BRUNO PETRATO BRUCK

CONTRIBUTIONS TO THE SINGLE AND MULTIPLE VEHICLE ROUTING PROBLEMS WITH DELIVERIES AND SELECTIVE PICKUPS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS – BRASIL
2012

Ficha catalográfica preparada pela Seção de Catalogação e Classificação da Biblioteca Central da UFV

T
B888c
2012

Bruck, Bruno Petrato, 1988-
Contributions to the single and multiple vehicle routing problems with deliveries and selective pickups / Bruno Petrato Bruck. – Viçosa, MG, 2012.
viii, 78f. : il. (algumas col.) ; 29cm.

Orientador: André Gustavo dos Santos.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f. 75-78

1. Pesquisa operacional. 2. Otimização combinatória. 3. Logística. 4. Otimização matemática. I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. II. Título.

CDD 22. ed. 003

BRUNO PETRATO BRUCK

CONTRIBUTIONS TO THE SINGLE AND MULTIPLE VEHICLE ROUTING PROBLEMS WITH DELIVERIES AND SELECTIVE PICKUPS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*

APROVADA: 26 de Outubro de 2012.

Prof. Geraldo Robson Mateus
(DCC - UFMG)

Prof. Haroldo Gambini Santos
(DECOM - UFOP)

Prof. José Elias Claudio Arroyo
(Coorientador)

Prof. André Gustavo dos Santos
(Orientador)

AKNOWLEDGEMENTS

À minha família, principalmente à minha mãe por todo o apoio e suporte.

Aos meus amigos pela boa amizade, pelo apoio e pelas horas de lazer.

Ao meu orientador André por todo o apoio, amizade, dedicação e por às vezes ficar até de madrugada online fazendo revisões comigo nos artigos para não perder o deadline.

Ao professor Arroyo pela amizade e por ajudar com o projeto para a bolsa de mestrado, sem a qual ficaria complicado continuar o mestrado.

À professora Luciana e ao Anand que acabaram por ajudar na decisão de qual problema abordar no mestrado.

À Deus acima de tudo por todas as oportunidades.

...

May the force be with us

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AC	Additional constraints
ANOVA	ANlysis Of VAriance
AS	Anti-symmetry constraints
CC	Constraint Disaggregation and Coefficient Improvement
CI	Cover Inequalities
CVRP	Capacitade Vehicle Routing Problem
EA	Evolutionary Algorithm
GA	Genetic Algorithm
GVNS	General Variable Neighborhood Search
ILP	Integer Linear Programming
ILS	Iterated Local Search
KC	Knapsack Constraints
LB	Lower Bound
LC	Lifted capacity constraints
LI	Logical Inequalities
LS	Lifted MTZ Subtour Elimination Constraints
MACS	Multi-ant Colony System
MILP	Mixed Integer Linear Programming
MTZ	Miller-Tucker-Zemlin Constraints
MVRPDSP	Multiple Vehicle Routing Problem with Deliveries and Selective Pickups
RCL	Restricted Candidate List
SVRPDSP	Single Vehicle Routing Problem with Deliveries and Selective Pickups
TS	Tabu Search
TSP	Traveling Sallesman Problem
VND	Variable Neighborhood Descent Algorithm
VRPB	Vehicle Routing Problems with Backhauls
VRPDP	Vehicle Routing Problems with Deliveries and Pickups
VRPSPD	Vehicle Routing Problem with Simultaneous Pick-up and Delivery services

ABSTRACT

BRUCK, Bruno Petrato, M.Sc., Universidade Federal de Viçosa, October 2012. **Contributions to the Single and Multiple Vehicle Routing Problems with Deliveries and Selective Pickups**. Adviser: André Gustavo dos Santos. Co-adviser: José Elias Claudio Arroyo.

The Single Vehicle Routing Problem with Deliveries and Selective Pickups (SVRPDSP) is a variation of the classical Vehicle Routing Problem (VRP) that has received limited attention. It has many practical applications in reverse logistic contexts, such as in drink factories, which besides having to supply stores and supermarkets with full bottles, have to pickup empty bottles, returning them to the factory in order to be clean and refilled. There is also the Multiple Vehicle Routing Problem with Deliveries and Selective Pickups (MVRPDSP), which shares the same applications of the SVRPDSP. It is even more practical, since in real world cases it is commom having multiple vehicles. However, regarding the MVRPDSP, to our knowledge, there is not a single approach in the literature. In the present work, for the SVRPDSP, in terms of heuristic approaches, we propose a Hybrid Evolutionary Algorithm (EA) which makes use of a data mining strategy in its crossover and mutation phases; and a Variable Neighborhood Descent Algorithm (VND). In addition we also propose a Branch&Cut algorithm for an exact formulation of the literature and a novel formulation. Regarding the MVRPDSP we propose two formulations based on the ones of the single vehicle version of this problem and a hybrid cluster-first constructive heuristic. Experimental results show that the proposed formulation for the SVRPDSP outperforms by far the others from the literature, finding optimal solutions for more than half the instances of the benchmark used in the literature. For the MVRPDSP we created a benchmark of instances and report several good solutions, including some optimals.

