the local search part, a sequential and a self-adaptive *Variable Neighbourhood Descent (VND)* with eleven neighbourhoods are proposed. The algorithm is tested on newly created data instances including 20-1000 customers. The data is generated to analyse the performance of the algorithm with different types of revenues such as fixed, demand-proportional and random. Since the input data is versatile enough to handle, a well-diversifying algorithm is needed. For this reason, the strongest shaking procedure is opted after several trials. The computational experiments show promising results. This implies that the algorithm is competent for being applied for the real-life problems.

Keywords: General Variable Neighbourhood Search, Profitable Pickup and Delivery Problem, Self-Adaptive Variable Neighbourhood Descent, Vehicle Routing Problem, Two-Stage Cheapest Insertion Heuristic

1 Introduction

The well-known combinatorial optimisation problem family *Classical Vehicle Routing Problem (VRP)* has been widely investigated due to its practical relevance to the real-world applications. After the first introduction of the problem, numerous extensions have emerged so that a notable number of exact and heuristic solution methods are developed in the competitive transportation logistics market. Several companies and organizations implemented *VRP* solution methods in practise. Applications of these optimisation methods in the distribution operations have provided them with an effective, efficient and innovative adaptation to the challenging market. Furthermore, they have gained leverage among the other firms and made a social impact by decreasing the traffic congestion and pollution.

The Vehicle Routing Problem (VRP) was first introduced by *Dantzig and Ramser* in 1959 as "Truck Dispatching". The described problem concerns with finding the optimum routing of a fleet of gasoline trucks between a terminal and several service stations. The service stations are supposed to be procured by the terminal. In this research paper, *Dantzig and Ramser* also state that *VRP* is a generalisation of the *Traveling Salesman Problem (TSP)* which aims at finding the shortest route of visiting a given number of vertices exactly once. *TSP* is the basic scheme of *VRP*. It evolves into *VRP* if the number of carriers is more than one. As a cornerstone problem, *TSP* has been studied extensively in the past and still arouses interest of the researchers.

A subclass of *Vehicle Routing Problem (VRP)* is the *General Pickup and Delivery Problem (GPDP)*. This class consists of the problems which are relevant to carrying commodities or people between particular origins and destinations. In many sources, *GPDP* is considered as a subclass of *VRP* whereas in some research articles such as in *Savelsbergh (1995)*, *VRP* is regarded as *GPDP*. For the sake of pursuing the majority, the *GPDP* is examined as a subclass of *VRP* in this research.

The purpose of this research is to scrutinise the *Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)* closely and thoroughly while developing an approximate solution method in order to solve the problem. *MVPPDP* can be acknowledged as a subclass of *GPDP* combined with *0-1 Knapsack Problem*. In *MVPPDP*, foreknown amounts of goods at the pickup customers have to be transported to the corresponding delivery customers without violating the total time and the carrier capacity limit. The carriers do not

have to visit all the customers necessarily but if they visit more customers, the chance of increasing the profit enhances.

The component of *MVPPDP* is *0-1 Knapsack Problem* which is related to the decision making process whether to choose some or all items among a given number of items with certain volumes or weights and profits. The total volume or weight of the chosen items is restricted with a capacity constraint so that in many cases all customers cannot be chosen. The aim is the maximisation of the total profits without violating the capacity constraint.

Likewise in *MVPPDP*, the objective function is to maximise the profit which is the subtraction of total costs from the total revenues collected. The word "profitable" refers to the inability of choosing all customers in many cases. Instead, the most profitable customers are opted to increase the profit as the capacity allows. This aspect of the problem fits well to the situation where a company has not have potentiality to serve all customers but some.

The other vital decision makings involved in *MVPPDP* are how to cluster customers and how to route them. Since there are multiple vehicles, the customers have to be clustered separately and they have to be put in a sequence properly within the routes. These facets of the problem can be customised to the real-life problems as well.

To inspect the topic more systematically, this master thesis is organised as follows: Chapter 2 reviews the literature for the *Vehicle Routing Problems*. Chapter 3 focuses on the *Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)* and presents the details. Chapter 4 analyses the solution method *GVNS* and explains the components of the algorithm while Chapter 5 displays the results of the computational experiment. Finally, the conclusion is stated in Chapter 6.

2 Literature Review

The basis of all logistic problems is designed on the road network structure. Prior to understand the *Vehicle Routing Problems*, the essential components of this structure should be overviewed.

All *Vehicle Routing Problems* are set on a network structure. A network is defined as a graph which has two kinds of components: arcs and vertices. Vertices (or nodes) are the road junctions and the arcs are the sections where link the vertices. The arcs can be directed or

undirected depending on the direction of the road. Directed arcs are one-way roads while undirected arcs are both-way roads. Each arc is associated with a weight which can be considered as distance, travel cost or travel time. The customers and depots are located on the road junctions (vertices) that the carriers are able to reach. The other terms used in logistics are listed below:

Customer Characteristics:

Demand: Amount of the goods customers need. The word "request" can be used as a substitution.

Time window: The time interval at which customers can be served.

Loading/Unloading times: The times required to deliver or collect the goods from the customers.

Penalty: The costs which have to be given to the customers because of inadequacy of services.

Vehicle Characteristics:

Depot: Mostly refers to the location where the service ends and vehicles are located.

Capacity: In many cases, capacity represents the restriction of volume or weight of goods which the vehicles have.

Also, the drivers of the vehicles should be taken into account. Subject to the context, there are diverse company regulations (maximum driving duration, duration of the breaks, working period of the day etc.) *(Toth and Vigo, 2002).*

2.1 Vehicle Routing Problems

The Classical Vehicle Routing Problems (VRP) are complicated and popular combinatorial optimisation problems.[1] *VRP's* have been studied broadly since the late fifties. The reason is that the *VRP's* are practically relevant to real-life problems such as bank deliveries, postal deliveries, industrial refuse collection, national franchise restaurant services, school bus routing, security patrol services and *Just In Time* manufacturing *(Bräysy and Gendreau, 2005).*

The *VRP's* description encompasses the modelling of least cost routes among the geographically dispersed customers while satisfying the customer requests which can be commodities, people etc. There are innumerable variations of the *VRP's* in the literature. Each variation of the *VRP* can be ramified into other subclasses by adding different criteria to the model. Nevertheless, the very basic *VRP* with multiple carriers can be considered as *Capacitated Vehicle Routing Problem* (*CVRP*). In this problem, a fleet of vehicles located in a central depot serve a set of customers by visiting them once and supplying their demands with specific service durations. The vehicles are capacitated and can fulfil just a single trip. At the end, they arrive at the depot. The objective is minimisation of the total distances travelled (travel costs or time).

Apart from the general definitions, inspecting the entire *VRP* family is beyond the scope of this research. However, the fundamental classes of the problem family will be probed in further chapters.

2.1.1 Classification of the Vehicle Routing Problems

Over the decades, several criteria have been added to the basic scheme of the *Vehicle Routing Problems*. For this reason, an extensive subclass family has been generated for the *VRP* in the literature. Hence, constituting a clear taxonomy of the *VRP* have become a challenging task. To classify the problems, particular characteristics should be distinguished. A very first known taxonomic outline stated by *Bodin and Golden (1981)* presents the criteria as shown in the table below.

[1] They are also known as "Truck Dispatching" or "Vehicle Scheduling".

	Type 1	Type 2	Type 3
Time to service a particular node	Time specified and fixed in advance	Time windows	Time unspecified
Number of depots	One depot	More than one depot	-
Size of vehicle fleet	One vehicle	More than one vehicle	-
Type of fleet	Homogeneous	Heterogeneous	-
Nature of Demands	Deterministic	Stochastic	-
Location of Demands	At nodes	On arcs	Mixed
Underlying Network	Undirected	Directed	Mixed
Vehicle Capacity Constraint	Imposed-all the same	Imposed-not all the same	Not imposed
Maximum Vehicle Route Times	Imposed-all the same	Imposed-not all the same	Not imposed
Costs	Variable or routing costs	Fixed operating or variable acquisition costs	-
Operations	Pickups only	Drop-offs only	Mixed
Objective	Min routing costs incurred	Min sum of fixed and variable costs	Min number of vehicles required
Other	Problem dependent constraints		

Table 1: Taxonomic criteria (Own representation based on *Bodin and Golden, 1981*)

A noteworthy detail in this classification pattern, even with many criteria presented, there still might be *Vehicle Routing Problems* which are problem dependent. These criteria shown above prove how powerful and rich the *VRP* family is. By composing a combination of the criteria above, a significant number of the *VRP* variants can be reproduced. For the reason of drawing a general taxonomic outline of the problem family, this research goes through the

main framework of *VRP*. The ensuing taxonomic review by *Toth and Vigo (2002)* links the major branches of the *VRP*.

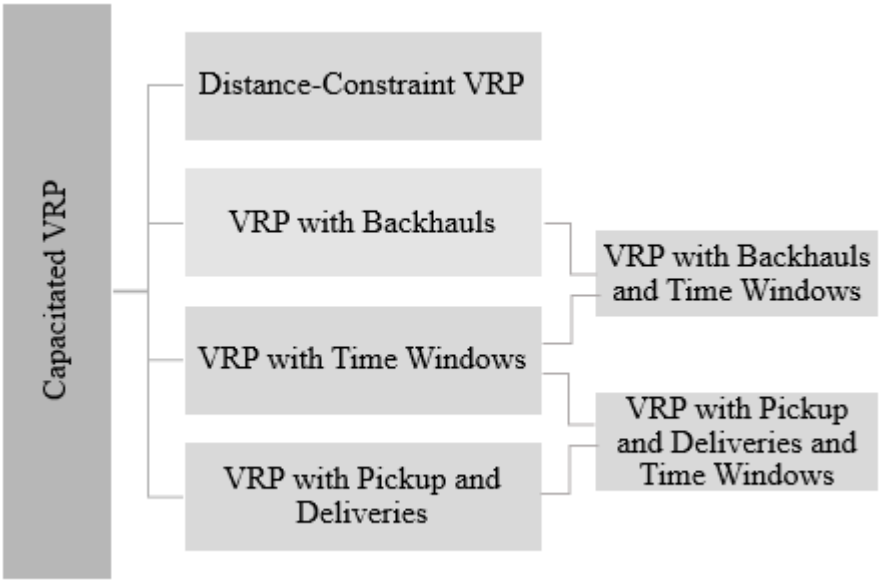


Figure 1: Basic classification of VRPs (Own representation based on *Toth and Vigo, 2002*)

The Capacitated Vehicle Routing Problem (CVRP)

The *CVRP* is the basic *VRP* subclass with the goal of minimisation of the total travelled distances. The carriers have volume or weight capacity for demand. Each customer is visited exactly once by just one vehicle and all customers are visited in order to cover the demands. Alternatively, service times can be added *(Toth and Vigo, 2002).*

The Distance-Constraint Vehicle Routing Problem (DCVRP)

The *DCVRP* is a variant of the *CVRP*. The only difference is that the *DCVRP* includes a total tour length (or time) constraint instead of a vehicle capacity limit. Service time can be added to the total time in preference *(Toth and Vigo, 2002).*

The Vehicle Routing Problems with Backhauls (VRPB)

The *VRPB* is also referred as *Linehaul-Backhaul Problem*. In the *VRPB*, customers are divided into two sets as to be linehaul customers which require a number of demand to be supplied and backhaul customers which have goods to be transported back to the depot. The objective is to minimise the total distance travelled by all vehicles. Each customer must be visited exactly once and the vehicles have a capacity constraint. Because of vehicles' having rear-load feature, there is a precedence constraint in this subclass so that the backhaul customers can be visited only if the linehaul customers are visited and the vehicles are empty *(Mingozzi et al., 1999).*

The Vehicle Routing Problems with Time Windows

The *VRPTW* is a widely studied and a very practical branch of the *Vehicle Routing Problems* which entail obligatory delivery time intervals added to the classical *VRP* features. Customers' requests have to be fulfilled in a certain time period named as "time windows". If the carriers arrive at the customers earlier than the time window beginning, they wait and the waiting time incurs. Imposing delivery deadlines makes the problem more practical *(Solomon, 1987).*

The Vehicle Routing Problems with Pickup and Deliveries

Since the problem belonging this subclass investigated in this research particularly, the *VRPPD* will be studied intensively in Chapter 2.2.

The *VRPBTW* and the *VRPPDTW* are extensions of the *VRPB* and the *VRPPD* respectively. The only additive element is the time window constraint.

2.1.2 Heuristics for the Vehicle Routing Problem

Excluding the exact solution methods, there are many heuristics developed in the literature for the *Vehicle Routing Problems*. The well-known and widely used heuristics are explained below:

2.1.2.1 The Route Construction Heuristics

The *Route Construction Heuristics* are created to find an initial solution from a sketch. In the metaheuristics concept, good initial solutions are favourable and help the algorithms to find good solutions in a shorter time than the ones which are found by the poor initial solutions. Good initial solutions do not guarantee neither optimality nor a better solution than those which are found by the poor initial solutions. But the most prominent construction heuristics can be distinguished by their unique approaches and the solution quality they yield.

The construction heuristics are generally created in parallel or sequential manner. Parallel version constructions trigger the starts of all the routes at the same time while in the sequential constructions, a route is started just after the previous route is completed.

The best known construction heuristic is probably the *Savings Algorithm.* It was first introduced by *Clarke and Wright (1964).* The algorithm starts with the construction of routes for every customer and then the routes are combined with respect to the savings values of merging two routes. The best routes with the highest savings value are merged so that in the end, a good initial solution is found.

The worst construction heuristics are the *Greedy Heuristics* such as *Nearest Neighbour Heuristic*. In this scheme, the vehicles choose always the nearest customers but often end up with the farthest customers. Thus, returning back to the depot costs too much. Nevertheless, the greedy methods are easy and quick to implement.

The *Insertion Methods* are based on the idea of inserting customers to an existing route *(Toth and Vigo, 2002).* The routes are mostly initiated by the seed customers. Afterwards, the remaining customers are inserted according to their insertion costs. This version has also both parallel and sequential types. The seed customer notion is beneficial for canalising a vehicle towards certain direction and building up the route around a particular point.

The other major constructive heuristics are the *Route-First-Cluster-Second (RFCS)* and the *Cluster-First-Route-Second (CFRS)* heuristics. The *RFCS* heuristic stems from the idea of creating a huge route and dividing it into the segments for each vehicle. The *CFRS* heuristics first focus on clustering then the routing. In the *Sweep Heuristic* which can be applied only to planar instances, customers are clustered by their coordinates for each vehicle, then the

routes are constructed with any *Traveling Salesman Problem* construction method. Another *CFRS* is known as *Fisher and Jaikumar Heuristic* which first assigns seed nodes to cluster the remaining nodes, then constructs the routes *(Toth and Vigo, 2002).*

2.1.2.2 The Route Improvement Heuristics

The *Route Improvement Heuristics* have been implemented in order to explore neighbouring solutions and this procedure is attributed as the classical local search procedure. The procedure launches exploration from the point of an initial solution which is obtained by the constructive heuristic. Then, the other solutions are scanned for a relatively better solution by performing the *Route Improvement Heuristics*. The terms "neighbourhood" and "operators" have to be defined before the upcoming details. A neighbourhood defines the neighbouring solutions of an initial solution and the operators can be described as the tools such as relocation, exchanging, inserting etc. which operate on an initial solution to explore the neighbourhood.

There exist a great number of improvement heuristics presented in the literature. *Bräysy and Gendreau (2005)* summarise the common ones. The prevailing idea of the improvement heuristics are relocating, exchanging, inserting or omitting the edges or the vertices. Hence, the improvement heuristics can be categorised according to the routes where they operate. Intra-tour heuristics allow the interactions between the customers in a route while inter-tour heuristics permit the interactions between different routes.

a) Intra-Tour Heuristics (Single-Route Heuristics):

These heuristics are performed only within a route. They explore the solution space by changing the orders of the elements of a route *(Toth and Vigo, 2002).*

2-Opt heuristic is one of the most used intra-tour improvement heuristics. In this heuristic, two edges are replaced with other edges in order to shorten the total tour distance.

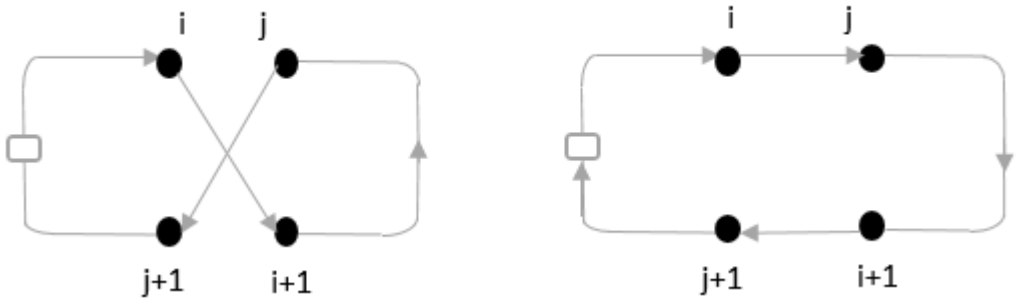


Figure 2: 2-Opt Exchange (Own illustration based on *Bräysy and Gendreau, 2005*)

2-Opt can be expanded to *k-Opt* by relocating k edges. Another improvement heuristic which is named after its inventor is *Or-Opt*. The basic idea of *Or-Opt* is to relocate a consecutive segment of vertices.

b) Inter-Tour Heuristics (Multi-Route Heuristics):

Inter-Tour Heuristics operate between different tours. The number of the tours involved can be one or more than one *(Toth and Vigo, 2002).*

An extension of the intra-tour heuristic *2-Opt* is referred as *2-Opt** which merges the tours in the same manner with *2-Opt* but operates within two routes.

One of the commonly used heuristics is the *Relocate Heuristic*. With this heuristic, three edges are replaced or in other words, simply a vertex in a route is inserted into the other route. An extended version of the *Relocate Heuristic* is the "GENI-exchange" which performs the same approach but the order of the inserted tour is changed.

Besides the *Relocate* operator, there are element-exchange heuristics. These heuristics are based on swapping elements. The well-known *Exchange Heuristic* swaps a vertex from a route with the other vertex in the other route. A multi-vertex version of the *Exchange Heuristics* is named as the "CROSS-exchange" which swaps two consecutive vertices between each route instead of swapping a vertex from each tour.

An operator referred as the "ejection chains" operates on more than two tours. It simultaneously inserts some vertices to other routes which are deleted from a route. This

procedure is applied to other routes simultaneously as well. This operator is named as "cyclic transfer operator" *(Bräysy and Gendreau, 2005).*

2.2 General Pickup and Delivery Problems

The *Vehicle Routing Problem with Pickup and Deliveries (VRPPD*) is a subgroup problem family of the *General Pickup and Delivery Problems (GPDP)* which have been reviewed comprehensively in the literature and extended broadly by the additions of several criteria. The elementary foundation of the problem is based on the transportation of certain goods between pickup and delivery customers. A set of routes are constructed in order to satisfy customer requests and the goal of the problem can vary depending on the context. To investigate better this *VRP* variant, the classification and the general framework of the problem should be inspected firstly.

Classification of the General Pickup and Delivery Problems

In the literature, the *VRPPD* has been studied under the *General Pickup and Delivery Problems (GPDP)* which is a subgroup of the *Vehicle Routing Problems*. The *GPDP* problem family is also divided into two subclasses as the *VRPB* and the *VRPPD*. There are other articles in the literature distinguishing the *VRPB* and the *VRPPD*. *Toth and Vigo (2002)* illustrate the *VRPB* on the different path other than the path of the *VRPPD*. Since there are not certain boundaries between the problem types, this research follows the majority and examines as many as possible classifications structured in the past.

A broad and detailed taxonomic review on the *General Pickup and Delivery Problems (GPDP)* by *Parragh et al. (2008)* categorises the *GPDP* into two main branches as shown in the Figure 3.

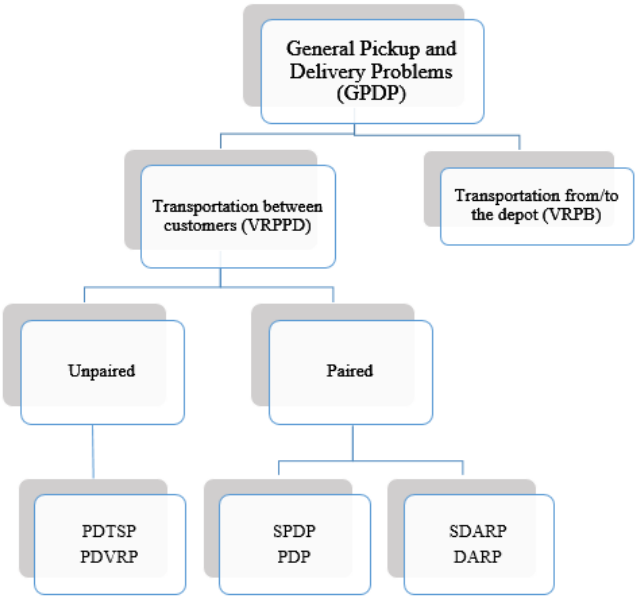


Figure 3: Subclasses of the General Pickup and Delivery Problems (Own illustration based on *Parragh et al., 2008*)[2]

The *Vehicle Routing Problem with Backhauls* has already been explained in Chapter 2.2.1. Although this subclass is a sophisticated class to be examined, the content is currently out of the scope of this research to study in depth.

One of the main branches of the *GPDP* is the *VRPPD* which consists of transporting the commodities from pickups to deliveries whereas in the *VRPB*, demands of linehaul customers are carried from the depot and demands of backhaul customers are fetched back to the depot.

Unpaired VRPPD

In this subdivision, any pickup customer can serve any delivery customer. There is no special connection between pickups and deliveries. The single vehicle case is known as the *PDTSP* (*the Pickup and Delivery Traveling Salesman Problem*) and the multi-vehicle case is the *PDVRP (the Pickup and Delivery Vehicle Routing Problem*) *(Parragh et al., 2008).*

[2] VRPB part of Figure 3 is truncated because it is out of scope.

Paired VRPPD

In contrast to its unpaired counterpart, the origins and destinations (pickups and deliveries) are associated with each other in this subclass so that the demands of delivery customers are fulfilled solely by corresponding pickup customers. This type is ramified into two subclasses as well. The distinctive criterion between subclass *Pickup and Delivery Problem (PDP)* and *Dial-A-Ride Problem (DARP)* is that while the commodities are transported in the *PDP*, the *DARP* deals with the transportation of passengers. For this reason, passenger convenience should be taken into consideration. Thus, in this branch, the problems are grouped by the criterion whether they are with one vehicle or multiple vehicles *(Parragh et al., 2008)*.

Another classification scheme can be constructed by dividing the entire problem group into two main subdivisions: static and dynamic. Static problems address the problems whose input data are known in advance while in dynamic cases, data appear and are refreshed during the period of the solution analysis. An example for dynamic case can be the dynamic *Dial-A-Ride Problem (DARP)* which copes with the transportation of elderly or handicapped people. In dynamic case, a partition of the customer requests and locations can reveal while the carriers are in action. In the literature, static cases have been studied more than dynamic cases but static problems have a limited adaptation to real-life problems *(Berbeglia et al., 2010)*. This study is a static version of the *PDP*. Thus, subsequent content addresses the static *PDP*.

According to *Berbeglia et al. (2010),* the *Static Pickup and Delivery Problems* can be classified based on the [Structure|Visits|Vehicle] idea. The structure refers to the number of the origins and destinations and is the main feature to categorise the problems. The second field defines the method of the pickup and delivery operations. If it is *PD*, the pickup and delivery operations are combined for each vertex. If it is *P-D*, these operations might be completed separately or together and if it is *P/D*, each vertex is either a pickup customer or a delivery customer. The last field indicates the number of the vehicles in the solution. On the basis of the structure attribute, the static *PDPs* can be categorised under three subclasses:

The 1-1 Problems

The *1-1 Problems* correspond to the pickup and delivery customers which are associated with each other. Commodities or passengers are delivered precisely from a specific pickup customer to the delivery customer which is considered as a pair of that pickup customer *(Berbeglia et al., 2010)*.

Courier operations and dial-a-ride system can be exemplified as real-world applications of this type *(Ting and Liao, 2013)*.

The M-M Problems

The *Many-to-Many Problem* scheme is encountered rarely in practise and is not studied thoroughly. Mostly, the single vehicle case exists in the literature. On the contrary of its 1-1 counterpart, the *Many-to-Many Problems* entail the case that any pickup customer can be used as a source to any delivery customer. In other words, the requests of the delivery customers can be fulfilled by any other pickup customer. This problem subclass corresponds the unpaired version of the previous classification *(Berbeglia et al., 2007)*.

The 1-M-1 Problems

The *1-Many-1 Problems* are different than the other two problem types. They involve in two types of commodities. The first type of the commodity's origin is the depot and it is destined to the linehaul customers whereas the second type commodity is collected from backhaul customers and is destined back to the depot *(Berbeglia et al., 2007)*.

An example of real-life application is the delivery of beverages. Soft drinks can be transported to the linehaul customers while empty containers are collected back to the depot *(Ting and Liao, 2013)*.

2.3 The Knapsack Problem

Since the late fifties, the *Knapsack Problems (KP)* have been extensively studied in the combinatorial optimization field after the initiative work of *Dantzig* in 1957. The *Knapsack Problems* are conceptually different from the *Vehicle Routing Problems* but relevant to the *Multi-Vehicle Profitable Pickup and Delivery Problem*.

The general outline of the problem can be drawn as the maximisation process of the profit by choosing some or all items among a set of given items which have weights or volumes and profits. The container where the items are kept, have a capacity constraint such that all items cannot be chosen. Some *Knapsack Problem* types are:

The 0-1 Knapsack Problem: A subset of a given number of items is chosen without the violation of the capacity constraint. Each item can be chosen at most once. The aim is to maximise the total profit.

The Bounded Knapsack Problem: This problem is similar to the *0-1 Knapsack Problem*. The only difference is that instead of having one item for each item type, there is a bounded amount for item types in this problem. For this reason, the decision maker should decide on the number of items from a certain type.

The Unbounded Knapsack Problem: On the contrary of the bounded version, in this problem there are unlimited number of items for each type.

The Multiple Knapsack Problem: This problem type includes the problems which include more than one knapsack. Their capacities can be different *(Pisinger, 1995).*

There are several variations of the *Knapsack Problem* in the literature. As in the *VRP*, many versions can be generated by adding extra criteria. Among these types of the *Knapsack Problem*, the *0-1 Knapsack Problem* is the most relevant one for the *Multi-Vehicle Profitable Pickup and Delivery Problem* since a subset of a given customers should be chosen without exceeding the capacity limits.

2.4 Vehicle Routing Problems with Profits

Another subdivision of the *Vehicle Routing Problem* is the *Vehicle Routing Problem with Profits (VRPP)* which is a synthesis of the classical *Vehicle Routing Problem* and the *Knapsack Problem*. The *VRPP* might be regarded as an enriched version of the *VRP* by adding a decision variable whether to choose to visit a particular customer or not. Instead of visiting all customers, a subset of customers is chosen with respect to certain criteria (mostly profits). The objectives and constraints may be varied as to adapt the problem to the real-life cases.

The single vehicle case of the *VRP* is referred as the *Traveling Salesman Problem*. Similarly, the single vehicle case of the *VRPP* is known as the *Traveling Salesman Problem with Profits*. According to *Feillet et al. (2005),* the *TSPP* can be categorised into three groups by the classification of their objectives and constraints.

Problem		**Objective**	**Constraint**	**#Vehicle**
Profitable Tour Problem (PTP)	Max	(Collected Profit - Traveling Cost)	-	Single
Orienteering Problem (OP)	Max	Collected Profit	On Traveling Time	Single
Team Prienteering Problem	Max	Collected Profit	On Traveling Time	Multiple
Prize-Collecting TSP	Min	Traveling Cost	On Profit	Single

Table 2: Types of the TSPP (Own representation based on *Feillet et al., 2005*)

The Profitable Tour Problem (PTP): This is the problem family aiming at the maximisation of profits collected minus total travel costs. Profits are associated with customers explicitly. The multiple vehicle case is named as the *Profitable Vehicle Routing Problem (PVRP)* and includes vehicle capacities unlike the *TSPP (Chbichib et al., 2012).* The problem which is studied in this research is a pickup and delivery version of the *PVRP*.

The Orienteering Problem (OP): The *Orienteering Problem* originates from a sport game. Each contestant begins the competition from a specific point and the goal is collecting the profits on the checkpoints as much as possible before returning back the start point. The *OP* is also recognised as the *Selective Traveling Salesperson Problem*, the *Maximum Collection Problem* or the *Bank Robber Problem*. The multi vehicle case of the *OP* is the *Team Orienteering Problem*. The objective of the *OP* is maximising the total collected profits while not exceeding the total time constraint for the vehicle *(Vansteenwegen et al., 2010).*

The Prize-Collecting TSP (PCTSP): The problem type is also referred as the *Quota TSP*. The objective function is a minimisation of the total distance travelled distinctly from the previous two problems. However, there is the constraint of the profit which has to be more than a lower bound *(Feillet et al., 2005).*

2.5 Variable Neighbourhood Search (VNS)

In computer science, the time complexity of an algorithm refers to the elementary steps performed by the algorithm to solve a specific problem. Moreover, the same size of inputs may yield different execution times (e.g. sorting algorithms). For this reason, in order to measure the complexity of a problem, a "worst case analysis" must be performed and the complexity of the problem is determined.

The problems which can be solved in non-deterministic polynomial time are known as *NP Problems*. Thus, *NP-complete* problems are hard to solve *NP Problems*. The hardest problems are *NP-hard* problems which are at least as difficult to solve as *NP-complete* problems *(Ausiello et al., 2003).*

The *NP-hard* optimisation problems are solved to optimality up to a certain number of input data. The *Vehicle Routing Problems* and the *Traveling Salesman Problems* have already been proved to be *NP-hard (Lenstra and Rinnooy Kan, 1981).* The *Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)* can be considered as a version of the *VRP* because both pickup and delivery customers can be accepted as independent customers as in the *VRP*. Thus, this reduces the problem to the *Vehicle Routing Problem*. Since the *Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)* is a variant of *MVPDP*, it can be considered *NP-hard* as well. Because the total time limit is an infinite number in *MVPDP* while it is a finite real number in *MVPPDP*.

In the literature, both exact solution methods and heuristics exist. Exact solution methods such as *Branch-and-Bound*, *Branch-and-Cut* etc. can solve the problems to optimality but the volume of the input data is limited. However, heuristics can find very good solutions but they do not guarantee the optimality. Additionally, metaheuristics are the general form of heuristics which can be adapted to any other *NP-hard* problems whereas heuristics are problem-dependent.

Many metaheuristic algorithms have been invented and developed in the literature. One of them is the *Variable Neighbourhood Search* which was first introduced by *Mladenović and Hansen* in 1997.

The *Variable Neighbourhood Search (VNS)* is based upon a search mechanism which changes the neighbourhoods systematically in the solution space to avoid the local optimum

traps and finds relatively good solutions. It consists of two parts: The *shaking* step is the stochastic part and the *local search* is the deterministic. One of two crucial aspects of exploring the solution space thoroughly is to diversify well and this is achieved by the *shaking* step. The second one is the intensification which is accomplished by the *local search*.

```
Function VNS (x, k_max, t_max)
1  Repeat
2      k ← 1;
3      Repeat
4          x' ← Shake (x, k)                    // Shaking
5          x'' ← FirstImprovement (x')          // Local Search
6          NeighbourhoodChange (x, x'', k)      // Change Neighbourhood
7      Until  k = k_max ;
8      t ← CPUtime()
9  Until  t > t_max;
```

Table 3: Algorithm for the basic VNS (Own representation based on *Hansen et al., 2008*)

In combinatorial optimisation, each problem has its own discrete solution space. To find a relatively good or optimal solution, firstly an initial solution is constructed by the construction heuristic. Then, the *VNS* starts from an initial discrete point x which is found by the constructive heuristic on the solution space. Afterwards, in the shaking step, the algorithm leaps from the current solution x to another point x' randomly (stochastic) and the neighbourhood N(x) is searched by the local search (deterministic). The local search starts the search process beginning from a point x and decreases in a course, in minimisation case, in order to find the minimum. Hence, the minimum does not have to be the global minimum necessarily. Then, the same procedure continues until the stopping criterion is met.

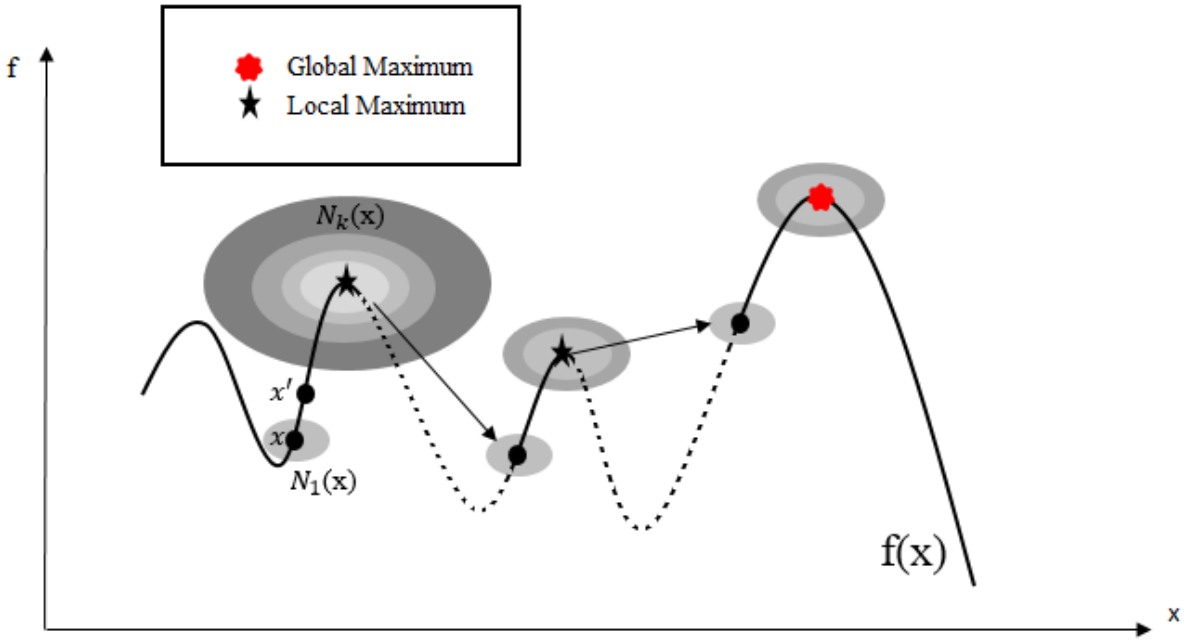


Figure 4: The illustration of the basic VNS on the solution space (Own representation inspired by *Hansen et al., 2008*)

In Figure 4, the neighbourhoods are illustrated as circles nested in each other. Moreover, this illustration is for maximisation problems. For the minimisation problems, the global minimum should be searched.

The metaheuristic *VNS* is very successful and well-known among the *VRP* solution methods. Over the decades, many researchers have studied it and numerous extensions of the *VNS* have been developed. Some of them are the *Skewed VNS*, the *General VNS*, the *Parallel VNS*, the *Primal-Dual VNS* etc.

Therefore, a great number of real-life applications exist. Some of them are:

- Industrial applications
- Design problems in communication
- Location problems
- Data mining
- Graph problems
- Knapsack and packing problems

- Mixed integer problems
- Time tabling
- Scheduling
- Vehicle routing problems
- Problems in biosciences and chemistry
- Continuous optimisation
- Discovery science

(Hansen et al., 2008)

2.6 The General Variable Neighbourhood Search (GVNS)

The *General Variable Neighbourhood Search* is a modified extension of the *Variable Neighbourhood Search*. Unlike the *VNS*, a *Variable Neighbourhood Descent (VND)* is used in the local search step of the *GVNS* algorithm.

	Function GVNS $(x, l_{max}, k_{max}, t_{max})$	
1	Repeat	
2	k ← 1;	
3	Repeat	
4	x' ← Shake (x, k);	// Shaking
5	x'' ← Seq-VND (x', l_{max}) / Mix-VND (x', l_{max})	// Local Search
6	k, x ← NeighbourhoodChange (x, x'', k);	// Change Neighbourhood
7	Until $k > k_{max}$;	
8	t ← CPUtime()	
9	Until $t > t_{max}$	
10	Return x	

Table 4: Algorithm for the *GVNS* (Own representation based on *Mladenović et al., 2012*)

The *VND* section consists of different neighbourhoods (at least two). With respect to their order several *VND* types can be created. Some of them are the *Sequential-VND*, the *Nested-VND* and the *Mixed-VND*.

Nested-VND: In the *Nested-VND*, the local search is performed with respect to the first neighbourhood for any solution in the second neighbourhood. So the solution space is explored in a nested manner *(Ilić et al., 2010)*.

Sequential-VND: The *Sequential-VND* uses the neighbourhoods one by one. The final solution has to be the local minimum of all neighbourhood structures.

Mixed-VND: This *VND* executes the *Nested-VND* up to a particular number (parameter b) of neighbourhoods and then switches to the *Sequential-VND*. The parameter b is used to increase the efficiency in terms of diversification and intensification.

If it is assumed that there are two neighbourhoods and both have cardinality *n*, *Seq-VND* visits *2n* solutions while *Nested-VND* visits n^2 solutions. For this reason, pure *Nested-VND* has larger neighbourhoods. Thus, it requires more computational time.

	Function Seq-VND (x, l_{max})	
1	l ← 1;	// Neighbourhood counter
2	Repeat	
3	i ← 0;	// Neighbour counter
4	Repeat	
5	i ← i+1;	
6	x' ← arg min {f(x), f(x_i)},$x_i \in N_l$(x)	// Compare
7	Until $(f(x') < f(x)$ or $i = \|N_l(x)\|)$	
8	l, x ← NeighbourhoodChange (x, x', l);	// Neighbourhood Change
9	Until $l > l_{max}$;	
10	Return x	

Table 5: Algorithm of the Sequential-VND (Own representation based *on Mladenović et al., 2012*)

The neighbourhood change part is the accelerator of the *Seq-VND* algorithms. After a certain neighbourhood is scanned and if there is no improvement comparing to the previous solution, the next neighbourhood is searched. If there is an improvement, then the same neighbourhood is explored. The process continues until all neighbourhood structures are scanned for a minimum solution.

	Procedure NeighbourhoodChange (x, x'', k)	
1	If f(x'') < f(x) then	
2	x ← x'' ; k ← 1;	// Make a move
3	Else	
4	k ← k + 1;	// Next neighbourhood

Table 6: The Neighbourhood Change algorithm (Own representation based on *Mladenović et al., 2012*)

Apart from the algorithms, the essential parts of the *General Variable Neighbourhood Search* with the *Sequential-VND* can be paraphrased as:

- Construct an initial solution *x*
- Randomly perturb the solution *x* to *x'*
- Explore the neighbourhood *x'* with the first neighbourhood N_1 and find the local minimum of N_1 which is *x''* (best improvement)
- If *x''* is smaller than *x*, then take that solution as a current point on the solution space and continue the search process with the first neighbourhood N_1
- If *x''* is not smaller than x, then the next neighbourhood N_2 should be explored.
- The neighbourhood search process (*VND*) stops if all neighbourhoods are explored.
- The entire algorithm terminates if the stopping criterion is met.

Stopping criteria can be: a certain time period, a certain number of iterations, a certain number of iterations without improvement etc.

3 The Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)

The *Classical Vehicle Routing Problems (VRP)* is derived from the idea of the companies' being capable of serving all emerging customers. In contrast to this generalisation, the companies in the transportation logistics market have limited sources, yet cannot fulfil the requests of all customers. The growing and severely competitive transportation market can be shown as a reason. If a company has few vehicles and a large number of customers, it is driven to optimise the decision and grab as many as customers in order to survive in the fierce market.

At this point, the *Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)* arises and this research presents the steps for developing an efficient algorithm to find approximately good solutions for this problem.

3.1 Problem Definition

The *MVPPDP* outlines the static problem with a central depot and pre-determined number of vehicles transporting the demands of pickup customers to the delivery customers. The pickup and delivery customers are paired so that a specific delivery customer can be only

served by the corresponding pickup customer. All customers might not be visited. Hence, the visited customers cannot be visited again.

The customer pairs have revenues which can be collected at the delivery customers and the distances between the customers are associated with a cost value. The pickup customers have demand surpluses destined to the delivery customers. Thus, the vehicles are initially located at the depot and expected to return back to the depot after the service. The delivery customers have to be visited after the visit of the paired pickup customer. As long as the constraints are not violated, many consecutive pickup customers or many consecutive delivery customers might be visited. These constraints refer to the vehicle capacity, total time limit and precedence constraints.

Moreover, the constraints imply that the first visited customer has to be a pickup customer and the last visited customer has to be a delivery customer. The objective of the problem is to maximise the total collected revenues minus the total travel time. Nonetheless, in real life, serving as many customers as possible might increase the number of the customers which can be loyal customers of the company in future. This might be a secondary and hidden objective of the problem.

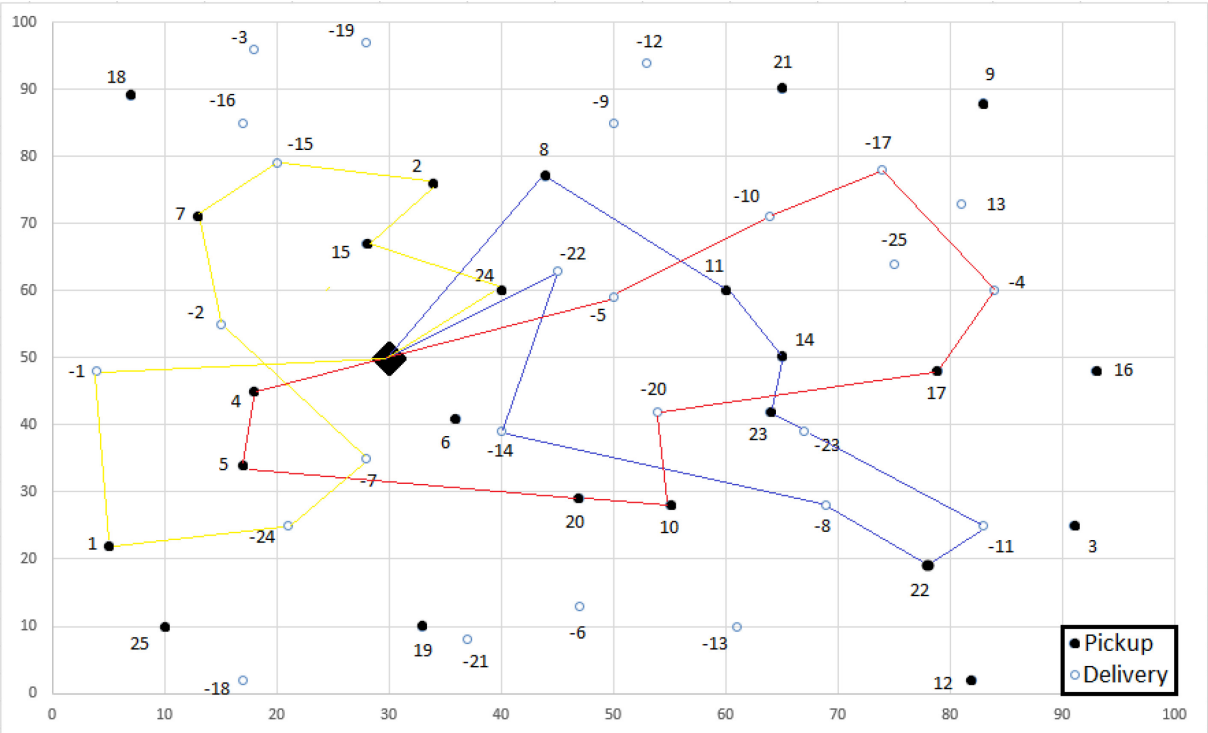


Figure 5: A best found solution for 50 customer-data with 3 vehicles and fixed revenues. The vehicles collect only the nearby customers in fixed revenue case. Pairs are denoted as (i,-i) ∈ Z

3.2 Assumptions

In order to define the boundaries of the problem, some assumptions should be stated in advance:

- The transported goods are identical and homogenous. Even though the goods are identical, demands loaded from pickup customers have to be transported to their pairs.
- The vehicles are identical, homogenous and have the same capacity.
- The vehicles are located at the central depot in an empty position and return to the depot at the end of their service.
- The vehicles have no rear-loading problem so that first loaded demand can be unloaded first.
- There is no service time and inconvenience constraint as it is in the *Dial-A-Ride Problems (DARP)*. Moreover, there is no time window for the deliveries.
- Revenues are assumed to be collected at the delivery customers.
- Each vehicle travels a certain distance at the same duration and cost.