RESUMO

BRUCK, Bruno Petrato, M.Sc., Universidade Federal de Viçosa, Outubro de 2012. **Contribuições para o Single e Multiple Vehicle Routing Problems with Deliveries and Selective Pickups**. Orientador: André Gustavo dos Santos. Coorientador: José Elias Claudio Arroyo.

O *Single Vehicle Routing Problem with Deliveries and Selective Pickups* (SVRPDSP) é uma variação do clássico *Vehicle Routing Problem* (VRP). Tem recebido pouca atenção, apesar de possuir muitas aplicações práticas em cenários de logística reversa, como por exemplo em fábricas de bebidas, que ao mesmo tempo em que há uma demanda de supermercados e outras lojas por garrafas cheias, também existe uma demanda pela coleta de garrafas vazias a retornar para o depósito a fim de serem limpas e reutilizadas. Além disso também existe o *Multiple Vehicle Routing Problem with Deliveries and Selective Pickups* (MVRPDSP), o qual compartilha as mesmas aplicações, podendo até ser considerado mais prático do que o SVRPDSP, já que em casos reais são usuais cenários com multiplos veículos. Entretanto, com relação ao MVRPDSP não é de nosso conhecimento qualquer abordagem na literatura. Neste trabalho, para o SVRPDSP, em termos de abordagens heurísticas, são propostos um Algoritmo Evolucionário Híbrido que faz uso de uma estratégia de *data mining* em seus operadores de crossover e mutação, além de um *Variable Neighborhood Descent Algorithm (VND)*. Além disso, também é proposto um *Branch&Cut* para uma formulação matemática da literatura e uma nova formulação, a qual utiliza um tipo diferente de restrições para eliminação de subciclos. Com relação ao MVRPDSP, são propostas duas formulações matemáticas baseadas nos modelos matemáticos do SVRPDSP, e uma heurística construtiva híbrida do tipo *cluster-first*. Resultados experimentais indicam que a formulação proposta para o SVRPDSP possui um desempenho muito superior às da literatura, conseguindo encontrar a solução ótima para mais da metade das instâncias. Para o MVRDPSP foram criadas instâncias de teste e são reportados vários bons resultados, incluindo algumas soluções ótimas.