3.3 Classification of the MVPPDP

The *Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)* belongs to the *Vehicle Routing Problem* family and the characteristics of the *MVPPDP* are inherited from two subclasses of the *VRP*. The feature of its being a pickup and delivery problem assures that the *MVPPDP* is related to the *Vehicle Routing Problem with Pickup and Deliveries*. Moreover, choosing a subset of customers according to their revenues indicates that the problem is a *Vehicle Routing Problem with Profits*. All in all, the ultimate classification of the *MVPPDP* is the intersection of the *VRPPD* and the *VRPP* and can be categorised as a *1-1 Problem* with respect to the taxonomy of *Berbeglia et al. (2007).*

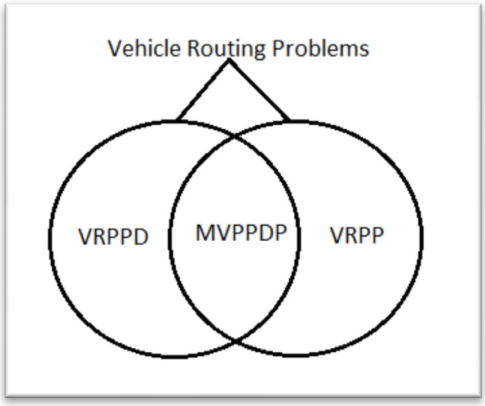


Figure 6: Classification of the MVPPDP

3.4 Mathematical Formulation

The following mathematical formulation of the *MVPPDP* is the modified version of the *PDPTW* of *Ropke and Cordeau (2009),* the *MVPDP* of *Parragh et al. (2008)* and the *SPDP* of *Ting and Liao (2013).*

There are n requests (customer pairs) on a given complete and directed graph $G = (V, A)$. V indicates the set of all vertices and $A_{i,j}$ is the arc between the vertices $i, j \in V$. Customers are divided into two groups, the pickup and delivery customers. All vehicles start from 0 (warehouse/depot) and return back to the same point 2n+1. The demands of the customers are designated as the following: for a specific pickup customer it is a positive value and the corresponding delivery customer which is its pair has a negative value. The reason is that the demand is transported from the pickup customer to the delivery customer. In other words, the demand of pickup customers can be considered as supply. The distances between the vertices $i = (i_x, i_y)$ and $j = (j_x, j_y)$ are calculated according to the *Euclidean Distance*. The decision variable $x_{i,j}^k$ is a binary variable and 1 if an arc between vertices *i* and *j* is visited by the vehicle $k \in K$, 0 otherwise.

All other notations are listed below:

n	number of requests (number of customer pairs)
$P = (1, 2, \dots, n)$	pickup customers
$D = (n+1, n+2, \dots, 2n)$	delivery customers
$W = (0, 2n+1)$	warehouse / depot (departure and arrival)
$V = P \cup D \cup W$	all vertices (combination of pickups, deliveries and the depot)
$K = (1, \dots, m)$	set of vehicles
R_i	revenues which are associated with each request (demand) $i \in P$
q_i	demands of vertex $i \in V \backslash W$. (for pickup $q_i > 0$, for delivery $q_i < 0$)
$c_{i,j}$	cost to traverse an arc $A_{i,j}$ by each vehicle
$t_{i,j}$	travel time from vertex i to vertex j by each vehicle
T	total route time limit for each vehicle
B_i^k	actual time when the vehicle $k \in K$ arrives at the vertex $i \in V \backslash W$
C	capacity of each vehicle $k \in K$
Q_i^k	load of the vehicle $k \in K$ when it has visited vertex $i \in V \backslash W$

The mathematical model of the *MVPPDP* can be presented as:

The objective of the problem is to maximise the profit. This can be calculated by subtracting the total travel costs from the total revenues collected. The first part of the function represents the total revenues and the second part expresses the total travel costs.

$$\max \left(\sum_{k\in K}\sum_{i\in P}\sum_{j\in V} R_i x_{i,j}^k - \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{i,j}\, x_{i,j}^k\right) \qquad (1)$$

The first constraint is special to the *MVPPDP*. Because in this problem, all customers (excluding the depot) may not be visited. Thus, they have to be visited at most once.

$$\sum_{k\in K}\sum_{j\in V} x_{i,j}^k \leq 1 \qquad \forall i \in V\backslash W \qquad (2)$$

The next constraint assures that each vehicle starts from the depot and returns back to the depot. The first part is for the departures and the second part is for the arrivals. This does not mean that all vehicles have to start their trip. The arc between 0 and 2n+1 can be visited as well. In this case, the vehicle does not start the trip *(Parragh et al., 2008).*

$$\sum_{i\in V} x_{0,i}^k = \sum_{i\in V} x_{i,2n+1}^k = 1 \qquad \forall k \in K \qquad (3)$$

For the intermediate vertices, if the vehicle arrives at a vertex, it has to leave it. This constraint is mostly referred as the flow constraint. The warehouse is excluded.

$$\sum_{i\in V} x_{i,j}^k - \sum_{i\in V} x_{j,i}^k = 0 \qquad \forall j \in K, \quad \forall j \in V\backslash W \qquad (4)$$

The pickup and delivery pairs have to be served by the same vehicle.

$$\sum_{j\in V} x_{i,j}^{k} - \sum_{j\in V} x_{i+n,j}^{k} = 0 \qquad \forall i \in P, \quad \forall k \in K \qquad (5)$$

The travel time between vertices *i* and *j* have to be added to the departure time from vertex *i*. Therefore, arrival to vertex *j* cannot be earlier than the departure time from vertex *i* if the arc between them is visited. If there was service time for vertex *i*, it has to be added to the equation.

$$B_j^k \geq \left(B_i^k + t_{i,j}\right)x_{i,j}^k \qquad \forall i,j \in V, \quad \forall k \in K \qquad (6)$$

The pickup customers have to be visited before the corresponding delivery customer. This constraint is mostly named as the "*Precedence Constraint*".

$$B_i^k \leq B_{i+n}^k \qquad \forall i \in P, \quad \forall k \in K \qquad (7)$$

Each vehicle has to complete the tour within the total travel time limit. The total time limit is the same for all vehicles. The following constraint ensures the time limit condition.

$$B_{2n+1}^k - B_0^k \leq T \qquad \forall k \in K \qquad (8)$$

Additionally, the constraints (9) and (10) guarantee that vehicle load is maintained during the trip.

$$Q_j^k = \sum_{i \in V} \left(Q_i^k + q_j\right) x_{i,j}^k \qquad \forall j \in V, \quad \forall k \in K \qquad (9)$$

The vehicle load at node i has to be smaller than the vehicle capacity. This constraint prevents the vehicle capacity violations throughout the process.

$$Q_i^k \leq \mathrm{C} \qquad \forall i \in V, \quad \forall k \in K \qquad (10)$$

The decision variables have to be binary. Either an arc is chosen or not.

$$x_{i,j}^k = \begin{cases} 1 & if\ the\ arc\ (i,j)\ is\ visited \\ 0 & otherwise \end{cases} \qquad \forall i,j \in V,\ \forall k \in K \qquad (11)$$

This model is a non-linear mixed integer program because the constraints (6) and (9) are not linear *(Ropke and Cordeau, 2009).*

4 The Solution Method

This research proposes an algorithm in the framework of the *General Variable Neighbourhood Search (GVNS)* in order to find an approximate solution for the *Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)*. The *MVPPDP* is an *NP-hard* combinatorial optimisation problem and can be solved to optimality up to a small number of input data. Hence, it is possible to obtain relatively good solutions with the *GVNS* which is based on the idea of changing of neighbourhoods systematically to explore the

neighbourhoods of the initial solution which is found by the constructive heuristics. The *GVNS* is an extension of the basic *VNS* presented in Chapter 2.5 with a sophisticated local search component. Moreover, it should be stressed that the neighbouring solutions of an initial solution can be explored in depth with a well-structured local search component.

The *GVNS* basically consists of three main parts affecting the solution quality. These are construction, shaking step and the *Variable Neighbourhood Descend (VND)*. For the construction part, a greedy heuristic and a two-stage cheapest insertion heuristic are developed. In the shaking step, there exist three parts which are randomly chosen in each iteration. The reason of its consisting of three parts is for diversification purposes over the solution space. *VND* includes eleven neighbourhoods in order to explore better. Therefore, the order of the neighbourhoods is very significant. For this reason, a sequential and a self-adaptive *VND* is presented in this research. The local search section is vital for the intensification on the exploration areas. The main parts are investigated elaborately under the following titles.

4.1 Data Structures

Before starting the actual algorithm, the data structures should be overviewed because during the development process, it is experienced that data structures play a crucial role on the *CPU* times. For the pickup and delivery problems, it is generally hard to find the exact spot of the delivery customer in the solution container.

Throughout the algorithm, the solution is stored in a vector of vectors (matrix alike) and the other input data such as revenues, demands and coordinates of the customers are contained in vector containers.

Since there are unvisited customers, they are contained in another vector container and updated several times during the process which is time-consuming. Instead of separating the customers and storing them in different containers, the tour info map is initiated. The map is an associated container which links the keys and the values. In this case, the customers are key values while the actual values are the route info values.

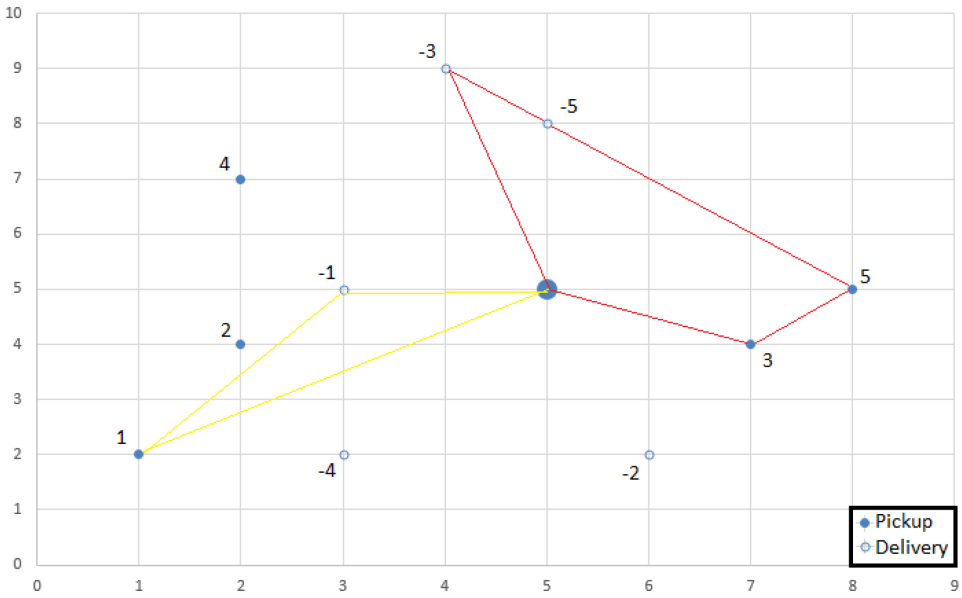


Figure 7: A small example which has *n=5* requests and *2* vehicles

The route info can be stored in a map such that upper part represents the *customer*, the lower left part is the *route number* while the lower right part is the *order number*. If a customer pair is not visited then, the route info is updated as 0.

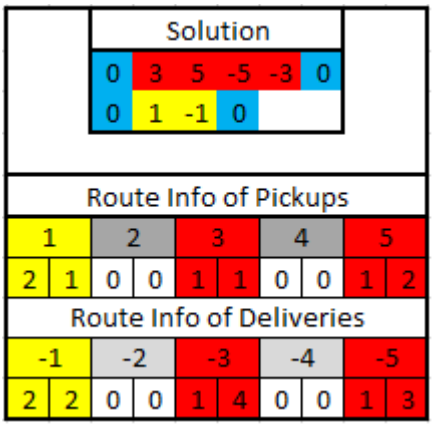


Figure 8: An illustration of the data structures of Figure 7

4.2 Construction Heuristics

The constructive heuristics can shorten the computational time by finding a good starting point for the search process. Nonetheless, these starting points do not guarantee that the final solution will be the optimal. Thus, a bad starting point might affect the algorithm so that better solutions can be finally found. In order to inspect the effects, both a bad and a good

construction heuristic are developed. The *Greedy Construction Heuristic* which is developed is quick, easy to implement but poor while the *Two-Stage Cheapest Insertion Heuristic* is sophisticated and time consuming but yields better results. The *Greedy Construction Heuristic* is denoted as *C1*. *C2* represents the *Two-Stage Cheapest Insertion Heuristic*.

4.2.1 The Greedy Construction Heuristic (C1)

The *Greedy Construction Heuristic* is built similar to the *Nearest Neighbour Heuristic*. In the classical *Nearest Neighbour Heuristic* for the *VRP*, starting from the depot vertex, always the nearest vertex is chosen sequentially until the total time limit is completed. Since the *MVPPDP* is a pickup and delivery problem and there is no obligation to visit all the customers, *C1* selects the customer pair based on a specific ratio. The distances and revenues affect the objective function so both characteristics have to be taken into account in the calculation of the ratio. After the selection of the customer pairs, the routes are constructed. Firstly the pickup customer, afterwards the delivery customer is inserted in order to avoid the feasibility checks of the route for precedence and demand capacity constraints.

If the current vertex is $i \in V$, the ratio below is calculated $\forall j, -j \in V$. But the vertices j and -j have to be unvisited customers. After the calculation of the ratios for all customers, the customer pair with the highest ratio is selected and inserted to the solution respectively.

$$\text{The Selection } Ratio = \frac{R_{j,-j}}{d_{i,j}+d_{j,-j}+d_{-j,0}}$$

An example of 2 iterations of the algorithm is illustrated below. In the first iteration, the current vertex is 0 (depot) and the chosen customer pair is (5, -5). For the next iteration, the current vertex is -5 and the chosen customer pair is (4,-4). This procedure continues until the total time limit is reached. Then, a new route is initiated. Since the revenues are fixed in this example, the algorithm chooses the nearest vertices.

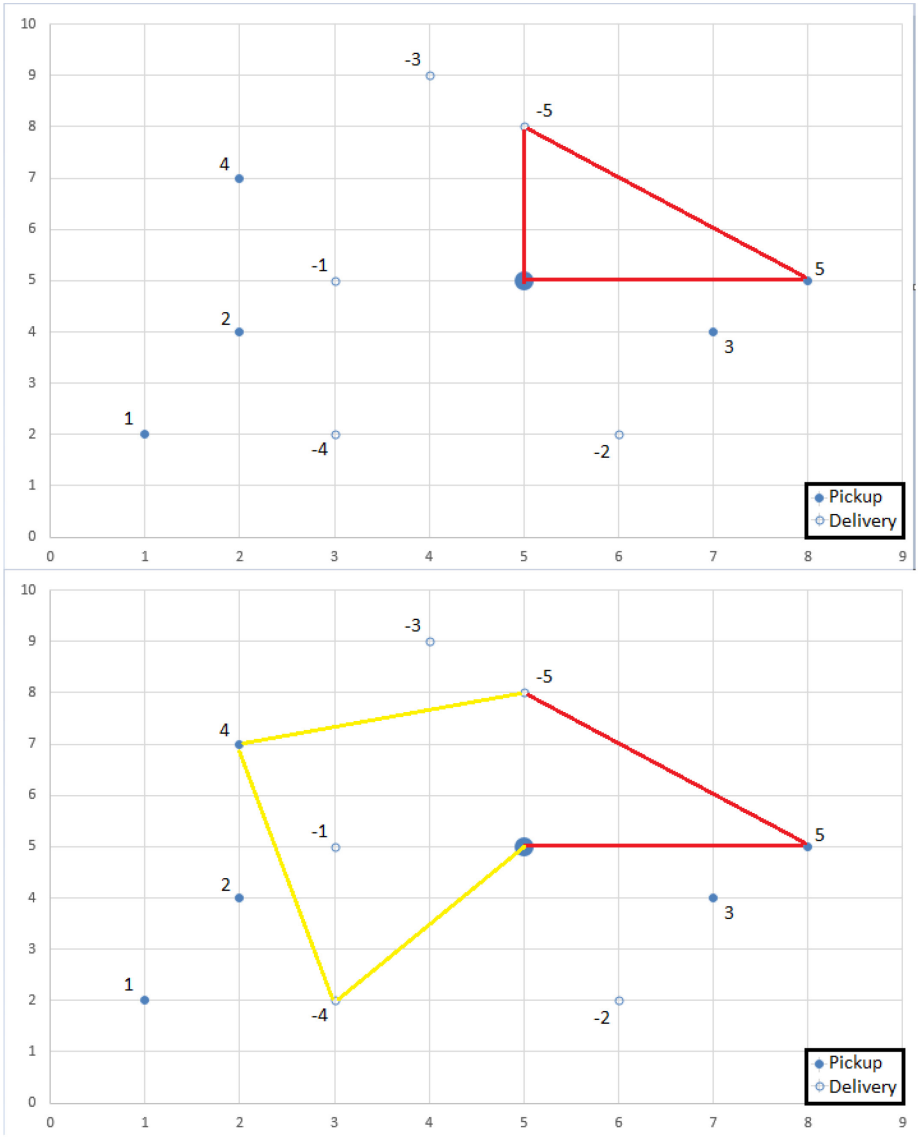


Figure 9: An example for the *Greedy Construction Heuristic*

4.2.2 The Two-Stage Cheapest Insertion Heuristic (C2)

The insertion heuristics are built on the concept of inserting vertices into an existing route by considering a particular criterion which is mostly cost-related. For the *Vehicle Routing Problems*, inserting a vertex to a route bears extra costs. Hence, the cheapest vertices are preferred to build the tour. For the *Pickup and Delivery Problem*, the costs for inserting two customers have to be considered because the customers are paired and have to be located in the same route.

In order to insert vertices, some initial routes must be constructed in advance. The most common idea for constructing initial routes depends on the seed node idea. Before the construction of the routes, the seed vertices are determined by any criterion (nearest, farthest, related, etc.). Then, the seed vertices are assigned and the insertion process starts.

This research proposes a two-stage cheapest insertion heuristic. Initially chosen customer pairs are assigned for each route. Then, the first stage of the algorithm commences. The first stage is a procedure in which all customer pairs are tried to be inserted. Some of them might not be inserted because of the demand capacity constraint and added to the second stage customer list. Meanwhile, those routes are forbidden for the un-inserted customers. In the second stage, all the customer pairs which are not inserted in the first stage are evaluated again for the new routes. Afterwards, the same procedure continues until either there is no insertion slot for the un-inserted customers or all the routes are completed.

a) The Seed Vertex Selection

The seed vertices can be selected with respect to many criteria. In this algorithm, the seed nodes are selected by the idle distances. Between the depot to the first customer (pickup) and the last customer (delivery) to the depot, the vehicles are idle. Selecting the customers with the shortest total idle distance allows inserting other profitable customers between the seed customer pair. For each route, a one pair of seed is assigned and the very basic routes are constructed.

P	set of the pickup customers
D	set of the delivery customers
$W = (0, 2n + 1)$	warehouse / depot (departure and arrival)
$V = P \cup D \cup W$	all vertices (combination of pickups, deliveries and the depot)

$$Total\ idle\ distance\ of\ a\ pair\ (i,d) = d(0,i) + d(0,-i) \qquad \text{for } \forall i \in P,\ -i \in D$$

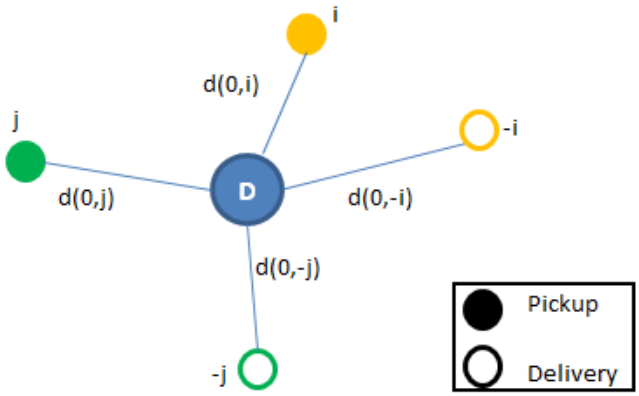


Figure 10: The seed vertex selection with two customer pairs

b) The Insertion Criterion

After the routes are constructed with the seed customers, the customers are chosen for expanding the routes. Since the revenues are very important for the construction of a profitable route, they have to be included in the selection criterion. For this reason, for each customer pair, a ratio is calculated by dividing the revenues of candidate customers to the insertion costs.

$$\forall a, -a, b, -b, c, -c \in V \backslash W, \qquad \forall i \in P, -i \in D$$

$$Insertion\ Ratio = \frac{R_{i,-i}}{(d_{-a,i} + d_{i,b} - d_{-a,b}) + (d_{-b,-i} + d_{-i,c} - d_{-b,c})}$$

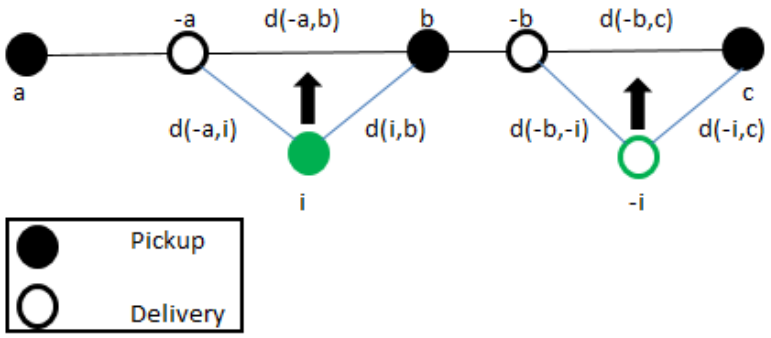


Figure 11: The insertion criterion

Note that the elements of the existing route can be any vertex, pickup or delivery.