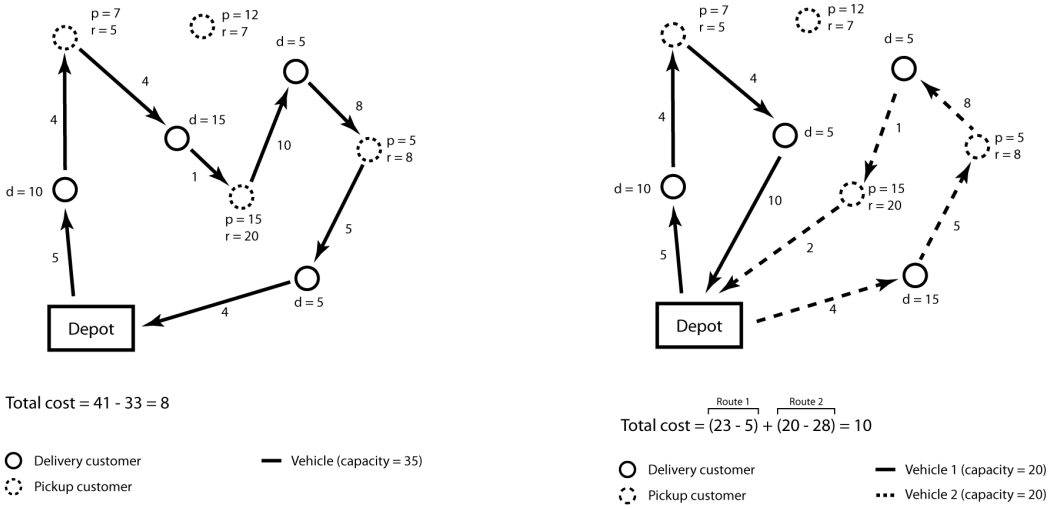
Chapter 1

Introduction

In the Single Vehicle Routing Problem with Deliveries and Selective Pickups (SVRPDSP) there are a set of customers to be served and a depot from where a vehicle departs to serve the customers. Each customer has a certain demand of goods either to be delivered or to be picked up, which generates a revenue if collected. It is possible for a customer to have both demands. In such case, if both are going to be served, they can be performed simultaneously or in two different visits, each completely fulfilling one of the demands. The vehicle that departs from the depot shall perform a route that visits a subset of customers performing deliveries and pickups, then return to the depot. All delivery demands must be fulfilled exactly once. The pickup demands, however, are not mandatory, therefore they are only performed if there is enough space in the vehicle and if they are profitable. Serving a pickup demand is profitable if the revenue generated by collecting it is greater than the additional routing cost. One can notice that some pickups might not be served at all and it is possible to argue that they would need to be performed at some point. To address this issue these pickup demands could either be delayed to be served in the following day, or a third party service can be used to collect these pickups, which could be a less costly option than forcing all pickups to be fulfilled or sending another vehicle only to perform a few pickups. The objective is to find a route that minimizes the total routing cost, which is the travel cost to visit the customers minus the revenue generated by the collected pickups. Figure 1.1a shows a small example with 8 customers and a vehicle with capacity equals to 35. In the figure, d is the delivery demand of a customer, while p stands for the pickup demand and r is the revenue generated by performing the respective pickup demand of the customer. The transportation cost of the solution presented is equal to $5+4+4+1+10+8+5+4 = 41$ and the total revenue generated by the three pickups collected is $5+20+8 = 33$. Therefore the total cost of this solutions is $41-33 = 8$. In

addition notice that one pickup was not performed, which does not make the solution unfeasible, since only the delivery demands are mandatory.

Similarly there is the Multiple Vehicle Routing Problem with Deliveries and Selective Pickups (MVRPDSP) which is a generalization of the SVRPDSP and only differs from it by adding the possibility of having multiple vehicles and, therefore, multiple routes. In addition we consider the fleet as homogeneous, what implies that all vehicles have the same capacity. Figure 1.1b depicts a small example with eight customers. In this example the route of vehicle 1 has a transportation cost of $5+4+4+10 = 23$ and it generates a revenue of 5 by performing one pickup demand, therefore the total cost of this route is $23 - 5 = 18$. The second route has a transportation cost of 20, but it generates a revenue of $8 + 20 = 28$, therefore the total cost of such route is -8, which means that this route ended up generating a profit greater than its cost. Considering both routes the total cost of this solution is $18+(-8) = 10$. Notice that route 2 cannot be done in the reverse order, since there is no space in the vehicle to collect the first pickup demand as it would be still full with the delivery goods.



(a) Example of the SVRPDSP (b) Example of the MVRPDSP

Figure 1.1: Examples for the SVRPDSP and the MVRPDSP

The SVRPDSP may be conveniently described using a complete directed graph $G = \{V, A\}$, where $V = \{0, 1, ..., n\}$ is a vertex set representing the depot (vertex 0) and the n customers and $A = \{(i, j) : i, j \in V, i \neq j\}$ is a set of arcs. Each customer i has a delivery demand $d_i \geq 0$, a pickup demand $p_i \geq 0$, and a revenue $r_i \geq 0$. Each arc $(i, j) \in A$ has a travel cost c_{ij}. The vehicle has a capacity Q, with $Q \geq \sum_{i \in V} d_i$. The problem is NP-hard, since the well known Traveling Salesman Problem (TSP) is a special case (when $p_i = 0$, $d_i \neq 0, \forall i \in V$).

In order to describe the MVRPDSP, instead of considering one vehicle, let K be the set of vehicles. We assume that all vehicles in K have the same capacity Q, therefore the fleet is homogeneous. This problem is clearly NP-hard since the SVRPDSP is a special case (when $|K| = 1$).

1.1 Motivation

Both, the SVRPDSP and the MVRPDSP, arise in several reverse logistics contexts. An example is drink factories, which besides having to supply stores and supermarkets with full bottles, have to pickup empty bottles, returning them to the factory in order to be clean and refilled [25]. This problem also arises in logistic companies, which besides having delivery demands of packages, also have a demand of packages to be picked up that will be transported to the depot before being delivered elsewhere.

Another context is refered by [4] and [5], in which as a consequence of the constant growth in global awareness to preserve the environment, companies that manufactures electronics and batteries are being responsible to also pickup their broken and used products that otherwise would go as normal trash and pollute. For instance in Brazil, there is a law, from 2010, that entrusts companies and sellers with reverse logistic tasks related to their products, avoiding unnecessary trash being discarded and polluting the environment [18].

Regardless their many practical applications and their similarity with other vehicle routing problems, the literature regarding these problems is rather scarce, with only a few papers approaching the single vehicle version and none studying the multiple vehicles one.

1.2 Contributions

In this work we propose some heuristic and exact approaches for both, the SVRPDSP and MVRPDSP. For the SVRPDSP we have implemented and tested an Hybrid Evolutionary Algorithm (EA) and a Variable Neighborhood Descent (VND). In terms of exact approaches we propose a Branch&Cut algorithm for a formulation of the literature and also a novel formulation which uses a different kind of subtour elimination constraints.

Since, to our knowledge, the MVRPDSP is a novel problem, we proposed two mathematical formulations, which are adaptations of formulations from the single vehicle version of this problem; and a hybrid constructive heuristic.

We report the results of our approaches for the SVRPDSP, including several new solutions and some new optimal solutions for the most used instances of the literature. In addition, we propose the MVRPDSP along with 72 benchmark instances, and present the solutions found by our methods.

1.3 Chapters

The present work is divided into six chapters, including this introduction. Chapter 2 provides a literature review of the problems, describing the heuristic and exact approaches already proposed, including three metaheuristics and two mathematical formulations for the SVRPDSP. Since the MVRPDSP is a novel problem, there is no literature regarding such problem.

We have divided the methodology chapter into two different chapters to better present the proposed approaches for each problem. Therefore, Chapter 3 presents the methodology used for the SVRPDSP, including the proposed heuristics and exact approaches. Similarly, Chapter 4 provides the methodology for the MVRPDSP describing the two mathematical formulations and the hybrid constructive heuristic proposed.

Chapter 5 presents the computational results of our approaches, comparing them with others from the literature. Finally, Chapter 6 provides the conclusions of the present work and proposes some further research topics.

Chapter 2

Literature review

Both, the SVRPDSP and the MVRPDSP belong to the same family of Vehicle Routing Problems with Deliveries and Pickups (VRPDP), also commonly referred by Vehicle Routing Problems with Backhauls (VRPB), where backhaul correspond to the pickup service. According to Parragh [24], this family of problems has four classes. In the first two, customers have either a delivery or a pickup demand while in the last two customers are allowed to have both demands.

The first class is often referreded as Vehicle Routing Problem with Backhauls (VRPB). In such problem customers have either a delivery demand (linehaul) or a pickup demand(backhaul). However all delivery demands must be performed before the pickup demands, thus mixed load are not allowed at any point. This constraint is important when the vehicles are rear-loaded and the rearrangement of the loads at the customers is not deemed feasible, as stated by Goetschalckx and Jacobs-Blecha in [13]. The multiple vehicle variant of this problem is the most studied in the literature, approached in several papers. Brandão in [3] proposed a Tabu Search (TS), and Ganesh and Narendran proposed a constructive heuristic and a Genetic Algorithm (GA) in [12]. In addition, in 2009, Gajpal and Abad [11] have presented a Multi-ant Colony System (MACS). Recently Zachariadis and Kiranoudis have developed an effective local search approach in [32]. Some authors have also approached the VRPB with time windows constraints. Among the approaches for this variant are an Insertion Based Ant System proposed by Reimann *et al.* in [26] and a heuristic based on Large Neighborhood Search, proposed by Ropke and Pisinger in [27].

In the other hand the second class allows mixed loads, while constrainting customer to have only one type of demand, either delivery or pickup. In the literature there are several approaches for both, the single and the multiple vehicle. Recently, for the single vehicle version, Dumitrescu *et al*, in [8], have proposed a Branch&Cut algo-

rithm. An Ant Colony System was proposed by Falcon *et al* [9]. The later considers only one commodity. For the multiple vehicle case we cite an integrated construction-improvement heuristic, by Nagy and Salhi[22].

The third class describes contexts where customer have both types of demand and can be visited twice, thus it is not mandatory to perform both demands simultaneously. Although all delivery and pickup demands must be performed until the end of the route, most of the approaches for this class in the literature allows demands to be partially performed at each visit. This class is not heavily studied in the literature, however all approaches for the second class are valid for this class if each demand is modeled as two different customers, one for the delivery demand and the other for the pickup demand, considering a transportation cost between them as zero.