For each specific customer pair, all possible slots in the existing route are checked and the proper slots are chosen. Then, the next function calculates the ratios for each pair and the pair with highest ratio is inserted to the tour.

An example for the trials of a pair's insertion to all possible slots in a route is presented below. The customer pair (b,-b) is inserted in a route in 3 iterations. The route has already a pair (a,-a). The precedence constraint is also not violated.

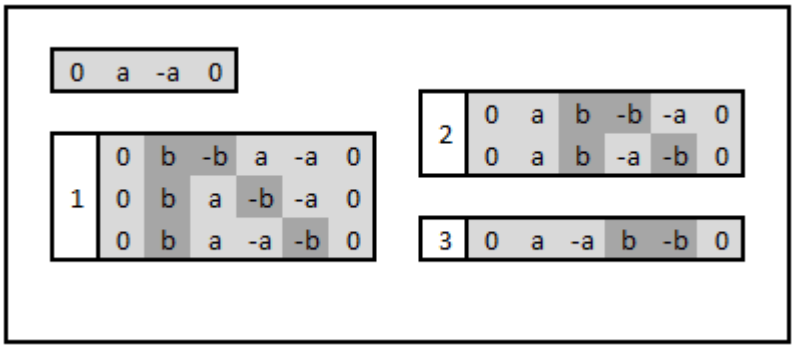


Table 7: The possible slots for the insertion of a customer pair

4.3 Shaking

The *General Variable Neighbourhood Search (GVNS)* investigates the solution space by changing neighbourhoods in a deterministic manner. Therefore, a stochastic shaking process is necessary and very important in order to diversify widely on the space because it enhances the possibility of finding better solutions.

A usual shaking step includes one or more random numbers and one or more operators to cover the space. The *GVNS* algorithm in this research has a shaking step which consists of 3 elements and it perturbs the initial solution x. Each element is a sub-shaking move (shake1, shake2 and shake3) and each has roughly 33% possibility to be implemented.

	Shake
1	Generate a random number between [1,3]
2	If the random number is 1
3	| Apply shake1
4	End if
5	If the random number is 2
6	| Apply shake2
7	End if
8	If the random number is 3
9	| Apply shake3
10	End if

Table 8: The pseudo-code of the shaking step

4.3.1 Shake1

Shake1 is an element of the main shaking step and comprises of a removal and reconstruction part. The removal process is stochastic while the latter is deterministic.

	Shake1
1	Generate a random number (R1) to choose the route
2	If the route has at least one pair of customers
3	| Generate a random number (R2) to choose the customer
4	| Remove a pair of customer randomly
5	| Reconstruct with cheapest insertion heuristic
6	End if

Table 9: The pseudo-code of Shake1

Shake1 generates two random numbers (R1, R2) for determining the route and the order of the customer respectively. Afterwards, either the random customer is a pickup customer or a delivery customer, that customer is removed with its pair. Only 1 pair of customer is erased for all routes in the solution and becomes unvisited (dummy customer).

An example of removal process with 10 pickup customers *(P1… P10)* and 10 delivery customers *(D1… D10)* is illustrated below. In the example, the red customer pair is removed with respect to R1 and R2. Even if R2 were 6, the same customer pair would be erased.

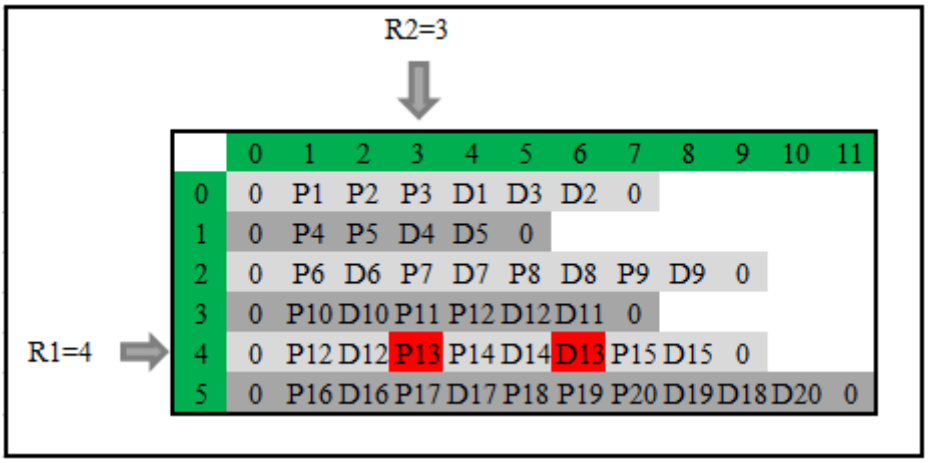


Table 10: The removal of a customer pair in *Shake1* and *Shake2*

The route whose customer pair is removed, is filled with an unvisited customer pair which is inserted by the cheapest insertion operator excluding the removed customer pair. If there is no customer inserted due to the capacity or precedence constraints, the route is not reconstructed and left one pair missing.

The reason for excluding the removed pair is that small data samples are not perturbed well enough because the same customers are inserted almost always.

4.3.2 Shake2

The second component of the shaking step is *Shake2* which has a removal operator and cheapest insertion reconstruct operator as in *Shake1* but reconstruction part does include the previously removed customer pair. The reason is that if all sub-shakes exclude the removed customer pair, the process becomes problematic in the case of visiting all customers.

Shake2 includes an additional random number (R3) besides (R1, R2). R3 is generated randomly between [10, 40] for determining a percentage of the customers in a chosen route by R1. If a randomly chosen route has x pairs of customers, then at least $\frac{x}{10}$ or at most $\frac{x}{40}$ pairs are removed. Furthermore, if $\frac{x}{10}$ is smaller than 1, there is a condition which obliges the removal of at least 1 pair of customers in the route in order to assure the shaking step.

It is experienced that the percentage limit should not be more than approximately 40%. If a bigger segment is destroyed in the route, the reconstruction consumes too much time and the *CPU* time increases significantly.

```
Shake2
Generate a random number (R1) to choose the route
Generate a random number (R3) to choose the percentage
If the route has at least one pair of customers
    If the remove percentage is below 1
        Assign the customer number to 1
    End if
    For customer pairs up to the chosen percentage
        Generate a random number (R2) to choose the customer
        Remove a pair of customer randomly
    End for
End if
Reconstruct with cheapest insertion heuristic
```

Table 11: The pseudo-code of the *Shake2*

4.3.3 Shake3

Shake3 is an extension of *Shake2* with a distinctive difference. While *Shake2* removes customers randomly, *Shake3* removes the worst customer pair in a route. By setting a loop, each time the less contributing customer pair is eliminated and marked as unvisited. The other components of *Shake2* and *Shake3* remain the same.

```
Shake3
Generate a random number (R1) to choose the route
Generate a random number (R2) to choose the percentage
If the route has at least one pair of customers
    If the remove percentage is below 1
        Assign the customer number to 1
    End if
    For customer pairs up to the chosen percentage
        Remove the worst customer pair
    End for
End if
Reconstruct with cheapest insertion heuristic
```

Table 12: The pseudo-code of the *Shake3*

The term “worst customer pair” refers to the customer pair determined by the removal ratio. This ratio is the same with the insertion ratio as in Chapter 4.2.2 but unlike it, the ratios are calculated for all customer pairs in an existing route and the customer pair which has the smallest removal ratio is erased from the route.

$$Removal\ Ratio = \frac{R_{i,-i}}{(d_{-a,i} + d_{i,b} - d_{-a,b}) + (d_{-b,-i} + d_{-i,c} - d_{-b,c})}$$

Implementing solely *Shake3* in the shaking step instead of applying the combination of 3 sub-shakes (*Shake1*, *Shake2 and Shake3*) with a percentage yields worse result because conceptually good metaheuristics allow some deteriorations in order to escape local minima or maxima traps.

4.4 Neighbourhood Structures

The *Variable Neighbourhood Search* is a metaheuristic which seeks for better solutions by changing the neighbourhoods in the search *(Mladenović and Hansen, 1997)*. Since the *General VNS* is an extension of the *VNS*, the neighbourhood structures are the crucial parts of the local search as well. Each neighbourhood explores a particular set of the solutions in the solution space and this neighbourhood can be nested in another neighbourhood, e.g. inserting a node from a route to another can be a neighbourhood within swapping two nodes between two routes because inserting twice equals swapping move. If the neighbourhoods are not nested, they might be considered as disjoint neighbourhoods.

This research proposes 11 different neighbourhoods, 6 of them are constructed by *intra-tour* improvement heuristics and 5 of them are formed by *inter-tour* improvement heuristics. All given examples in this chapter include the infeasible routes which are sorted in the algorithm with the constraint checking functions.

4.4.1 Intra-Tour Improvement Heuristics

The *intra-tour improvement heuristics* concentrate on the routes individually. Since the routes' customers are already determined, each route can be treated as a *Traveling Salesman Problem* so *intra-tour heuristic*s minimise only the total distance of the routes. Therefore, the elements of the routes are relocated or swapped with the elements of the same route to obtain better solutions.

In this research, the *intra-tour improvement heuristics* follow the *first improvement method* which always takes the firstly improved solution as a current solution and starts the exploration procedure from that current solution. Instead of the *first improvement method*, the *best improvement method* could be implemented. In the *best improvement method*, all neighbouring solutions are evaluated and the best one is chosen as the current solution. Afterwards, the procedure continues from the current solution. This method is named "steepest descent".[3]

4.4.1.1 The Pairwise Forward Exchange (PFE) and the Pairwise Backward Exchange (PBE)

The customers are swapped with other customers following a pattern for each route. Thus, for each time they are swapped, if the new route is not feasible in terms of capacity and precedence constraints, they are swapped back. The *PFE* starts swapping from the beginning to the end of the route while the *PBE* applies it from the end.

An example with 2 pairs of customers ((a,-a) (b,-b)) in 3 iterations is presented below. There exists just one route. The *PFE* and the *PBE* operate within the route by swapping elements.

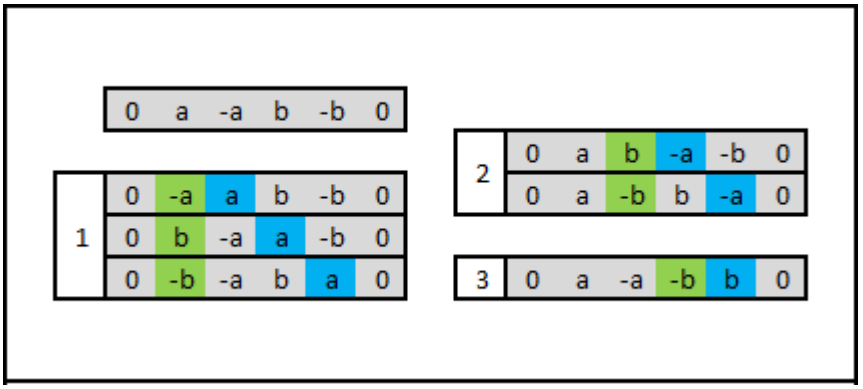


Table 13: An example for the *Pairwise Forward Exchange (PFE)*

[3] Steepest ascent in the case of maximisation.

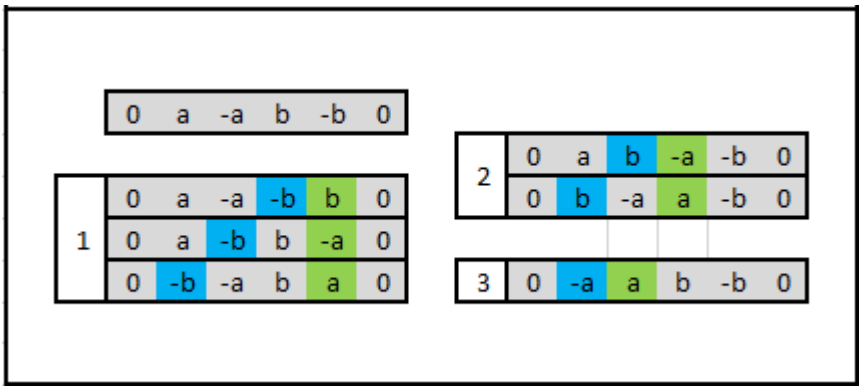


Table 14: An example for the *Pairwise Backward Exchange (PBE).* The blue nodes/customers are swapped with the green ones

4.4.1.2 The Relocate Pairs (RP)

The *Relocate Pairs heuristic* is based on the idea of relocating a consecutive pair within a route. Swapping consecutively ordered customer pairs (a pickup followed by a delivery) instead of regular customer segments (a mixed order) reduces the *CPU* time. The reason is that for each iteration, the precedence feasibility is not checked. On the contrary, the demand feasibility should be assured because there is no guarantee for the route's being feasible after relocating the pickup customer to its new slot.

The *RP* is explained in an example with 3 customer pairs ((a,-a), (b,-b), (c,-c)). The order of the customers may differ but the *RP* can be applied if and only if there is a consecutive pair or more pairs.

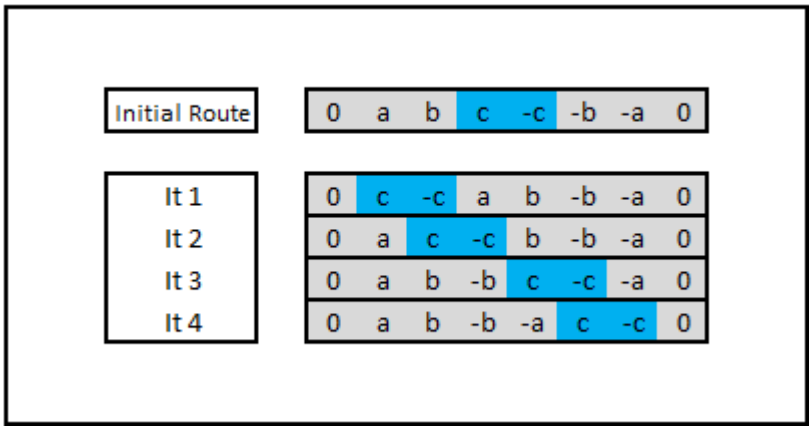


Table 15: An example for the *Relocate Pairs (RP)*

4.4.1.3 The 2-Opt

The *2-Opt heuristic* is a very well-known improvement heuristic for the *Vehicle Routing Problems*. It is a special case of the *k-Opt* which aims at finding improvement for the given solution by erasing k edges and reconnecting them. Therefore, the *2-Opt* focuses only two edges and it is observed that it yields many infeasible neighbouring solutions for the *Pickup and Delivery Problems*. The reason is that by applying the *2-Opt*, the elements in a route are flipped and the precedence constraint might be violated. Nonetheless, it is very crucial to implement it for catching the feasible and good solutions.

An example of the *2-Opt heuristic* with two pairs of customers ((a,-a), (b,-b)) in 3 iterations is shown below. The *2-Opt* flips some segments in the route depending on the indices.

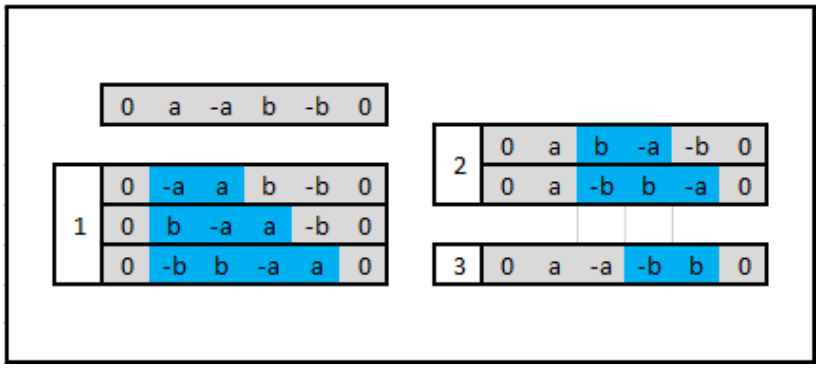


Table 16: An example for the *2-Opt*

4.4.1.4 The Forward Insertion (FI) and the Backward Insertion (BI)

In addition to the intra-route exchange operators, the *Forward Insertion (FI)* removes a customer which can be a pickup or a delivery customer and inserts it forwardly. On the other hand, the *Backward Insertion (BI)* applies the same pattern but the customers taken from the end of the route are inserted from the end to the beginning of the route. These improvement heuristics strengthen the local search by operating in each route and help the algorithm to find round-shape routes which are more likely good solutions.

The examples below analyse both the *FI* and the *BI*. There are two pairs of customers ((a,-a), (b,-b)) in the example.

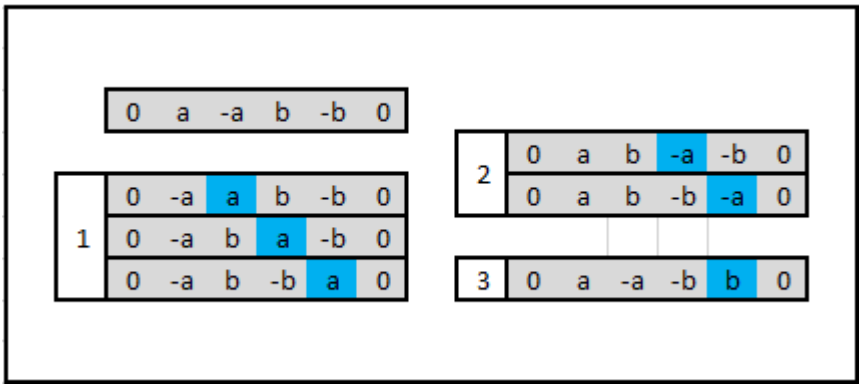


Table 17: An example for the *Forward Insertion (FI)*

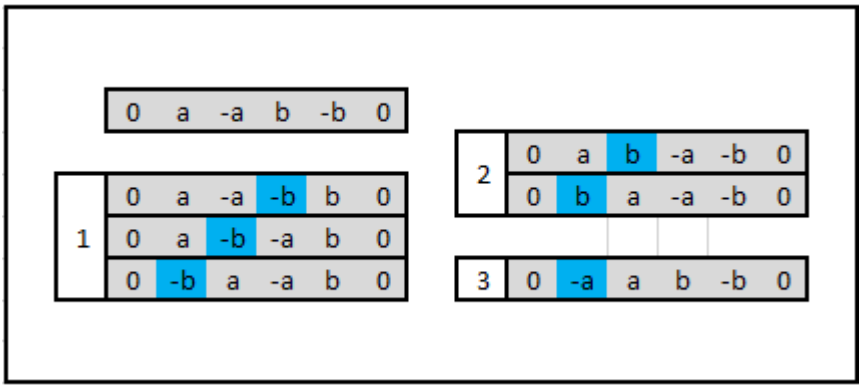


Table 18: An example for the *Backward Insertion (BI)*

4.4.2 Inter-Tour Improvement Heuristics

On the contrary of the *Classical Vehicle Routing Problems*, the *Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)* is a problem type which all customers do not have to be visited necessarily but if more customers are visited then the possibility of increasing the profit may rise. Therefore, the customer set can be divided into two subsets (i) visited and (ii) unvisited customers which form an imaginary "dummy tour". In this research, unvisited customers are referred as "dummy tour customers" in order to cover the customer set and their tour info is stored as 0 in the data structure. Additionally, the dummy tour can be empty in the case of visiting all customers.

Applying solely the *Intra-Tour Improvement Heuristics* may not yield satisfactory results if the dummy tour customers are not evaluated and included in the initial solution. Therefore, an adequate interaction between the initial solution which consists of the visited customers and the dummy tour which contains unvisited customers have to be fulfilled in order to diversify effectively over the solution space. This interaction is rendered by *Inter-Tour Improvement Heuristics*.

The *Inter-Tour Improvement Heuristics* are performed based on the same idea of swapping, inserting and removing vertices but these heuristics are used to maximise the profit instead of minimising just the total distances. The reason is that they manipulate both existing tours and the dummy tour. Thus, the revenues of the unvisited customers and the precedence constraint feasibility check are involved in the procedure.

4.4.2.1 The Inter Tour Exchange (ITE) and the Inter Dummy Exchange (IDE)

The *Inter Tour Exchange (ITE)* heuristic is one of the heuristics which enables the exchanges of customer pairs between the individual routes of a given solution. *ITE* simply compares two routes, exchanges pickup customers with pickup customers and delivery customers with delivery customers at the same positions. If the current solution is better than the original solution, then it is accepted. If it is not, swapped customers are swapped back. The heuristic terminates if all the routes in the solution are compared and there is no better solution.

ITE is explained in an example with 4 pairs of customers ((a,-a), (b,-b), (c,-c), (d,-d)).

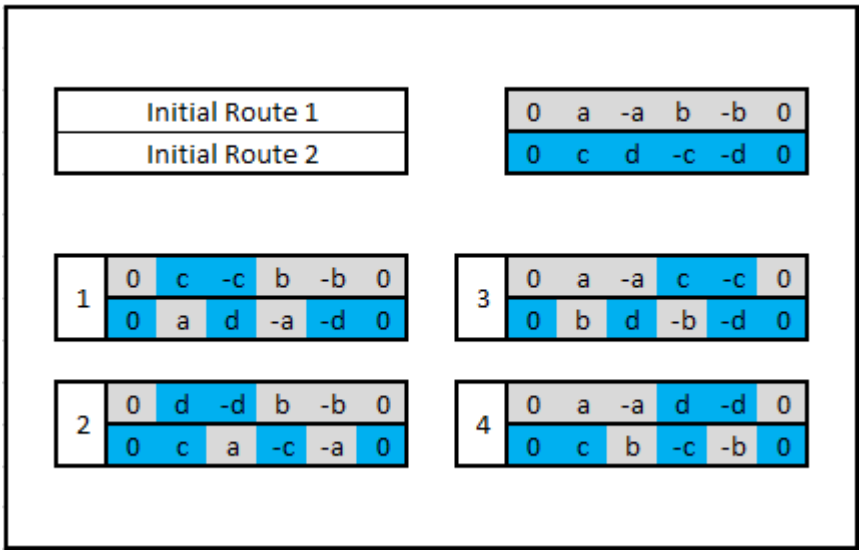


Table 19: An example for the *Inter Tour Exchange (ITE)*

The *Inter Dummy Exchange (IDE)* is performed by the same exchange operator as *ITE* but *IDE* compares each route of the solution with the dummy route (unvisited customers). Therefore, *IDE* explores a wider area over the solution space because the new customers are involved in the solution.