Finally, in the fourth class, customers may have both demands but may be visited only once. This includes maybe one of the most studied problems the Vehicle Routing Problem with Simultaneous Pick-up and Delivery services (VRPSPD). In such problem every customer has both demands and the pickup services are not optional. In addition, a customer must be visited only once, therefore both demands must be satisfied in a single visit to the customer. This problem has been studied by a number of authors. Most approaches are heuristics including a Tabu Search (TS), proposed by Montané and Galvão [21], a hybrid metaheuristc proposed by Zachariadis [33]. Recently, Subramanian *et al* proposed a Parallel heuristic [29] using VND. There is also a variant of the VRPSPD, including time windows for the customers and consequently increasing the complexity of the problem. For such problem, Boubahri *et al* in [2] proposed an Ant Colony System and, recently Mingyong and Erbao [20] proposed a Differential Evolution Algorithm.

Notice that although our problems can be considered as VRPDP problems, none of these classes include them, since in all of them the pickups are mandatory. Therefore we suggest a new class, where customers are allowed to have only one demand or both, there are no order in which the services must be performed and the pickups are optional. In the suggested fifth class, that includes both problems approached in this work, there is also the Vehicle Routing Proble with Deliveries, Selective Pickups and Time Windows. This problem only differs from the MVRPDSP by adding the time windows for the customers, which constraints the time in which a customer can be visited by the vehicle, thus increasing the complexity of the problem. To our knowledge, the only approach in the literature for this problem is a Branch&Price algorithm, proposed by Gutiérrez-Jarpa, *et al* in [15].

In [17], Laporte and Gribkovskaia present the One-to-Many-to-One Single Vehicle Pickup and Delivery Problems, where a vehicle based at the depot must perform

pickups and deliveries at customers and then return to the depot. They cite some natural extensions which have received limited attention so far. One of them is the SVRPDSP, which considers the pickups optional, but generating a revenue. Although the many practical applications there are few papers approaching the SVRPDSP and, to our knowledge, none about MVRPDSP.

Suŕal and Bookbinder [30] are the first to directly address this problem. They present the problem using the notation $\alpha/\beta/\gamma$, where α denotes the number of vehicles (1 for single and M for multiple), β the pickup service options ($must$ or $free$ if the pickup is respectively mandatory or optional) and γ the precedence order for visiting the customers ($prec$ if all deliveries must precede the pickups, or any if they can be visited in any order). While the SVRPDSP is *1/free/any*, according to this notation, the MVRPDSP can be described as *M/free/any*. They cited papers dealing with the *1/free/prec* and *1/must/any* problems, and claimed to be the first to address the *1/free/any*. For the multiple vehicle versions, they list papers for the *M/must/prec* and *M/free/prec*, however no mention is made about the *M/free/any*, which is one of our objects of study. They propose a mixed integer linear programming formulation for the SVRPDSP along with some improvements on the constraints to strenghten the formulation, such as constraint disaggregation, coefficient improvement, cover and logical inequalities, and lifted subtour elimination constraints. They modified 24 instances from the literature, with sizes of 10, 20 and 30 customers, to test their formulation. These instances were adapted for the SVRPDSP by setting some of the delivery demands as pickups in 3 different ways (20% of the customers reset as pickups, then 30% and 40%), generating a total of 72 instances, which were tested with some combinations of the formulation and the improvements resulting in about 75% of the instances optimally solved in a reasonable computational time for the best combination.

Gribkovskaia, Laporte and Shyshou in [14] proposed another MILP formulation and a Tabu Search metaheuristic (TS) for the SVRPDSP to compare their efficiency with some classical constructive heuristics. They gathered 17 instances of the Capacitated Vehicle Routing Problem from the VRPLIB [31]. As these instances only have delivery demands they had to generate the pickup demands and profit values. The pickup demands were generated based on the values of the delivery demands and the profit values generated based on a formula that results in values proportionals to the average cost of an instance, multiplied by a factor ω. They considered 4 values for the parameter ω, therefore resulting in a total of 68 instances. Computational tests with classical heuristics and the TS yielded gap values ranging from very small to high values. However they only present average results. Regarding their formulation, it was only able to solve instances with number of customers up to 22.

Coelho [6], in his master's thesis, proposed a General Variable Neighborhood (GVNS) to solve the SVRPDSP. At first it solves the Travelling Salesman Problem (TSP) and the Knapsack problem for the instance and then uses their solutions to create an initial solution for a General Variable Neighborhood Search algorithm (GVNS). A theoretical lower bound is calculated in the same way as proposed by [14], using the values of the optimal solutions of the TSP and Knapsack formulations. The authors were able to significantly improve the solutions found by [14], including 3 optimal solutions as their values are equal to their respective lower bound.

Since we adapted the formulation of Sural and Bookbinder [30] to the MVRPDSP and proposed a Branch&Cut algorithm for the formulation of Gribkovskaia, Laporte and Shyshou [14], we describe them in details in the following section.

2.1 MILP Formulations

As previously stated, there are two mixed linear integer programming formulations for the SVRPDSP in the literature, while there is none for the multiple vehicle version.

Let us define the following notation, which will be used from now on to describe the input data of the problems:

- V_d: set of delivery customers
- V_p: set of pickup customers
- V: set of nodes ($V = V_d \cup V_p \cup 0$)
- K: set of vehicles ($|K| = 1$ for the SVRPDSP)
- d_i: delivery demand of customer i
- p_i: pickup demand of customer i
- r_i: revenue associated with the pickup of customer i
- c_{ij}: cost for travelling through arc (i, j)
- D: total delivery demand ($D = \sum_{i \in V_d} d_i$)
- P: total pickup demand ($P = \sum_{i \in V_p} p_i$)
- Q: vehicle capacity

Notice that this notation does not allow a customer to have both demands at the same time. Therefore, in such case, the customer is split into two different customers, one having the delivery demand, the other the pickup demand, and assuming a zero transportation cost between them.

2.1.1 Süral-Bookbinder

This formulation, presented in [30], uses three sets of decision variables. Let $x_{ij} = 1$ if customer i immediately precedes customer j ($i \neq j$) in the route and 0 otherwise. In order to define subtour elimination constraints, let y_i be the total delivery load on the vehicle when *departing* from customer i. Similarly, let z_i be the total pickup load on the vehicle when *arriving* at customer i. Then their mathematical model for the SVRPDSP is as the following.

Süral-Bookbinder formulation for the SVRPDSP

$$Min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} - \sum_{i \in V} \sum_{j \in V_p} r_j x_{ij} \tag{2.1}$$

Subject to:

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V_d \cup 0 \tag{2.2}$$

$$\sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = 0 \qquad \forall i \in V \tag{2.3}$$

$$y_i - y_j \geq d_j - D(1 - x_{ij}) \qquad \forall i, j \in V \setminus \{0\} \tag{2.4}$$

$$z_j - z_i \geq p_i - Q(1 - x_{ij}) \qquad \forall i, j \in V \setminus \{0\} \tag{2.5}$$

$$(y_i + d_i) + (z_i + p_i) \leq Q \qquad \forall i \in V \setminus \{0\} \tag{2.6}$$

$$y_i, z_i \geq 0, x_{ij} = \{0, 1\} \qquad \forall i, j \in V \tag{2.7}$$

The objective is to minimize the function (2.1), that represents the total cost of the solution, which is the sum of the transportation cost minus the revenue generated by all pickups collected. Constraints (2.2) specify that all delivery customers must be visited exactly once and that the vehicle must arrive at the depot exactly once. Equations (2.3) are flow conservation constraints, which specify that each arrival forces a departure. The inequalities (2.4) and (2.5) are adaptations of the well known Miller-Tucker-Zemlin (MTZ) subtour elimination constraints for the TSP [19]. The capacity of a vehicle must not be exceeded at any time (2.6). In case customer i is a delivery customer, these constraints specify that the total load arriving at i does not exceeds

the maximum capacity of the vehicle by the inequality $y_i + d_i + z_i \leq Q$. Similarly, if i is a pickup customer, the inequality $y_i + z_i + p_i \leq Q$ specifies that the load of the vehicle, when leaving customer i, must be within the capacity limit. Finally (2.7) are non-negativity and binary restrictions.

In addition Süral and Bookbinder proposed some improvements to strengthen their formulation and improve its efficiency, which will be described in the following.

2.1.1.1 Improvements

The set of arcs A can be partitioned into the disjoints subsets $V_d \times V_d$, $V_d \times V_p$, $V_p \times V_p$ and $V_p \times V_d$ where $C_1 \times C_2$ denotes the arcs from the customer set C_1 to C_2. They improve the coeficients of constraints 2.4 and 2.5 resulting in the following sets.

Constraint Disaggregation and Coeficient Improvement (CC)

$$y_i - y_j \geq d_j - min\{D, Q - p_j\}(1 - x_{ij}) \qquad \forall i, j \in V \setminus \{0\} \qquad (2.4.1)$$

$$z_j - z_i \geq p_i - min\{P, Q - d_i\}(1 - x_{ij}) \qquad \forall i, j \in V \setminus \{0\} \qquad (2.5.1)$$

Note that these disaggregation will only improve the arc sets $\{V_d \cup V_p\} \times V_p$ in 2.4.1 and $V_d \times \{V_d \cup V_p\}$ in 2.5.1. For constraints 2.4.1 this is valid since if the pickup customer j is not being served in a given solution $y_j = 0$, the left-hand side cannot be a negative value and, consequently, cannot be less than the value of the right-hand side. Furthermore, if j is in the solution, the smallest value the left-hand side can assume will be when i is a pickup customer not being served and $y_i = 0$. In such case, for a route to be feasible there must be enough space for the pickup demand of customer j in the vehicle, therefore $y_j \leq min\{D, Q - p_j\}$ and the constraint remains valid. The disaggregation in constraints 2.5.1 follows a similar idea.