An example of *IDE* is illustrated below. The blue route is the dummy route in which all customers are ordered by their indices.

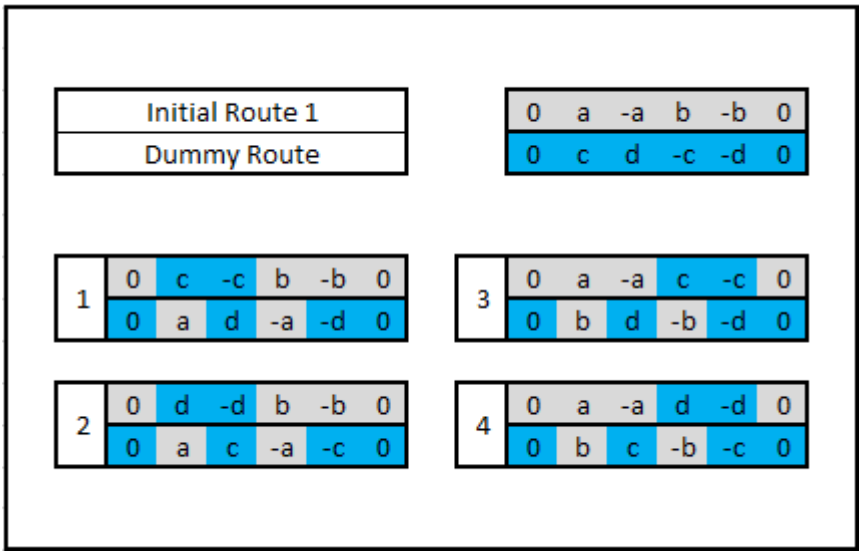


Table 20: An example for the *Inter Dummy Exchange (IDE)*

In order to reduce the *CPU* time, the exchange process is performed sequentially. The pickup customers are exchanged firstly before the new route total distance is checked. If the total distance is within the limits, then the delivery customers are swapped. It is observed that the sequential and conditional moves consume less time than the simultaneous moves (exchanging pickup and delivery customers at the same time).

4.4.2.2 The Inter Tour Insert (ITI) and the Inter Dummy Insert (IDI)

The *Inter Tour Insert (ITI)* and the *Inter Dummy Insert (IDI)* utilise the insert operator instead of the exchange operator. While *ITI* operates between the existing solution routes by inserting customers from one route to another, *IDI* inserts the unvisited customers to each solution route and expands the number of the customers which are visited. In both heuristics, the solutions with more profits are accepted. The precedence constraint is not needed to be checked owing to the indices but the demand capacity constraint has to be checked. Similar to *ITE* and *IDE*, the customers are moved sequentially because of the *CPU* time limitation.

ITI moves are displayed below with 2 routes and 3 pairs of customers. On the left part of the table the insertion process of the first customer pair and on the right part the second customer pair's insertion moves are exemplified.

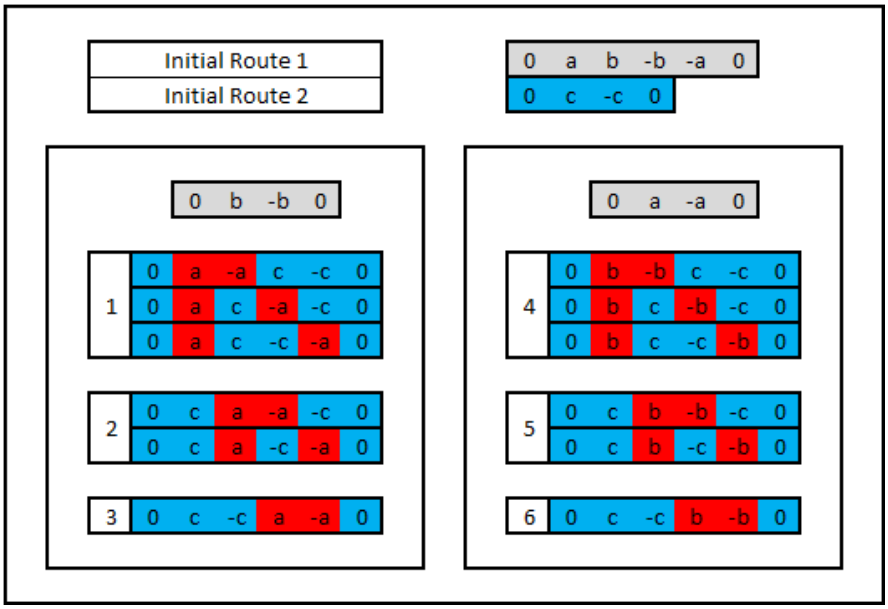


Table 21: An example for the *Inter Tour Insert (ITI)*

IDI's insertion pattern is akin to the *ITI's* pattern but in *IDI*, instead of initial Route 1 there exist dummy tour. The customers taken from the dummy tour are inserted to the solution routes.

4.4.2.3 The Gravity Centre Exchange (GCE)

The *Gravity Centre Exchange (GCE)* is an improvement heuristic. It removes the pair of customers which is farthest from the gravity point of the route and inserts the unvisited customers to the route until there is no customer to insert because of the total time constraint. The *GCE* focuses only the least contributing customer pair to the route. Thus, it can be considered as an extension of the *Inter Dummy Exchange (IDE)* heuristic.

Let V be the set of all vertices in a route. V consists of P pickup customers, D delivery customers and a depot. *n* addresses the customer pairs in a route, R_i ($i \in P$) refers to the revenues of the pairs. Each customer ($j \in V \backslash 0$) has two coordinate points in the 2-dimensional plane (X_j, Y_j).

$$V = \{P, D, 0\} \quad P = \{1,2, \dots, n\} \quad D = \{n+1, n+2, \dots, 2n\}$$

The weighted gravity point of a route is calculated for all customers excluding the depot. The revenues are the weights in this case. For each individual route in a solution, only one gravity point is calculated as $G = (G_x, G_y)$. Each coordinate point is multiplied by the corresponding revenue and the total sum is divided by the total revenue of the customers in the route. The sum is also divided by 2 because each pickup and delivery customers have only one revenue. The formulae are shown below.

$$G_x = \frac{\sum_{j \in V \backslash 0} R_j X_j}{2 \sum_{i \in P} R_i} \qquad G_y = \frac{\sum_{j \in V \backslash 0} R_j Y_j}{2 \sum_{i \in P} R_i}$$

Afterwards, for each customer pair in the route, both the pickup and delivery customers' Euclidean distances to the gravity point G is summed up. The customer pair whose total distances to the gravity point is highest is removed from the route.

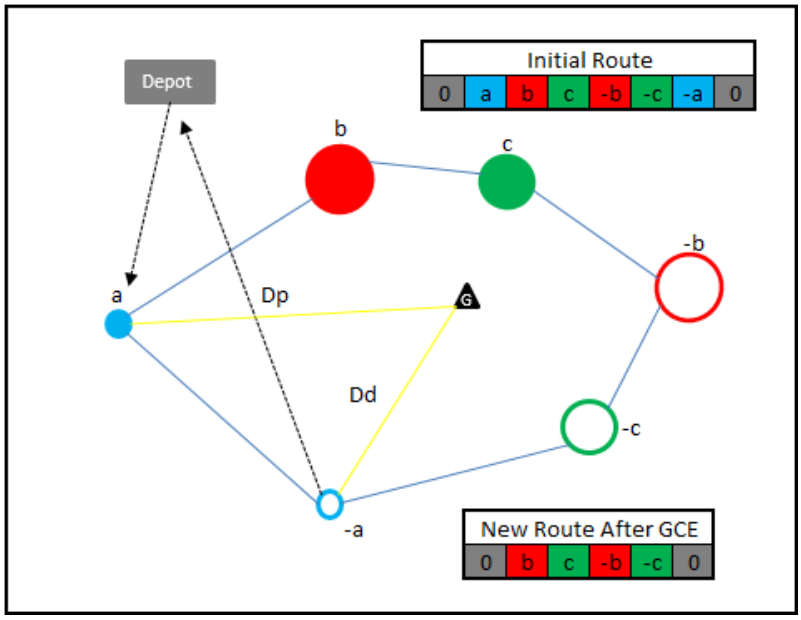


Figure 12: An example for the *Gravity Centre Exchange (GCE)*

An example of the *GCE* with 3 pairs of customers is illustrated above. The filled circles represent the pickup customers while the empty circles are the delivery customers. The magnitudes of the circles symbolise the revenues and the big circles belong to the profitable customers. Moreover, the gravity point (black triangle) is close to the customers which have bigger revenues and for each customer, the distance of pickup customer to the gravity point D_p and the distance of the delivery customer to the gravity point D_d is calculated. In the example, the yellow lines show the biggest distance $D = (D_p + D_d)$ among all pairs' distances. In this case customer pair (a,-a) has the biggest distance. Hence, the customer pair (a,-a) is removed. The new route consists of 2 pairs of customers. The black dashed lines show the original direction of the route but note that the depot is not taken into account in the calculation of the gravity point. Only the customers which are connected by the blue lines are calculated instead.

In the second leg of the process, the shrunk route is filled according to the *Inter Dummy Insert (IDI) heuristic*. The unvisited customers are inserted to the route beginning from the left to right in the data structure.

All in all, the *GCE* resembles to the *Inter Dummy Exchange*. The only difference is that the *GCE* removes the less contributing customer pair for the determination of the gravity point of the route.

4.5 The Local Search with Variable Neighbourhood Descent (VND)

The *Variable Neighbourhood Descent (VND)* is a heuristic which applies different neighbourhood operators to an initial solution in order to improve it. In this research, it is used as local search heuristic after the perturbation of the initial solution. There are several variants of the *VND* in the literature e.g. *Nested-VND*, *Sequential-VND*, *Mixed-VND*, etc.

This research presents a sequential and a self-adaptive *VND* as a local search for the *Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)*. For each *VND*, previously mentioned 11 neighbourhoods are used but the order of the neighbourhoods distinguishes the quality of the two methods.

4.5.1 The Sequential Variable Neighbourhood Descent (Seq-VND)

The *Sequential-VND* is based on the idea of exploring the neighbourhoods one by one in a sequential manner. The *Seq-VND* algorithm starts with a given neighbourhood and explores the solution space. As long as an improvement is found, it resumes to search within the same neighbourhood. If there is no improvement, it leaps to the second neighbourhood. Afterwards, if there is no improvement, other unused neighbourhoods are explored. In case of an improvement, it starts from the first neighbourhood. The process terminates when a local minimum (or maximum) for all neighbourhoods is detected. This means that the algorithm ends if there is no improvement for all neighbourhoods. The minimum (or maximum) of the *Seq-VND* is the minimum (or maximum) of all neighbourhoods in the *Seq-VND*.

```
Function Seq-VND(x`)
1  l ← 1;                                                        // Neighbourhood counter
2  while (l ≤ 11) do
3      If (l = 1) then
4          x'' ← arg min {f(x)|| x ∈ N_ITE(x')}                  // Apply ITE
5      If (l = 2) then
6          x'' ← arg min {f(x)|| x ∈ N_IDE(x')}                  // Apply DIE
7      If (l = 3) then
8          x'' ← arg min {f(x)|| x ∈ N_ITI(x')}                  // Apply ITI
9      If (l = 4) then
10         x'' ← arg min {f(x)|| x ∈ N_IDI(x')}                  // Apply IDI
11     If (l = 5) then
12         x'' ← arg min {f(x)|| x ∈ N_GCE(x')}                  // Apply GCE
13     If (l = 6) then
14         x'' ← arg min {f(x)|| x ∈ N_PFE(x')}                  // Apply PFE
15     If (l = 7) then
16         x'' ← arg min {f(x)|| x ∈ N_BI(x')}                   // Apply BI
17     If (l = 8) then
18         x'' ← arg min {f(x)|| x ∈ N_2-Opt(x')}                // Apply 2-OPT
19     If (l = 9) then
20         x'' ← arg min {f(x)|| x ∈ N_RLC(x')}                  // Apply RLC
21     If (l = 10) then
22         x'' ← arg min {f(x)|| x ∈ N_FI(x')}                   // Apply FI
23     If (l = 11) then
24         x'' ← arg min {f(x)|| x ∈ N_PFE(x')}                  // Apply PFE
25     If (f(x'') > f(x')) then
26         x' ← x''; l ← 1;                                      // Restart
27     Else
28         l ← l + 1;                                            // Next Neighbourhood
29 Retun x'
```

Table 22: The Sequential-VND pseudo-code with 11 neighbourhoods (Own representation based on *Mladenović et al., 2013*)

Since 11 neighbourhoods are used in this research, the variable *l* has to be smaller than or equal to 11. The order of the *Seq-VND* affects the solution quality a lot. At this point, a natural question arises: "In which order should the neighbourhoods be used?"

In many research articles, the order is determined by the experience (testing a few orders roughly) or randomly. Thus, it is reasonable to order the neighbourhoods in an increasing order. The reason is that the first neighbourhoods are implemented more often and if a time-consuming neighbourhood is applied, then the total *CPU* time increases substantially.

4.5.2 The Self-Adaptive Variable Neighbourhood Descent (Sa-VND)

In order to control the entire neighbourhoods in an effective way, *Hu and Raidl (2006)* propose a *self-adaptive VND* which adapts itself to the entire process. It applies the solution-improving neighbourhoods more often by changing the order of them through the execution.

According to *Hu and Raidl (2006),* an initial order of neighbourhoods $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ is chosen randomly or in an intuitive way. Each neighbourhood N_i, $N = (N_1, N_2, \dots, N_n)$ is associated with a rating value $W_i > 0$. In the local search process, after a neighbourhood N_{λ_i} of an initial solution x is explored, the rating belonging to neighbourhood W_{λ_i} is updated. If that specific neighbourhood does not improve the solution x, then the *CPU* time t_{λ_i} of that neighbourhood is added to its rating W_{λ_i}. Otherwise, the rating is halved and $\frac{t_{\lambda_i}}{\alpha}$ value is added. α is a parameter which can be fit in the process.

If any rating value W_i is smaller than the current W_{min} or bigger than the current W_{max}, the order λ is updated and search continues as the same method.

The *Sa-VND* aims at applying the most profitable neighbourhoods frequently and reducing the *CPU* time. Thus, the algorithm has more control over the order of the neighbourhoods.

	Sa-VND(x)
1	$w_1 = w_2 = \ldots = w_n = \mathrm{W}$
2	$w_{min} = w_{max} = \mathrm{W}$
3	$\lambda = (1, 2, \ldots, n)$
4	$i = 1$
5	Repeat
6	Find the best neighbour x' $\in N_{\lambda_i}(\mathrm{x})$, requiring time t_{λ_i}
7	If x' is better than x then
8	$\mathrm{x} = x'$
9	$w_{\lambda_i} = {}^{w_{\lambda_i}}/_{2} + {}^{t_{\lambda_i}}/_{\alpha}$
10	$\mathrm{i} = 1$
11	Else
12	$w_{\lambda_i} = w_{\lambda_i} + t_{\lambda_i}$
13	$\mathrm{i} = i + 1$
14	If $w_{\lambda_i} < w_{min} \vee w_{\lambda_i} > w_{max}$ then
15	$nextN = \lambda_i$ // Store the neighborhood to be considered next
16	sort $\lambda_1, \ldots, \lambda_n$ s.t. $w_1 \leq w_2 \leq \ldots \leq w_{\lambda_n}$
17	$w_{min} = w_{\lambda_1}$
18	$w_{max} = w_{\lambda_n}$
19	reset i s.t. $\lambda_i = nextN$
20	Until $i > n$

Table 23: The *Sa-VND* pseudo-code (Own representation based on *Hu and Raidl, 2006*)

5 The Computational Experiment

The computational experiment aims at quantitatively measuring the performance of the *General Variable Neighbourhood Search (GVNS)* algorithm for the *Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)* with different and newly created data instances and comparing the results of different variations of the algorithm to draw a reasonable conclusion.

Since two versions of the construction heuristics (*C1* and *C2*) and *Variable Neighbourhood Descent* (*Seq-VND* and *Sa-VND*) are developed, all combinations of the algorithm with these components are needed to be tested. Hence, the general framework of the algorithm is preserved. The variations comprise of *C1 – Seq-VND*, *C2 – Seq-VND*, *C1 – Sa-VND* and C2 – *Sa-VND*. Likewise, the performance of the construction heuristics (*C1* and *C2*) is tested individually and compared in order to comprehend how good the initial solutions are and how much the initial solutions are improved in the end.

The *GVNS* algorithm is coded in *C++* programming language and compiled in the *Microsoft Visual Studio Express 2013*. All data instances are executed *in Intel Core i5 – 4200U 1.6GHz* computer with *4 GB RAM*. Thus, the computer has an operating system of *Microsoft*.

Next sub-chapter explains the logic behind the creation of the data instances and renders the further details.

5.1 Test Instances

In the literature, there exist various research articles in which the *Pickup and Delivery Problem (PDP)* algorithms are presented. Therefore, there is no algorithm in the literature exactly for the *MVPPDP* and no test instance is created. The existing benchmark instances have different features.

For this reason, 36 new data instances are randomly generated in *C++* programming language. The numbers of the customers are set to be 20, 50, 100, 250, 500 and 1000. Customers are scattered on a two-dimensional plane with an x and y axis. The coordinates (x, y) are generated as integers between [-1000, 1000] excluding the point (0, 0) which is the location of the depot. The range [-1000, 1000] is deliberately chosen to allow further creations of the data with more customers. Each customer pair or request has an integer demand value between [1, 50]. Demands are positive for the pickup customers and negative for the delivery customers because it is supposed that positive demand loads are dropped at the delivery customers.

The amounts of the revenues are decisive in terms of testing the behaviour of the algorithm in different conditions so the revenues are generated as to be fixed for all customers, proportional to the demands and randomly.

Table 24 illustrates the structure of the test instances. Three different types of revenues are generated for each group of customers (20, 50, 100, 250, 500 and 1000). Additionally, the constraints for some instances are tightened and for the remaining samples, they are relaxed for enabling the generation of relatively short and long routes. The aim is to check if the

algorithm can find good solutions for each route which can be considered as small-scaled *Traveling Salesman Problems*.

In total, 36 instances are generated and each instance has an index number between [1, 36] as shown in Table 24.

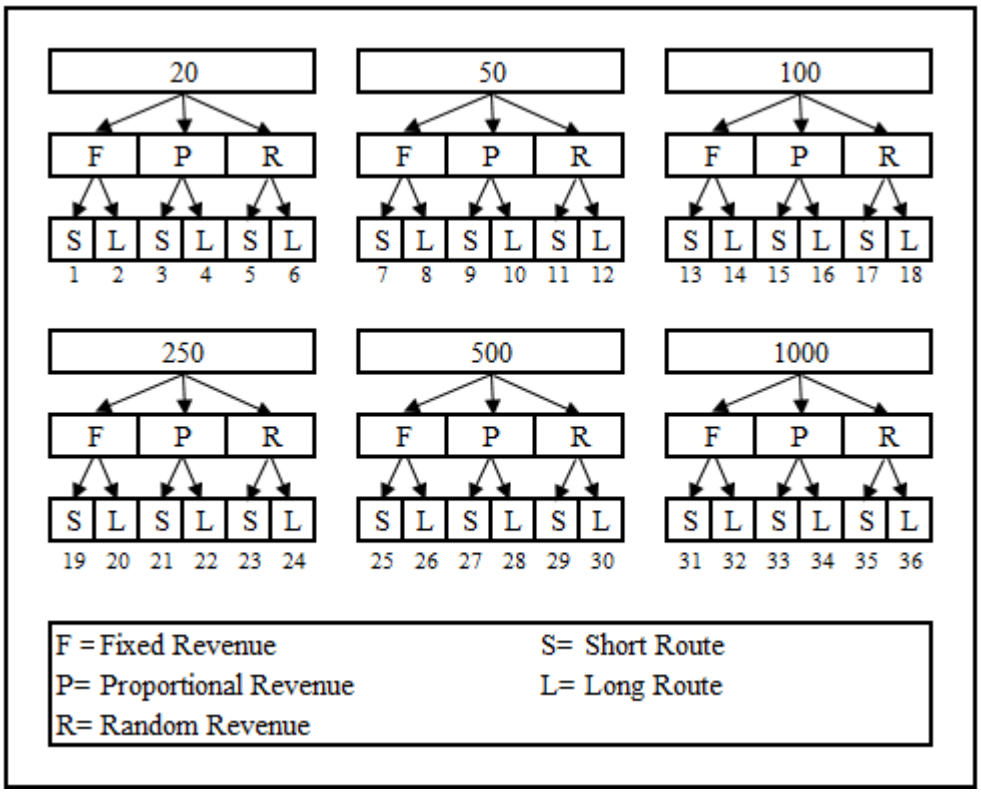


Table 24: The structure of the data instances

The test instances are named as [Index|The capital letters of the author|Revenue type|The length of the route|Number of the customers]. E.g. 1-MR-F-S-20

Tuning the constraints is important as well. In *MVPPDP*, all customers do not have to be served but might be. The constraints are tuned to allow the customer cover percentage to be between (0, 100] and determined experimentally. So in some cases, all customers are served but there is no case in which no customer is visited.

Additionally, the vehicle numbers for each instance vary between [2, 8] while the vehicle capacities are between [50, 120] and the total time limits diversify within [2500, 15000].

Revenues are also determined through the experiment. Table 25 summarises the ranges of the fixed and random revenues and the multipliers for the proportional revenues. In the proportional case, the multipliers below are multiplied by the demands of each request and proportional revenues are determined. It should be noted that as the number of the customers increases, the amounts of the revenues decrease to balance the outcome.

	Fixed	Proportional	Random
20	5000	400	1000-10000
50	4000	300	1000-10000
100	3000	200	500-8000
250	2000	100	500-6000
500	2000	100	500-5000
1000	1000	100	100-4000

Table 25: Revenue ranges and multipliers

5.2 Results

In order to comprehend and interpret the final results, initial solutions which are the outcomes of the construction heuristics must be found in advance. So the construction heuristic algorithms (*The Greedy Heuristic* and *Two-stage Cheapest Insertion Heuristic*) are executed once for all data instances and compared with each other. The reason for running them precisely once is that the construction heuristics are deterministic and yield the same result for every run. The results are shown in Table 26.

Data	Greedy (C1)			Cheapest Insertion (C2)			Profit Gap
	Profit	Pairs	Cover %	Profit	Pairs	Cover %	
1-MR-F-S-20	16185,4	5	50,0	21400	6	60,0	32,2%
2-MR-F-L-20	22006,2	7	70,0	32400,1	9	90,0	47,2%
3-MR-P-S-20	39646,5	4	40,0	43528	5	50,0	9,8%
4-MR-P-L-20	54849,9	7	70,0	55775,5	10	**100,0**	1,7%
5-MR-R-S-20	22954,4	4	40,0	26884	4	40,0	17,1%
6-MR-R-L-20	30486,7	6	60,0	41933	10	**100,0**	37,5%
7-MR-F-S-50	15000,5	5	20,0	17958,2	6	24,0	19,7%
8-MR-F-L-50	34988,2	12	48,0	43008,3	14	56,0	22,9%
9-MR-P-S-50	53694,8	5	20,0	38796,9	4	16,0	**-27,7%**
10-MR-P-L-50	99109	10	40,0	62731,2	11	44,0	**-36,7%**
11-MR-R-S-50	19789,7	3	12,0	24619,2	6	24,0	24,4%
12-MR-R-L-50	45326,9	8	32,0	51723,3	14	56,0	14,1%
13-MR-F-S-100	19033,1	11	22,0	28818,1	14	28,0	51,4%
14-MR-F-L-100	31825,6	18	36,0	55322,2	26	52,0	73,8%
15-MR-P-S-100	44329,1	7	14,0	46547,4	12	24,0	5,0%
16-MR-P-L-100	69459,3	11	22,0	79541,3	25	50,0	14,5%
17-MR-R-S-100	49912,7	9	18,0	70214,6	13	26,0	40,7%
18-MR-R-L-100	63116,1	12	24,0	91663,1	25	50,0	45,2%
19-MR-F-S-250	27676,1	25	20,0	40906,1	32	25,6	47,8%
20-MR-F-L-250	44456	43	34,4	67845,1	55	44,0	52,6%
21-MR-P-S-250	63930	22	17,6	43247,3	23	18,4	**-32,4%**
22-MR-P-L-250	112471	38	30,4	94126,8	57	45,6	**-16,3%**
23-MR-R-S-250	79486,1	22	17,6	102873	31	24,8	29,4%
24-MR-R-L-250	130371	36	28,8	143886	54	43,2	10,4%
25-MR-F-S-500	49210	43	17,2	78580,2	59	23,6	59,7%
26-MR-F-L-500	73299,8	68	27,2	135652	99	39,6	85,1%
27-MR-P-S-500	124075	39	15,6	84513,6	52	20,8	**-31,9%**
28-MR-P-L-500	179001	62	24,8	169098	103	41,2	**-5,5%**
29-MR-R-S-500	108049	35	14,0	116568	57	22,8	7,9%
30-MR-R-L-500	170205	58	23,2	218965	102	40,8	28,6%
31-MR-F-S-1000	27655,6	94	18,8	66345,3	133	26,6	139,9%
32-MR-F-L-1000	32078,4	148	29,6	112840	226	45,2	251,8%
33-MR-P-S-1000	264997	81	16,2	152307	117	23,4	**-42,5%**
34-MR-P-L-1000	380514	121	24,2	318268	222	44,4	**-16,4%**
35-MR-R-S-1000	193390	81	16,2	197083	125	25,0	1,9%
36-MR-R-L-1000	275805	123	24,6	362266	224	44,8	31,3%

Table 26: The results of the construction heuristics. The data names are written in the first column of the table. Due to convenience, data names are addressed by their index numbers for the next result tables.

Columns 2 and 5 of Table 26 are the obtained profits, Columns 3 and 6 are the numbers of the pairs which are served among all customer pairs, and Columns 4 and 7 are the percentages of covering the customers. Finally, Column 8 include the gap percentages of the profits of (*C2-C1*).

All experimental results in this research are guaranteed to be feasible with a function located the end of the algorithm and not included in the *CPU* time. The function checks the final results in terms of the total time, vehicle capacity and the precedence constraint. If there is an infeasible solution, it alerts the infeasibility. During the computational experiment, there has not been any warning and yet all solutions are feasible.

The *GVNS* algorithm consists of two primary sections one of which is deterministic (construction heuristic and local search) and the latter is stochastic (shaking). On the grounds of that, it is crucial to run the whole algorithm more than once to check the competence of the *GVNS*.

In this research, the entire algorithm is run 5 times for each data sample and for the variations of the algorithm. Since there are two options for the construction and *VND* sections, *C1 – Seq-VND*, *C2 – Seq-VND*, *C1 – Sa-VND* and *C2 – Sa-VND* are tested by fixing the shaking step and the general framework each time. The stopping criterion is chosen as a 5-minute time limit for every run. On the other hand, the number of iterations are counted to figure out how many times the solution is perturbed and searched locally. So one iteration purports applying the shaking and *VND* once.

The order of the neighbourhoods affects the final result profoundly in the *Sequential-VND*. There exist 11 neighbourhoods in this *GVNS* algorithm. Hence, the number of possible orders is *11!* which is high enough to not compute all to find the best order. Another approach is to choose the order randomly or intuitively but for the sake of applying a satisfactory order, instead of ordering neighbourhoods individually, main groups of neighbourhoods in this research are investigated. Ordering the chief groups might be:

- Locating the inter-tour neighbourhoods firstly and afterwards the intra-tour neighbourhood or vice versa
- Locating the intra-tour operators in between inter-tour operators or vice versa
- Locating the neighbourhoods with exchange operator firstly and the other neighbourhoods later. Similarly, the insert operators can be located firstly.

Once a main group is set, the inner order of the group does not affect the solution significantly. (E.g. If inter-tour neighbourhoods are located first and then the intra-tour

neighbourhoods, the order of the inter-tour operators (*PFE, PBE, 2-Opt, RLC, FI, BI*) does not change the final result as much as applying all neighbourhoods randomly or intuitively).

A relatively good order is determined through the experiment. Nonetheless, it is no guaranteed that the order is the best order.

The order of the neighbourhoods		
1	The Inter Tour Exchange	ITE
2	The Inter Dummy Exchange	IDE
3	The Inter Tour Insert	ITI
4	The Inter Dummy Insert	IDI
5	The Gravity Centre Exchange	GCE
6	The Pairwise Forward Exchange	PFE
7	The Backward Insertion	BI
8	The Two-Opt	2-Opt
9	The Relocate Pairs	RLC
10	The Forward Insertion	FI
11	The Pairwise Backward Exchange	PBE

Table 27: The order of the neighbourhoods in the *Sequential-VND*

Table 28 and 29 present the results of the experiment for the algorithm with *Sequential VND* and the construction heuristics *C1, C2*. Data names are indexed in the very left column. The second column gives the average profits of 5 runs as the third column compares the average profits of final results with the initial solutions (*C1* and *C2*) and states the improvements which are achieved by the shaking and local search steps. The fourth column shows the number of the iterations. Columns 5, 6 and 7 are average numbers of customers which are served, comparison of the average number of the visited customers to the number of customers found by the construction heuristic and customer cover percentage, respectively. The subsequent two columns are the best and worst profits next to the best known solution. The last three columns are the gaps of average, best and worst results compared to the best known solutions.

Best known solutions are acquired during the process of algorithm development via executing the algorithm for many hours (up to 20 hours) or at the time of the computational experiment.

C1 - Sequential VND													
Data	Ave. Profit		Ave Iter.	Ave. # of Customers			Best-Worst Results			Gaps		
	Ave. Profit (a)	Imp.	# Iter.	# V. Pairs	Imp	Cover %	Best Profit (b)	Worst Profit (c)	Best Known (d)	Gap (d-a)	Gap (d-b)	Gap (d-c)
1	30315,5	87,3%	2443704	8	60,0%	80,0%	30315,5	30315,5	30315,5	0,0%	0,0%	0,0%
2	38765,6	76,2%	825343	10	42,9%	100,0%	38765,6	38765,6	39340,3	1,5%	1,5%	1,5%
3	51473,4	29,8%	1861929	6	50,0%	60,0%	51473,4	51473,4	52855,2	2,7%	2,7%	2,7%
4	58365,6	6,4%	982234	10	42,9%	100,0%	58365,6	58365,6	58365,6	0,0%	0,0%	0,0%
5	36469,5	58,9%	2696969	8	100,0%	80,0%	36469,5	36469,5	36469,5	0,0%	0,0%	0,0%
6	43368,6	42,3%	879389	10	66,7%	100,0%	43368,6	43368,6	43764,3	0,9%	0,9%	0,9%
7	22306	48,7%	1005524	7	40,0%	28,0%	22306	22306	22441	0,6%	0,6%	0,6%
8	58050,8	65,9%	277339	18	50,0%	72,0%	58050,8	58050,8	58050,8	0,0%	0,0%	0,0%
9	57651,7	7,4%	1099734	6	20,0%	24,0%	57651,7	57651,7	57651,7	0,0%	0,0%	0,0%
10	132648	33,8%	251115	16	60,0%	64,0%	132648	132648	132648	0,0%	0,0%	0,0%
11	33340,4	68,5%	920362	7	133,3%	28,0%	33340,4	33340,4	33340,4	0,0%	0,0%	0,0%
12	79350,3	75,1%	234242	18	125,0%	72,0%	79350,3	79350,3	79350,3	0,0%	0,0%	0,0%
13	41464,84	117,9%	160336	19	70,9%	37,6%	42174,4	39630,9	44402,3	7,1%	5,3%	12,0%
14	81271,2	155,4%	75691	35	93,3%	69,6%	81999,4	79306	81999,4	0,9%	0,0%	3,4%
15	70387,38	58,8%	182012	14	100,0%	28,0%	70836	68734,3	71512,4	1,6%	1,0%	4,0%
16	124288	78,9%	69166	29	161,8%	57,6%	124770	123381	125032	0,6%	0,2%	1,3%
17	76885,4	54,0%	154826	17	86,7%	33,6%	80878,8	72550,8	80878,8	5,2%	0,0%	11,5%
18	131052,6	107,6%	69037	32	165,0%	63,6%	131473	130460	131845	0,6%	0,3%	1,1%
19	73259,94	164,7%	19511	49	95,2%	39,0%	73610,9	72075,7	75309,1	2,8%	2,3%	4,5%
20	132996	199,2%	4586	88	105,6%	70,7%	134889	130534	136594	2,7%	1,3%	4,6%
21	111334,4	74,2%	18845	40	83,6%	32,3%	112427	110139	113886	2,3%	1,3%	3,4%
22	197899,8	76,0%	5411	79	107,4%	63,0%	200402	195462	200402	1,3%	0,0%	2,5%
23	162410,8	104,3%	18832	44	100,0%	35,2%	162960	161492	164070	1,0%	0,7%	1,6%
24	269072,8	106,4%	5510	80	122,2%	64,0%	271185	265983	274563	2,0%	1,2%	3,2%
25	158425,4	221,9%	3545	100	132,1%	39,9%	161063	156463	167080	5,5%	3,7%	6,8%
26	243210	231,8%	1005	154	126,5%	61,6%	249171	236906	259319	6,6%	4,1%	9,5%
27	201195,4	62,2%	3321	77	98,5%	31,0%	203832	197604	206788	2,8%	1,5%	4,6%
28	319239,2	78,3%	1253	135	118,4%	54,2%	321471	316354	331291	3,8%	3,1%	4,7%
29	271154,4	151,0%	3591	93	165,7%	37,2%	279105	268097	281711	3,9%	0,9%	5,1%
30	402557,2	136,5%	1171	145	150,7%	58,2%	407689	396339	417282	3,7%	2,4%	5,3%
31	150920,4	445,7%	378	222	136,2%	44,4%	154196	146676	173826	**15,2%**	**12,7%**	**18,5%**
32	218211	580,2%	60	337	127,4%	67,3%	221958	212699	247385	**13,4%**	**11,5%**	**16,3%**
33	439681,8	65,9%	415	188	132,6%	37,7%	450349	433735	464580	5,7%	3,2%	7,1%
34	678396,2	78,3%	122	299	147,4%	59,9%	693407	667446	712918	5,1%	2,8%	6,8%
35	455054,6	135,3%	437	204	152,3%	40,9%	463625	450433	492083	8,1%	6,1%	9,2%
36	668655	142,4%	75	322	161,8%	64,4%	673951	663984	705167	5,5%	4,6%	6,2%

Table 28: The results of *C1-Sequential VND* combination

C2 - Sequential VND													
Data	Ave. Profit		Ave Iter.	Ave. # of Customers			Best-Worst Results			Gaps		
	Ave. Profit (a)	Imp.	# Iter.	# V. Pairs	Imp	Cover %	Best Profit (b)	Worst Profit (c)	Best Known (d)	Gap (d-a)	Gap (d-b)	Gap (d-c)
1	30315,5	41,7%	2517611	8	33,3%	80,0%	30315,5	30315,5	30315,5	0,0%	0,0%	0,0%
2	38765,6	19,6%	829418	10	11,1%	100,0%	38765,6	38765,6	39340,3	1,5%	1,5%	1,5%
3	52128,4	19,8%	1963341	6	20,0%	60,0%	52565,1	51473,4	52855,2	1,4%	0,6%	2,7%
4	58365,6	4,6%	1023144	10	0,0%	100,0%	58365,6	58365,6	58365,6	0,0%	0,0%	0,0%
5	36469,5	35,7%	2855964	8	100,0%	80,0%	36469,5	36469,5	36469,5	0,0%	0,0%	0,0%
6	43368,6	3,4%	919363	10	0,0%	100,0%	43368,6	43368,6	43764,3	0,9%	0,9%	0,9%
7	22306,0	24,2%	1114252	7	16,7%	28,0%	22306	22306	22441	0,6%	0,6%	0,6%
8	58050,8	35,0%	292219	18	28,6%	72,0%	58050,8	58050,8	58050,8	0,0%	0,0%	0,0%
9	57651,7	48,6%	1196131	6	50,0%	24,0%	57651,7	57651,7	57651,7	0,0%	0,0%	0,0%
10	132648,0	111,5%	268141	16	45,5%	64,0%	132648	132648	132648	0,0%	0,0%	0,0%
11	33340,4	35,4%	992522	7	16,7%	28,0%	33340,4	33340,4	33340,4	0,0%	0,0%	0,0%
12	79350,3	53,4%	250481	18	28,6%	72,0%	79350,3	79350,3	79350,3	0,0%	0,0%	0,0%
13	41955,2	45,6%	119506	19	35,7%	38,0%	42174,4	41846	44402,3	5,8%	5,3%	6,1%
14	81971,0	48,2%	77320	35	34,6%	70,0%	81999,4	81857,2	81999,4	0,0%	0,0%	0,2%
15	70398,5	51,2%	183933	13	11,7%	26,8%	71326,4	68734,3	71512,4	1,6%	0,3%	4,0%
16	123931,0	55,8%	72246	29	15,2%	57,6%	124770	121787	125032	0,9%	0,2%	2,7%
17	78418,8	11,7%	166601	16	24,6%	32,4%	78818	76821,8	80878,8	3,1%	2,6%	5,3%
18	130670,4	42,6%	71135	32	27,2%	63,6%	131845	129847	131845	0,9%	0,0%	1,5%
19	72445,3	77,1%	19942	48	50,6%	38,6%	73575,7	71971,7	75309,1	4,0%	2,4%	4,6%
20	132438,0	95,2%	4717	88	60,4%	70,6%	134186	130394	136594	3,1%	1,8%	4,8%
21	111817,8	158,6%	19241	41	77,4%	32,6%	112480	111113	113886	1,8%	1,3%	2,5%
22	197214,2	109,5%	5596	79	37,9%	62,9%	198611	195168	200402	1,6%	0,9%	2,7%
23	162448,2	57,9%	19367	44	43,2%	35,5%	164070	161399	164070	1,0%	0,0%	1,7%
24	268713,8	86,8%	5740	82	52,2%	65,8%	271193	267089	274563	2,2%	1,2%	2,8%
25	159146,2	102,5%	3837	100	69,5%	40,0%	161040	155888	167080	5,0%	3,8%	7,2%
26	244591,6	80,3%	1119	155	56,2%	61,8%	249661	239559	259319	6,0%	3,9%	8,2%
27	202102,2	139,1%	3710	79	51,5%	31,5%	205480	199248	206788	2,3%	0,6%	3,8%
28	321181,6	89,9%	1264	136	31,8%	54,3%	325511	315234	331291	3,1%	1,8%	5,1%
29	272715,2	134,0%	3735	93	62,8%	37,1%	278233	268071	281711	3,3%	1,3%	5,1%
30	406689,6	85,7%	1197	147	43,9%	58,7%	409755	402660	417282	2,6%	1,8%	3,6%
31	151943,2	129,0%	377	223	67,4%	44,5%	154932	150401	173826	**14,4%**	**12,2%**	**15,6%**
32	218213,0	93,4%	41	336	48,8%	67,3%	221322	213528	247385	**13,4%**	**11,8%**	**15,9%**
33	441120,2	189,6%	418	189	61,4%	37,8%	450337	432125	464580	5,3%	3,2%	7,5%
34	651754,6	104,8%	79	309	39,3%	61,8%	672910	639534	712918	9,4%	5,9%	**11,5%**
35	461394,0	134,1%	442	206	64,6%	41,2%	472626	451891	492083	6,7%	4,1%	8,9%
36	662605,4	82,9%	51	320	43,0%	64,1%	669926	660106	705167	6,4%	5,3%	6,8%

Table 29: The results of *C2-Sequential VND* combination

The *Self-Adaptive VND* is the counterpart of the *Sequential-VND*. To test the quality of both local search heuristics, it is crucial to compare them. So the *Self-Adaptive VND* is tested with *C1* and *C2* in the same manner with testing the *Seq-VND*. As it is stated in Chapter 4.5.2, in order to launch the *Sa-VND*, initial rating values $W_i > 0$ associated with the neighbourhoods have to be set at the beginning. The ratings might be assigned randomly or intuitively but in this research, the averages of the *CPU* times of executing the neighbourhoods individually right after the construction heuristic *C2* are taken into consideration. *C2* is chosen because it is faster than *C1* and there is less improvement achieved in the local search part. Parameter α is chosen as 1 because the ratings are small enough. Since testing small data samples takes very short time, the bigger instances are tested. Even with the large-size instances, the *CPU* times are so trivial in seconds in many cases.

	PFE	PBE	2-OPT	FI	BI	RLC	ITE	DTE	ITI	DTI	GCE
19-MR-F-S-250	0	0	0	0,001	0,001	0	0,001	0,006	0,006	0,013	0,036
20-MR-F-L-250	0	0,001	0,001	0,004	0,002	0,001	0,002	0,011	0,206	0,011	0,088
21-MR-P-S-250	0	0,001	0,001	0	0,001	0,001	0,002	0,007	0,004	0,004	0,024
22-MR-P-L-250	0,001	0,001	0,002	0,003	0,002	0	0,002	0,012	0,035	0,015	0,087
23-MR-R-S-250	0	0,001	0,001	0,001	0,001	0,001	0,001	0,003	0,002	0,007	0,009
24-MR-R-L-250	0,001	0,001	0,001	0,002	0,006	0	0,001	0,012	0,138	0,014	0,063
25-MR-F-S-500	0,001	0,001	0	0,002	0,004	0,001	0,002	0,035	0,038	0,036	0,252
26-MR-F-L-500	0,001	0,001	0,001	0,004	0,005	0,001	0,003	0,033	0,809	0,031	0,102
27-MR-P-S-500	0,001	0,001	0	0,001	0,001	0,001	0,001	0,047	0,045	0,04	0,342
28-MR-P-L-500	0,001	0,002	0,001	0,009	0,008	0,001	0,009	0,049	0,307	0,025	0,087
29-MR-R-S-500	0	0,001	0	0,002	0,001	0,001	0,001	0,02	0,03	0,03	0,106
30-MR-R-L-500	0,002	0,001	0,001	0,011	0,007	0	0,005	0,042	0,362	0,051	0,115
31-MR-F-S-1000	0,007	0,004	0,001	0,017	0,02	0,001	0,009	0,19	1,975	0,214	2,718
32-MR-F-L-1000	0,016	0,015	0,004	0,128	0,092	0,003	0,03	0,294	43,394	0,341	4,083
33-MR-P-S-1000	0,003	0,002	0,001	0,006	0,008	0,001	0,01	0,501	1,093	0,242	1,636
34-MR-P-L-1000	0,011	0,008	0,002	0,04	0,053	0,004	0,046	0,473	46,245	0,323	3,636
35-MR-R-S-1000	0,002	0,002	0,001	0,008	0,01	0,001	0,013	0,34	2,242	0,358	1,264
36-MR-R-L-1000	0,022	0,018	0,006	0,068	0,075	0,004	0,034	0,339	35,765	0,35	1,235
AVERAGE	0,0038	0,0034	0,0013	0,0171	0,0165	0,0012	0,0096	0,1341	7,3720	0,1169	0,8824

1	RLC	0,0012
2	2-OPT	0,0013
3	PBE	0,0034
4	PFE	0,0038
5	ITE	0,0096
6	BI	0,0165
7	FI	0,0171
8	IDI	0,1169
9	IDE	0,1341
10	GCE	0,8824
11	ITI	7,3720

Min	0,001
Max	7,372

Table 30: Setting the rating values of the neighbourhoods for the *Sa-VND*

The results of the *Sa-VND* with *C1* and *C2* are presented in Table 31 and Table 32 respectively. The format of the tables is the same with Table 28 and Table 29.