These MTZ constraints can be improved even more by applying a lifting by bringing the variable x_{ji} into these inequalities, resulting in the following set of constraints.

Lifted MTZ Subtour Elimination Constraints (LS)

$$y_i - y_j + [min\{D - d_i - d_j, Q - p_j - d_i\}]^+ x_{ji} \geq d_j - \min\{D, Q - p_j\}(1 - x_{ij}) \qquad \forall i, j \in V \setminus \{0\} \qquad (2.4.2)$$

$$z_j - z_i + [min\{P, Q - d_i\} - p_i - p_j]^+ x_{ji} \geq p_i - \min\{P, Q - d_i\}(1 - x_{ij}) \qquad \forall i, j \in V \setminus \{0\} \qquad (2.5.2)$$

Regarding constraints 2.4.2 the basic idea is to tight as much as possible the constraint by improving the left hand side when $x_{ji} = 1$, which implies that $x_{ij} = 0$. For matters of simplicity, consider the term $[min\{D - d_i - d_j, Q - p_j - d_i\}]^+ x_{ji}$ as T. Notice that there are four possible cases, considering the types of customers i and j.

Therefore, in the first case suppose that $i, j \in V_p$. Thus, $y_j = y_i$, since none of the customers has delivery demand, and in T the resulting value will be $Q - p_j$. One can easily notice that in such case the left hand side will be equal to the right hand side, which satisfies the lifted constraint. The case where $i, j \in V_d$ can be proved to satisfy these constraints by a similar process.

The third possible case for constraints 2.4.2, refers to when $i \in V_p$ and $j \in V_d$. In such case, $y_j = y_i$, the term T will result in $D - d_j$, and the right hand side $D - d_j$, since $D \leq Q$. Therefore the lifted constraints remain valid. Finally, there is the case in which $i \in V_d$ and $j \in V_P$. Notice that the term $y_j - y_i \geq 0$ and that it is equal to the delivery demand d_i. Therefore if $D - d_i \leq Q - p_j - d_i$ in term T, the constraint will result in $D \leq min\{D, Q - p_j\}$, remaining valid. Otherwise, the constraint will result in $Q - p_j \leq Q - p_j$, which is also valid. Constraints 2.5.2 can be proved by a similar process.

In order to improve the capacity constraints (2.6), consider an additional variable q_i, which assumes the value 1 if pickup i is collected and 0 otherwise.

We may define these variables by (2.8), disaggregate the constraints for delivery and pickup nodes and set the capacity to 0 if the pickup node is not visited. The capacity may be improved further using the demand of the previous and next nodes, yielding the lifted constraints (2.6.1) and (2.6.2).

Logical Inequalities (LI)

$$\sum_j x_{ji} - q_i = 0 \qquad \forall i \in V \qquad (2.8)$$

$$y_i + z_i + \sum_{j \in V_d} d_j x_{ji} \leq Q - d_i \qquad \forall i \in V_d \qquad (2.6.1)$$

$$y_i + z_i + \sum_{j \in V_p} p_j x_{ij} \leq (Q - p_i) q_i \qquad \forall i \in V_p \qquad (2.6.2)$$

Note that the variable q_i in (2.6.2) may be eliminated using its definition (2.8), then none of the improvements presented so far increases the number of variables neither the number of constraints of the formulation. The following additional constraints increases the size of the formulation, but can still be useful: the first is 0-1 knapsack constraints, and the second is a set of cover inequalities, which is proved by [23] to be valid inequalities for a subset K if $\sum_{j \in M} p_i q_i \leq Q, \forall M \subset K$.

Knapsack Contraints (KC) and Cover Inequalities (CI)

$$\sum_{i \in V_p} p_i q_i \leq Q \tag{2.9}$$

$$\sum_{i \in V_p} q_i \leq |K| - 1 \qquad \forall K \subseteq V_p, \sum_{i \in K} p_i q_i > Q \tag{2.10}$$

2.1.2 Gribkovskaia-Laporte-Shyshou

In this formulation, each customer i is represented by two vertices, i and $i + n$, but these vertices do not represent delivery and pickup. Vertices $i = 1 \dots n$ represent the first visit on customer i and vertices $i+n = 1+n \dots 2n$ the second visit. The deliveries are performed in the first visit. The pickup may be performed simultaneously (in the first visit), separately (in the second visit) or not performed at all. Notice that this formulation assumes that all customers have both demands.