C1 - Self Adaptive VND												
Data	Ave. Profit		Ave Iter.	Ave. # of Customers			Best-Worst Results			Gaps		
	Ave. Profit (a)	Imp.	# Iter.	# V. Pairs	Imp	Cover %	Best Profit (b)	Worst Profit (c)	Best Known (d)	Gap (d-a)	Gap (d-b)	Gap (d-c)
1	30315,5	87,3%	3230284	8	60,0%	80,0%	30315,5	30315,5	30315,5	0,0%	0,0%	0,0%
2	37866,3	72,1%	1656550	10	42,9%	100,0%	38765,6	37266,7	39340,3	3,9%	1,5%	5,6%
3	52565,1	32,6%	2304400	6	50,0%	60,0%	52565,1	52565,1	52855,2	0,6%	0,6%	0,6%
4	58365,6	6,4%	1807270	10	42,9%	100,0%	58365,6	58365,6	58365,6	0,0%	0,0%	0,0%
5	36469,5	58,9%	3352715	8	100,0%	80,0%	36469,5	36469,5	36469,5	0,0%	0,0%	0,0%
6	43368,6	42,3%	1390762	10	66,7%	100,0%	43368,6	43368,6	43764,3	0,9%	0,9%	0,9%
7	22306,0	48,7%	1071408	7	40,0%	28,0%	22306	22306	22441	0,6%	0,6%	0,6%
8	57303,4	63,8%	414858	18	48,3%	71,2%	58050,8	54582,8	58050,8	1,3%	0,0%	6,4%
9	57651,7	7,4%	919571	6	20,0%	24,0%	57651,7	57651,7	57651,7	0,0%	0,0%	0,0%
10	128353,0	29,5%	444751	16	60,0%	64,0%	132648	124733	132648	3,3%	0,0%	6,3%
11	33340,4	68,5%	1146566	7	133,3%	28,0%	33340,4	33340,4	33340,4	0,0%	0,0%	0,0%
12	77655,5	71,3%	450220	18	122,5%	71,2%	79350,3	70876,4	79350,3	2,2%	0,0%	**12,0%**
13	41523,5	118,2%	219224	19	70,9%	37,6%	42215,5	39498,6	44402,3	6,9%	5,2%	**12,4%**
14	80080,4	151,6%	116144	34	91,1%	68,8%	81621	76196,8	81999,4	2,4%	0,5%	7,6%
15	71113,8	60,4%	190781	14	100,0%	28,0%	71399,2	70774,1	71512,4	0,6%	0,2%	1,0%
16	122691,8	76,6%	113882	30	176,4%	60,8%	123766	121772	125032	1,9%	1,0%	2,7%
17	76954,4	54,2%	202110	16	77,8%	32,0%	80878,8	72846	80878,8	5,1%	0,0%	**11,0%**
18	130403,4	106,6%	111590	32	166,7%	64,0%	131339	129273	131845	1,1%	0,4%	2,0%
19	72703,0	162,7%	28994	48	93,6%	38,7%	73835,5	71765,9	75309,1	3,6%	2,0%	4,9%
20	132394,4	197,8%	9630	88	104,7%	70,4%	132870	132031	136594	3,2%	2,8%	3,5%
21	109449,4	71,2%	29279	42	90,9%	33,6%	112428	107662	113886	4,1%	1,3%	5,8%
22	194131,2	72,6%	10790	83	117,9%	66,2%	195135	193426	200402	3,2%	2,7%	3,6%
23	158164,4	99,0%	28398	44	100,9%	35,4%	162115	156169	164070	3,7%	1,2%	5,1%
24	266559,0	104,5%	11661	83	131,7%	66,7%	272957	262541	274563	3,0%	0,6%	4,6%
25	157399,8	219,9%	5525	99	130,7%	39,7%	159064	153128	167080	6,2%	5,0%	9,1%
26	245600,8	235,1%	2015	155	127,9%	62,0%	247986	241314	259319	5,6%	4,6%	7,5%
27	197227,6	59,0%	5645	87	123,6%	34,9%	200764	191561	206788	4,8%	3,0%	7,9%
28	315854,6	76,5%	2345	134	116,1%	53,6%	320872	311214	331291	4,9%	3,2%	6,5%
29	271264,2	151,1%	5638	96	173,7%	38,3%	274419	265838	281711	3,9%	2,7%	6,0%
30	399990,8	135,0%	2228	145	149,7%	57,9%	407849	395830	417282	4,3%	2,3%	5,4%
31	151990,0	449,6%	659	223	137,0%	44,6%	154357	146284	173826	**14,4%**	**12,6%**	**18,8%**
32	223999,8	598,3%	187	342	131,4%	68,5%	224999	221903	247385	**10,4%**	9,9%	**11,5%**
33	433353,8	63,5%	776	187	131,1%	37,4%	442311	414588	464580	7,2%	5,0%	12,1%
34	677511,6	78,1%	275	310	156,5%	62,1%	684121	667477	712918	5,2%	4,2%	6,8%
35	459167,4	137,4%	754	206	153,8%	41,1%	465022	452916	492083	7,2%	5,8%	8,6%
36	668615,4	142,4%	220	332	170,2%	66,5%	675720	657217	705167	5,5%	4,4%	7,3%

Table 31: The results of *C1-Self Adaptive VND* combination

C2 - Self Adaptive VND												
Data	Ave. Profit		Ave Iter.	Ave. # of Customers			Best-Worst Results			Gaps		
	Ave. Profit (a)	Imp.	# Iter.	# V. Pairs	Imp	Cover %	Best Profit (b)	Worst Profit (c)	Best Known (d)	Gap (d-a)	Gap (d-b)	Gap (d-c)
1	30315,5	41,7%	3219689	8	33,3%	80,0%	30315,5	30315,5	30315,5	0,0%	0,0%	0,0%
2	38765,6	19,6%	1377087	10	11,1%	100,0%	38765,6	38765,6	39340,3	1,5%	1,5%	1,5%
3	52565,1	20,8%	3178948	6	20,0%	60,0%	52565,1	52565,1	52855,2	0,6%	0,6%	0,6%
4	58365,6	4,6%	1481428	10	0,0%	100,0%	58365,6	58365,6	58365,6	0,0%	0,0%	0,0%
5	36469,5	35,7%	3412441	8	100,0%	80,0%	36469,5	36469,5	36469,5	0,0%	0,0%	0,0%
6	43368,6	3,4%	1769544	10	0,0%	100,0%	43368,6	43368,6	43764,3	0,9%	0,9%	0,9%
7	22306,0	24,2%	1091617	7	16,7%	28,0%	22306	22306	22441	0,6%	0,6%	0,6%
8	57996,7	34,8%	429681	18	28,6%	72,0%	58050,8	57915,5	58050,8	0,1%	0,0%	0,2%
9	57651,7	48,6%	931779	6	50,0%	24,0%	57651,7	57651,7	57651,7	0,0%	0,0%	0,0%
10	128334,8	104,6%	450153	17	50,9%	66,4%	132648	125021	132648	3,4%	0,0%	6,1%
11	33340,4	35,4%	1222698	7	16,7%	28,0%	33340,4	33340,4	33340,4	0,0%	0,0%	0,0%
12	79350,3	53,4%	435287	18	28,6%	72,0%	79350,3	79350,3	79350,3	0,0%	0,0%	0,0%
13	42174,0	46,3%	228968	19	35,7%	38,0%	42215,5	42215,5	44402,3	5,3%	5,2%	5,2%
14	80500,2	45,5%	120288	34	32,3%	68,8%	81848,1	79447,7	81999,4	1,9%	0,2%	3,2%
15	71082,8	52,7%	204745	14	13,3%	27,2%	71399,2	70774,1	71512,4	0,6%	0,2%	1,0%
16	122183,2	53,6%	109090	30	20,0%	60,0%	124910	120524	125032	2,3%	0,1%	3,7%
17	79651,2	13,4%	217368	15	16,9%	30,4%	80878,8	79018,5	80878,8	1,5%	0,0%	2,4%
18	130366,6	42,2%	121119	32	26,4%	63,2%	131339	129766	131845	1,1%	0,4%	1,6%
19	72287,9	76,7%	28570	48	50,6%	38,6%	73544,7	71821,1	75309,1	4,2%	2,4%	4,9%
20	133404,0	96,6%	9637	89	61,1%	70,9%	135622	132202	136594	2,4%	0,7%	3,3%
21	109355,8	152,9%	29079	42	81,7%	33,4%	110062	107844	113886	4,1%	3,5%	5,6%
22	194549,0	106,7%	10459	80	40,4%	64,0%	196524	193272	200402	3,0%	2,0%	3,7%
23	159129,6	54,7%	28174	44	40,6%	34,9%	162115	156253	164070	3,1%	1,2%	5,0%
24	267291,4	85,8%	11201	82	52,6%	65,9%	269790	265574	274563	2,7%	1,8%	3,4%
25	159526,6	103,0%	5423	100	69,8%	40,1%	161375	156864	167080	4,7%	3,5%	6,5%
26	245183,6	80,7%	2036	155	56,6%	62,0%	249312	241181	259319	5,8%	4,0%	7,5%
27	198740,0	135,2%	5516	82	58,5%	33,0%	201590	197311	206788	4,0%	2,6%	4,8%
28	317610,8	87,8%	2244	137	33,0%	54,8%	321155	312552	331291	4,3%	3,2%	6,0%
29	273095,0	134,3%	5436	96	68,4%	38,4%	276392	271370	281711	3,2%	1,9%	3,8%
30	404517,0	84,7%	2161	147	44,3%	58,9%	409262	400507	417282	3,2%	2,0%	4,2%
31	153820,6	131,8%	723	225	68,9%	44,9%	156106	151865	173826	**13,0%**	**11,4%**	**14,5%**
32	222072,8	96,8%	179	341	50,8%	68,2%	226913	217257	247385	**11,4%**	9,0%	**13,9%**
33	429853,0	182,2%	750	193	64,8%	38,6%	442218	412378	464580	8,1%	5,1%	**12,7%**
34	669212,8	110,3%	219	316	42,3%	63,2%	683211	657070	712918	6,5%	4,3%	8,5%
35	462999,2	134,9%	718	212	69,8%	42,4%	469162	453274	492083	6,3%	4,9%	8,6%
36	667184,8	84,2%	221	327	46,2%	65,5%	684217	655252	705167	5,7%	3,1%	7,6%

Table 32: The results of *C2- Self Adaptive VND* combination

Data	Average Profits and Average Number of Customer Pairs of 5 Runs								Best Found Profits of 5 Runs			
	C1-Seq VND		C2-Seq VND		C1-SaVND		C2-SaVND		C1-Seq VND	C2-Seq VND	C1-SaVND	C2-SaVND
	Profit	Pairs	Profit	Pairs	Profit	Pairs	Profit	Pairs	Profit			
1	**30315,5**	8	**30315,5**	8	**30315,5**	8	**30315,5**	8	30315,5	30315,5	30315,5	30315,5
2	**38765,6**	10	**38765,6**	10	37866,3	10	**38765,6**	10	38765,6	38765,6	38765,6	38765,6
3	51473,4	6	52128,4	6	**52565,1**	6	**52565,1**	6	51473,4	52565,1	52565,1	52565,1
4	**58365,6**	10	**58365,6**	10	**58365,6**	10	**58365,6**	10	58365,6	58365,6	58365,6	58365,6
5	**36469,5**	8	**36469,5**	8	**36469,5**	8	**36469,5**	8	36469,5	36469,5	36469,5	36469,5
6	**43368,6**	10	**43368,6**	10	**43368,6**	10	**43368,6**	10	43368,6	43368,6	43368,6	43368,6
7	**22306,0**	7	**22306,0**	7	**22306,0**	7	**22306,0**	7	22306	22306	22306	22306
8	**58050,8**	18	**58050,8**	18	57303,4	18	57996,7	18	58050,8	58050,8	58050,8	58050,8
9	**57651,7**	6	**57651,7**	6	**57651,7**	6	**57651,7**	6	57651,7	57651,7	57651,7	57651,7
10	**132648,0**	16	**132648,0**	16	128353,0	16	128334,8	17	132648	132648	132648	132648
11	**33340,4**	7	**33340,4**	7	**33340,4**	7	**33340,4**	7	33340,4	33340,4	33340,4	33340,4
12	**79350,3**	18	**79350,3**	18	77655,5	18	**79350,3**	18	79350,3	79350,3	79350,3	79350,3
13	41464,8	19	41955,2	19	41523,5	19	**42174,0**	19	42174,4	42174,4	42215,5	42215,5
14	81271,2	35	**81971,0**	35	80080,4	34	80500,2	34	81999,4	81999,4	81621	81848,1
15	70387,4	14	70398,5	13	**71113,8**	14	71082,8	14	70836	71326,4	71399,2	71399,2
16	**124288,0**	29	123931,0	29	122691,8	30	122183,2	30	124770	124770	123766	124910
17	76885,4	17	78418,8	16	76954,4	16	**79651,2**	15	80878,8	78818	80878,8	80878,8
18	**131052,6**	32	130670,4	32	130403,4	32	130366,6	32	131473	131845	131339	131339
19	**73259,9**	49	72445,3	48	72703,0	48	72287,9	48	73610,9	73575,7	73835,5	73544,7
20	132996,0	88	132438,0	88	132394,4	88	**133404,0**	89	134889	134186	132870	135622
21	111334,4	40	**111817,8**	41	109449,4	42	109355,8	42	112427	112480	112428	110062
22	**197899,8**	79	197214,2	79	194131,2	83	194549,0	80	200402	198611	195135	196524
23	162410,8	44	**162448,2**	44	158164,4	44	159129,6	44	162960	164070	162115	162115
24	**269072,8**	80	268713,8	82	266559,0	83	267291,4	82	271185	271193	272957	269790
25	158425,4	100	159146,2	100	157399,8	99	**159526,6**	100	161063	161040	159064	161375
26	243210,0	154	244591,6	155	**245600,8**	155	245183,6	155	249171	249661	247986	249312
27	201195,4	77	**202102,2**	79	197227,6	87	198740,0	82	203832	205480	200764	201590
28	319239,2	135	**321181,6**	136	315854,6	134	317610,8	137	321471	325511	320872	321155
29	271154,4	93	272715,2	93	271264,2	96	**273095,0**	96	279105	278233	274419	276392
30	402557,2	145	**406689,6**	147	399990,8	145	404517,0	147	407689	409755	407849	409262
31	150920,4	222	151943,2	223	151990,0	223	**153820,6**	225	154196	154932	154357	156106
32	218211,0	337	218213,0	336	**223999,8**	342	222072,8	341	221958	221322	224999	226913
33	439681,8	188	**441120,2**	189	433353,8	187	429853,0	193	450349	450337	442311	442218
34	**678396,2**	299	651754,6	309	677511,6	310	669212,8	316	693407	672910	684121	683211
35	455054,6	204	461394,0	206	459167,4	206	**462999,2**	212	463625	472626	465022	469162
36	**668655,0**	322	662605,4	320	668615,4	332	667184,8	327	673951	669926	675720	684217

Table 33: The comparison of the *GVNS* variations

Table 33 compares the average profits, the average number of the customers served and best profits of 5 runs for four combinations of the *GVNS*. Bold black numbers are the best average profits and bold red numbers are the best results among 4 combinations.

5.3 Interpretation of the Results

On the basis of the detailed result tables, certain concrete facts can be deducted. These deductions may provide the reader with a good perspective for grasping the pros and cons of the algorithm. For this reason, the pinpointed facts are specified below:

- *Two-Stage Cheapest Insertion Construction Heuristic (C2)* outperforms the *Greedy Construction Heuristic* for all instances except the instances with proportional revenues. The reason is that *C2* obtains the customers with respect to a ratio. The higher the revenues are, the more possibility of ratios' being higher is. So the heuristic most likely takes the customers with higher revenues and yet with higher demands. In many cases, the customers with higher demands cannot be inserted due to the vehicle capacity. As a result, the routes are left idle. (Table 26)
- In few cases, all customers are served. (Table 26)
- The differences of profits obtained by *C2* and *C1* are higher for the instances with fixed revenues. Since the revenues are fixed, *C2* inserts the nearest customer pairs. Thus, total travel time remains small. This enhances the possibility of serving more customers and increases the final profits. (Table 26)
- As the size of the data instances increases, the difference between the best known and average results grows remarkably for the instances with fixed revenues. For four variations, the differences are above 10%. This implies that running the instances with fixed revenues for many hours yields better results and increases the differences. [4] (Table 28, 29, 31, 32)
- The number of the iterations for the *Sa-VND* is significantly higher than the number of iterations of the *Seq-VND*. Since, the *Sa-VND* prioritises the neighbourhoods with less *CPU* time and the algorithm mostly executes these

[4] Note that best known results are mostly found by running the algorithm for several hours.

cheap neighbourhoods. Hence, this reduces total *CPU* time for one shaking and one *VND* step and increases the number of iterations in total. (Table 28, 29, 31, 32)

- As the number of customers decreases, the number of the iterations boosts. Therefore, more shaking steps are achieved and the possibility of finding better results increases. (Table 28, 29, 31, 32)
- The percentage of the visited customers varies between 24%-100% for all data instances and for four variations. (Table 28, 29, 31, 32)
- Starting with *C2* is advantageous in most cases in terms of average profits, particularly for large instances. Nevertheless, there are not too many enormous gaps in between which are reflected to the final solutions. (Table 33)
- If the algorithm is launched with *C1*, the *Seq-VND* outperforms the *Sa-VND* in most cases based on the average results but if it is started with *C2*, there is no prominent gap between them. (Table 33)
- All variations are prone to find best known solutions of the instances with 20 and 50 customers in 5 runs. For 3/6 of the instances with 20 customers and 5/6 of the instances with 50 customers, the best known solutions are achieved for all variations. (Table 33)
- There are some cases in which better results are obtained with less customers. This implies that serving rather profitable customers may increase the final profits. Therefore, the number of customers which are served is important as well in real-world applications. Since, the companies can lose the customers which might be regular customers afterwards. (Table 33)

Conclusion

This research delves into the *Multi-Vehicle Profitable Pickup and Delivery Problem (MVPPDP)* which is a combination of the *Vehicle Routing Problem (VRP)* and the *Knapsack Problem (KP)*. Moreover, a *General Variable Neighbourhood Search* algorithm is developed in order to seek for approximate solutions for the *MVPPDP*.

The developed *GVNS* algorithm has mainly three major components: Construction, shaking and the Local Search (VND). As a starting point, a *Greedy Construction Heuristic (C1)* and a *Two-Stage Cheapest Insertion Heuristic (C2)* are created for the solution construction section. Shaking part is formed by three sub-shaking parts one of which is chosen randomly in each iteration. The local search which is vital for intensification purposes over the solution space is constituted by 11 neighbourhoods. Since the order of the neighbourhoods affects the quality of the final solution, two alternative *VND* heuristics are built up. One of them is the *Sequential-VND* and the latter is the *Self-Adaptive VND*. The orders of the neighbourhoods are determined through the experiment for the *Sequential-VND* while the *Self-Adaptive VND* determines its own order by adapting itself to the input data.

The quality of the algorithm is tested on newly created data instances with randomly generated 20, 50, 100, 250, 500 and 1000 customers and the revenues are diversified as to be fixed, proportional to the demands and random. The construction heuristics and four versions of the algorithm are tested on the instances. As a result, *Two-Stage Cheapest Insertion Heuristic* outperforms the *Greedy Construction Heuristic* except for the instances with proportional revenues. Moreover, it is experienced that there is no significant difference between the final results of the variations of the algorithm. Nevertheless, certain behaviours of these variations are detected. The most noticeable one of them is that the *Self-Adaptive VND* iterates more but it cannot reflect this to the quality of the final solutions.

All in all, the best components are chosen to build the final algorithm among several trials. Thus, the final algorithm improves the initial solutions significantly but the strength and efficiency of the algorithm remain ambiguous unless it is compared with another algorithm.

References

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M. (2003). *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Berlin, Heidelberg, Germany. Springer. 4-29.

Berbeglia, G., Cordeau, J. F., Gribkovskaia, I., Laporte, G. (2007). Static pickup and delivery problems: A classification scheme and survey. *TOP, 15,* 1-31.

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

Bodin, L. D., Golden, B. (1981). Classification in vehicle routing and scheduling. *Networks, 11* (2), 97-108.

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

Chbichib, A., Mellouli, R., Chabchoub, H. (2012). Profitable vehicle routing problem with multiple trips: Modeling and variable neighborhood descent algorithm. *American Journal of Operational Research, 2* (6), 104-119.

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

Dantzig, G. B. (1957). Discrete variable extremum problems. *Operations Research, 5,* 266-277.

Dantzig, G. B., Ramser, J. H. (1959). The truck dispatching problem. *Management Science, 6* (1), 80-91.

Feillet, D., Dejax, P., Gendreau, M. (2005). Traveling salesman problems with profits. *Transportation Science, 36,* 188-205.

Hansen, P., Mladenović, N., Moreno-Pérez, J. A. (2008). Variable neighborhood search: Methods and applications. *4OR, 6,* 319-360.

Hu, B., Raidl, G. R. (2006). Variable neighborhood descent with self-adaptive neighborhood-ordering. In C. Cotta, A. J. Fernandez, J. E. Gallardo (Eds). *Proceedings of the 7th EU/MEeting on Adaptive, Self-Adaptive, and Multi-Level Metaheuristics.*

Ilić, A., Urošević, D., Brimberg, J., Mladenović, N. (2010). A general variable neighborhood search for solving the uncapacitated single allocation p-hub median problem. *European Journal of Operational Research, 206* (2), 289-300.

Lenstra, J., Rinnooy Kan, A. H. G. (1981). Complexity of vehicle routing and scheduling problems. *Networks, 11,* 221-227.

Mingozzi, A., Giorgi, S., Baldacci, R. (1999). An exact method for the vehicle routing problem with backhauls. *Transportation Science, 33,* 315-329.

Mladenović, N., Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24,* 1097-1100.

Mladenović, N., Urošević, D., Hanafi, S. (2013). Variable neighborhood search for the travelling deliveryman problem. *4OR: A Quarterly Journal of Operations Research, 11* (1), 57-73.

Mladenović, N., Urošević, D., Hanafi, S., Ilić, A. (2012). A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. *European Journal of Operational Research, 220* (1), 270-285.

Parragh, S. N., Doerner, K., Hartl, R. (2008). A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft, 58* (2), 81-117.

Pisinger, D. (1995). *Algorithms for knapsack problems* (Ph.D. dissertation). DIKU, Dept. Comput. Sci., University of Copenhagen, Copenhagen, Denmark.

Ropke, S., Cordeau, J. F. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science, 43* (3), 267-286.

Savelsbergh, M. W. P., Sol, M. (1995). The general pickup and delivery problem. *Transport Science, 29,* 17-29.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research, 35,* 254-265.

Ting, C. K., Liao, X. L. (2013). The selective pickup and delivery problem: Formulation and a memetic algorithm. *International Journal of Production Economics, 141* (1), 199-211.

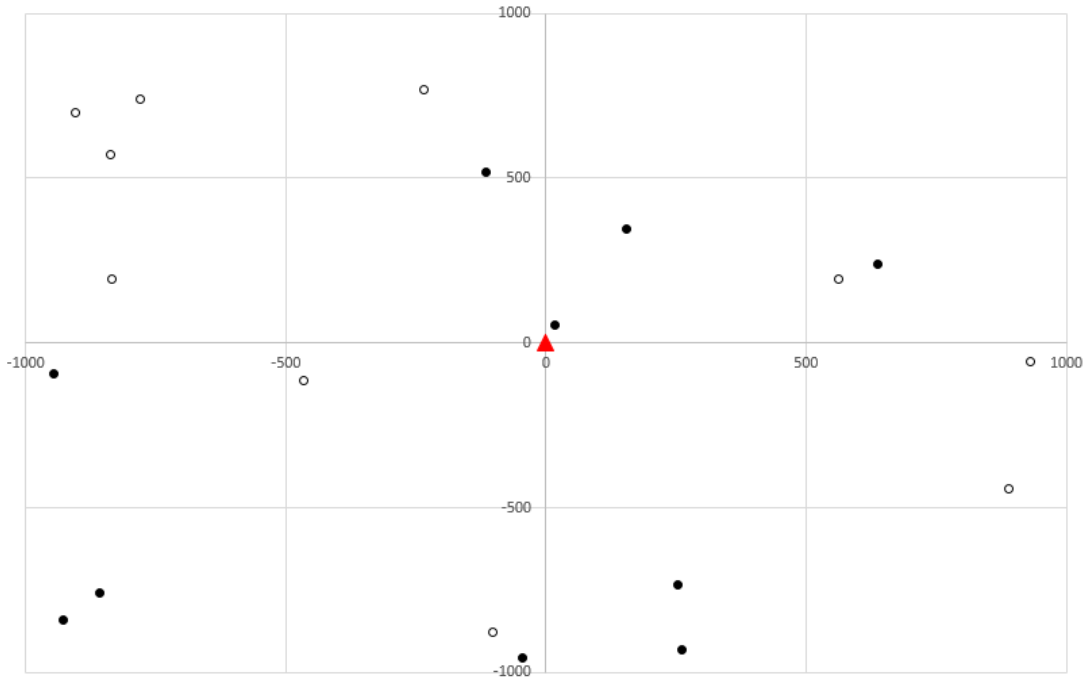
Toth, P., Vigo, D. (Eds.). (2002). *The vehicle routing problem*. Philadelphia, USA. Society for Industrial & Applied Mathematics, SIAM. 2-10, 109-128.

Vansteenwegen, P., Souffriau, W., Oudheusden, D. V. (2010). The orienteering problem: A survey. *European Journal of Operational Research, 209* (1), 1-10.

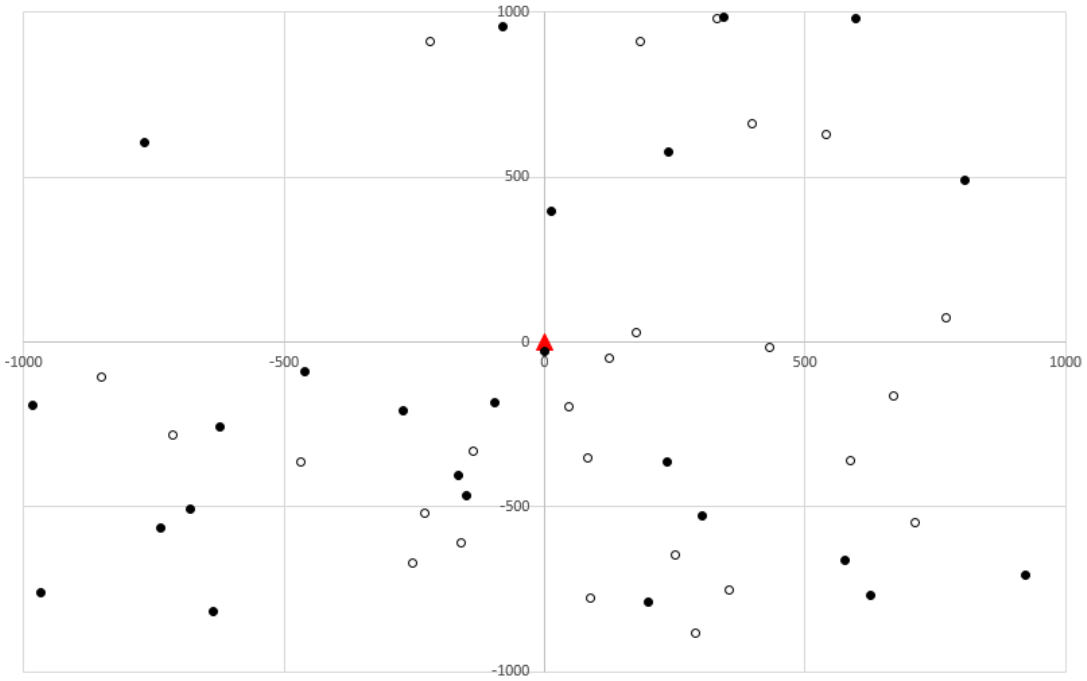
Appendix

1. The distribution of customers (data with 20 customers)

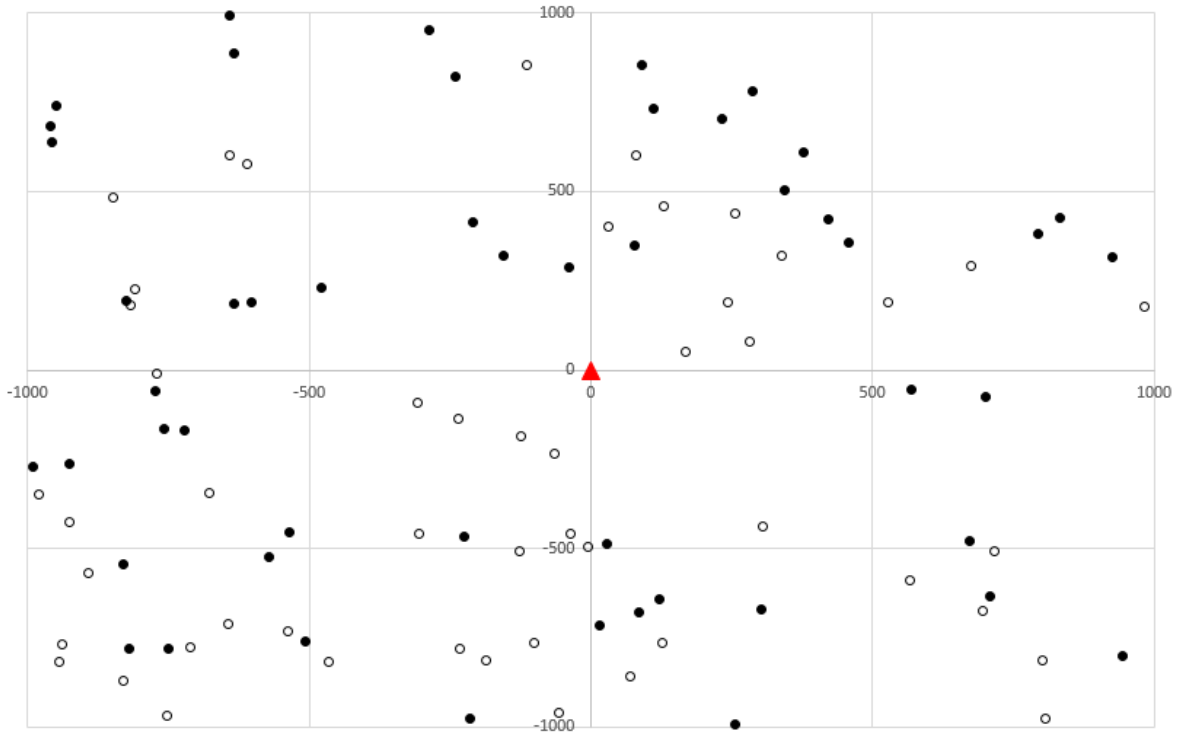
Note: *Filled circles are the pickup customers, empty circles are the delivery customers and the red triangle is the depot. The coordinate range is [-1000, 1000].*



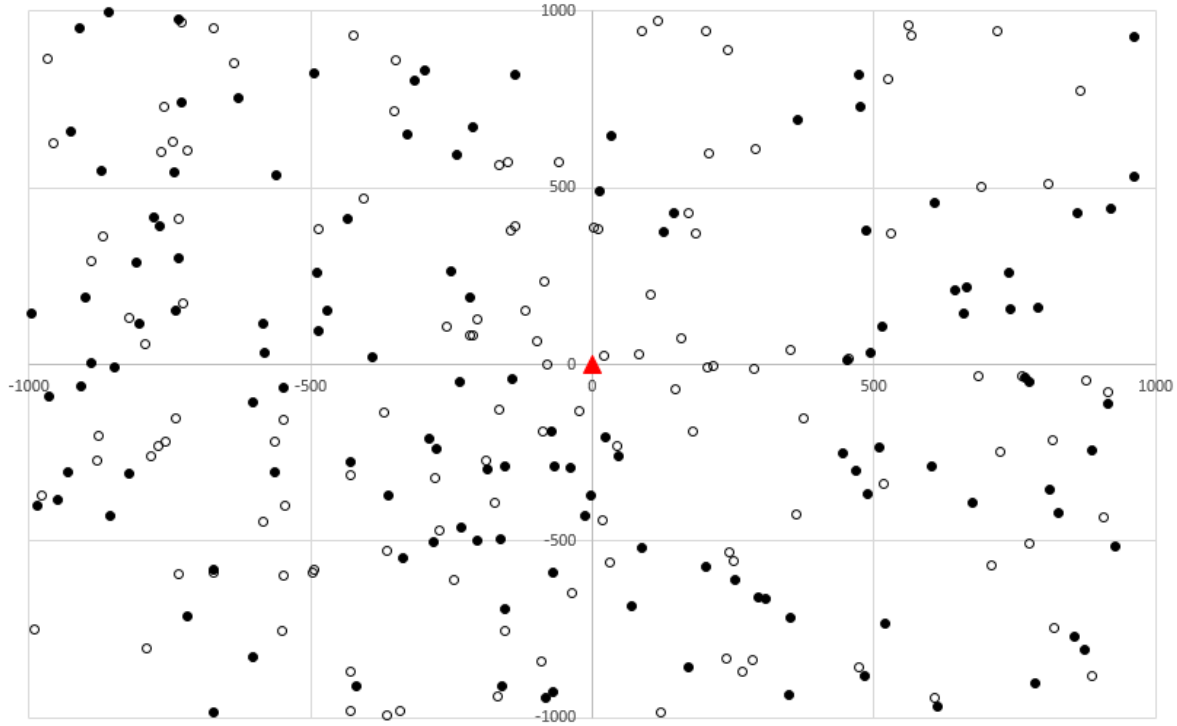
2. The distribution of customers (data with 50 customers)



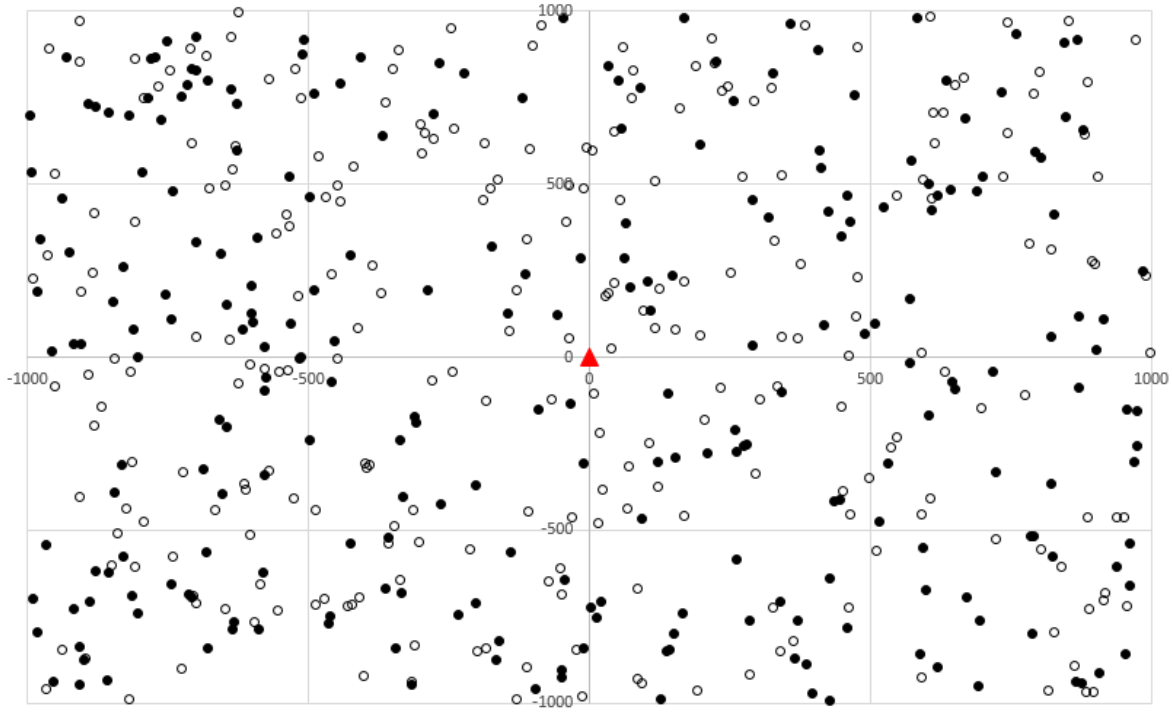
3. The distribution of customers (data with 100 customers)



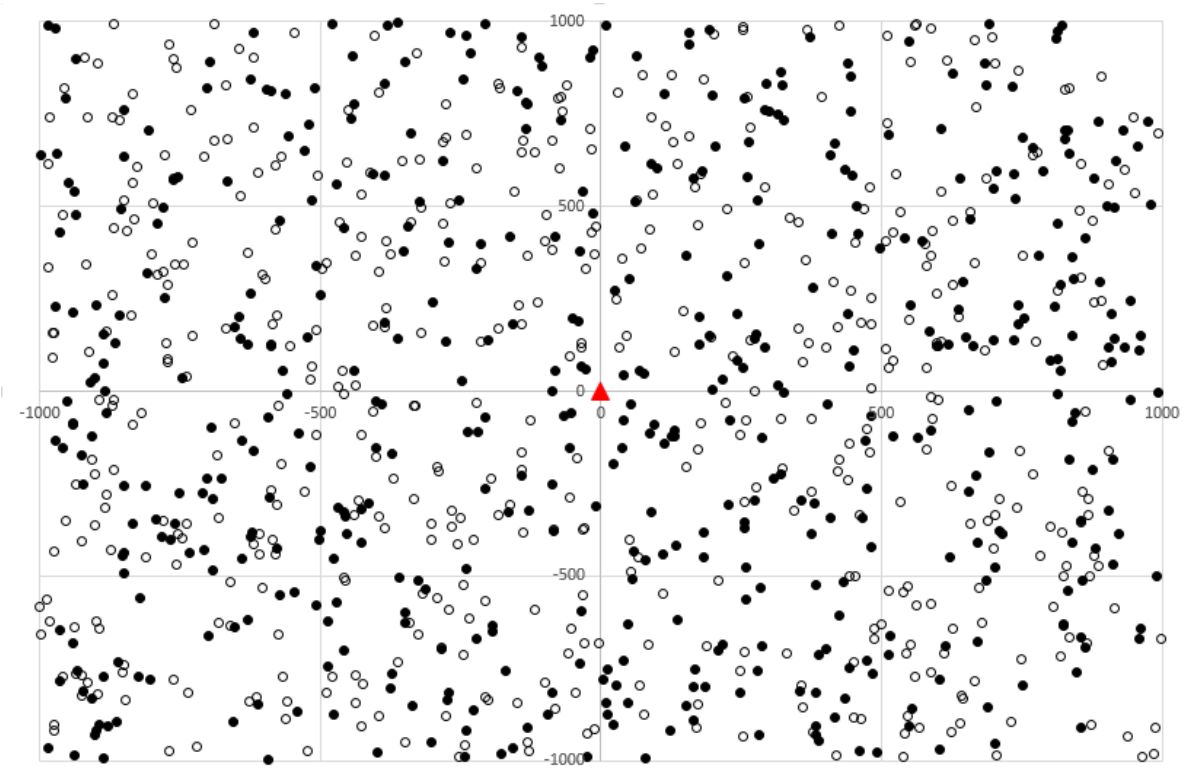
4. The distribution of customers (data with 250 customers)



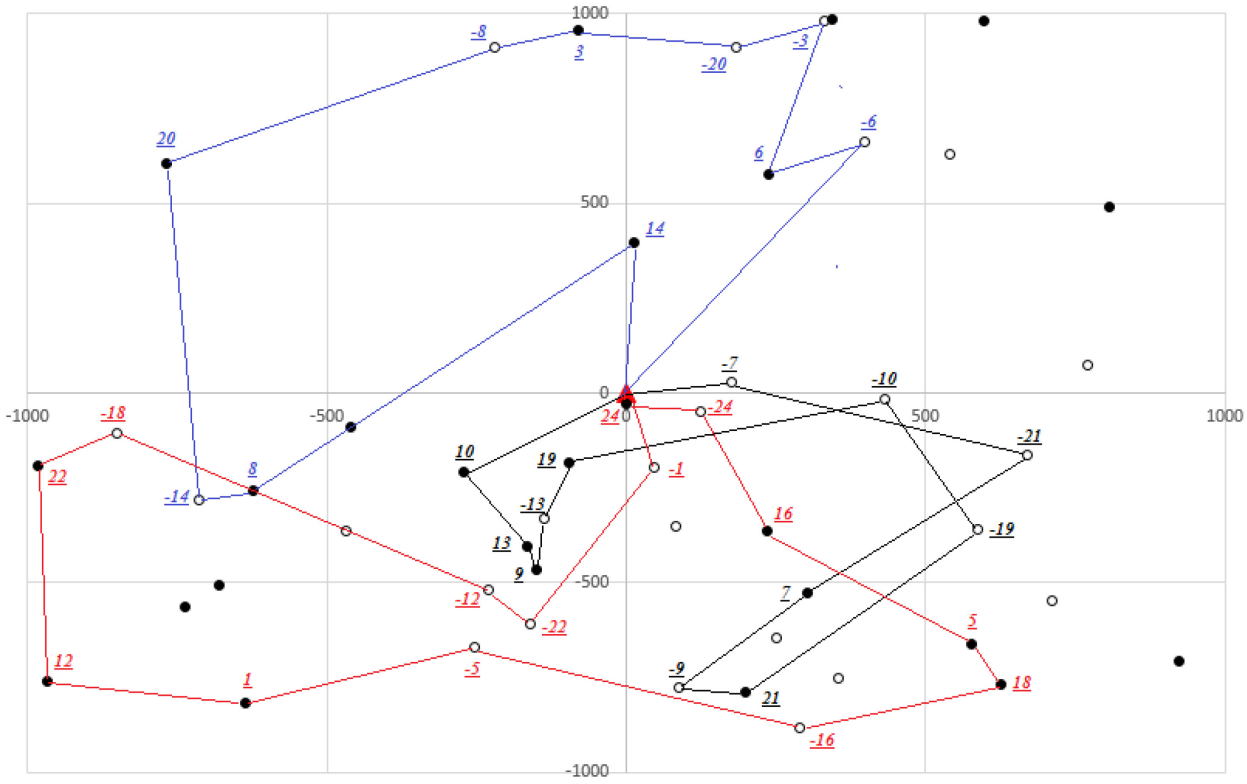
5. The distribution of customers (data with 500 customers)



6. The distribution of customers (data with 1000 customers)



7. An example of a best known solution (data 8-MR-F-L-50)



Black Route: 0, 10, 13, 9, -13, 19, -10, -19, 21, -9, 7, -21, -7, 0

Blue Route: 0, 14, 8, -14, 20, -8, 3, -20, -3, 6, -6, 0

Red Route: 0, 24, -24, 16, 5, 18, -16, -5, 1, 12, 22, -18, -12, -22, -1, 0

Total distance of the black route:	*4088, 53*
Total profit of the black route:	*19911, 5*
Total distance of the blue route:	*4872, 06*
Total profit of the blue route:	*15127, 9*
Total distance of the red route:	*4988, 58*
Total profit of the red route:	*23011, 4*

Total Distance:	*13949, 2*
Total Revenue:	*72000*
Total Profit:	***58050, 8***

Note: Since the revenues are fixed, nearest customers are collected.

8. Curriculum Vitae

Personal Data

Name:	Murat Küçüktepe
Date of birth	January 14, 1987
Citizenship	Turkey

Education

Since 03/2012	University of Vienna, Austria
	Master's program: Quantitative Economics, Management and Finance
	Specialization: Operations Research
	Thesis: A General Variable Neighbourhood Search Algorithm for the Multi-Vehicle Profitable Pickup and Delivery Problem
09/2011 – 12/2011	University of Wisconsin, Madison, Wisconsin, USA
	Certificate: English Language Course (ESL)
04/2011 – 07/2011	Marmara University, Istanbul, Turkey
	Certificate: English Language Course
09/2005 – 09/2009	University of Istanbul, Istanbul, Turkey
	Bachelor's program: Industrial Engineering
	Thesis: Six Sigma Applications
09/2001 – 06/2005	Erzincan Anatolian High School, Erzincan, Turkey

Work Experience

01/2013 – 08/2014	Supervisor for Anatolian University Open (Remote) Education Faculty, Vienna, Austria
07/2010 – 03/2011	Infantry third lieutenant at Turkish Army Land Forces 5th Commando Infantry Command, Bozcaada and Isparta, Turkey
07/2009 – 01/2010	Quality engineer at Ozler Plastic Industry, Istanbul, Turkey
05/2008 – 01/2010	Province soccer referee at Soccer Federation of Turkey, Istanbul, Turkey
07/2008 – 09/2008	Quality controller at Hershey's Chocolate, Palmyra, Pennsylvania, USA

Additional Skills

Turkish	Native
English	Fluent
German	Basic
IT Skills	C++, Xpress IVE, SAP, CSS, HTML, PHP
Driving Licence	B