The variables x_{ij} are the same as in the previous formulation (1 if arc (i, j) is in the route, 0 otherwise). However they are not defined for arcs $(i, i + n)$ and $(i + n, i)$, since if a customer is visited twice, the two visits will never be consecutive in the route. Variables y_i and z_i have different meanings. They do not control the load on the vehicle: $y_i = 1$ if the pickup of customer i is made on a first visit, 0 otherwise; $z_i = 1$ if the pickup of customer i is made on a second visit, 0 otherwise. To control the load on the vehicle they introduce a new set of variables, w_i, which is the upper bound on the vehicle load when leaving vertex i.

For uniformity, we use the same notation V_d and V_p, however $V_d = \{1 \dots n\}$ and $V_p = \{n + 1 \dots 2n\}$, *i.e.* here V_p represents only the pickups served on a second visit.

Gribkovskaia-Laporte-Shyshou formulation for the SVRPDSP

$$Min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} - \sum_{i \in V_d} r_i y_i - \sum_{j \in V_p} r_{j-n} z_{j-n} \tag{2.11}$$

Subject to:

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V_d \cup 0 \tag{2.12}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V_d \cup 0 \tag{2.13}$$

$$\sum_{i \in V} x_{ij} = z_{j-n} \qquad \forall j \in V_p \tag{2.14}$$

$$\sum_{j \in V} x_{ij} = z_{i-n} \qquad \forall i \in V_p \tag{2.15}$$

$$z_i + y_i \leq 1 \qquad \forall i \in V_d \tag{2.16}$$

$$0 \leq w_i \leq Q \qquad \forall i \in V \tag{2.17}$$

$$w_0 = D \tag{2.18}$$

$$w_j \geq w_i - d_j + p_j y_j - Q(1 - x_{ij}) \qquad \forall i \in V; j \in V_d \tag{2.19}$$

$$w_j \geq w_i + p_{j-n} z_{j-n} - Q(1 - x_{ij}) \qquad \forall i \in V; j \in V_p \tag{2.20}$$

$$\sum_{(i,j) \in S} x_{ij} \leq |S| - 1 \qquad \begin{array}{r} \forall S \subset \{V\}\ ; \\ 2 \leq |S| \leq 2n\ ; \\ S \not\supseteq \{0, ..., n\} \end{array} \tag{2.21}$$

$$x_{ij}, y_i, z_i \in \{0, 1\} \qquad \forall i, j \in V \tag{2.22}$$

The objective function (2.11) minimizes the total routing cost deducting the revenue generated by collecting pickups. Constraints (2.12) and (2.13) imposes that the vehicle must leave the depot, visit all delivery customers only once and then return to depot. Constraints (2.14) and (2.15) assures that the route visit the customer a second time if the pickup is to be made on a second visit, otherwise the customer is not visited a second time. Constraints (2.16) guarantee that a pickup is made at most once for each customer. Constraints (2.17) control the upper bound on the load of the vehicle and constraints (2.18) set the upper bound load when leaving the depot as the total delivery demand. The set of constraints (2.19) and (2.20) defines w_j in terms of w_i and the pickup and delivery demand of j, when j is visited just after i, respectively for the first and second visit of j. Constraints (2.21) are classical subtour eliminations constraints. Notice that, since a solution in which all the deliveries are served is feasi-

ble, we do not add subtour constraints for sets including all those customers. Finally, constraints (2.22) define the binary variables.

2.1.2.1 Improvements

The formulation may be strenghten through lifting process on constraints (2.19) and (2.20). The process is similar to the one presented for the MTZ constraints in the Sural-Bookbinder formulation. For example, (2.19) can be rewritten as $w_j \geq w_i - d_j + p_j y_j - Q(1 - x_{ij}) + \alpha_{ji} x_{ji}$, where currently $\alpha_{ji} = 0$. They established the largest value of α_{ji} so that (2.19) remains valid. This value depends if i is the first or the second visit of the customer, and then (2.19) and (2.20) become four sets of lifted constraints. We present these improvements in the following equations.

Liftings on constraints 2.19 and 2.20

$$w_j \geq w_i - d_j + p_j y_j - (1 - x_{ij})Q + (Q + d_j + d_i - p_i - p_j)x_{ji} \quad \forall i \in V_d; j \in V_d \quad (2.23)$$

$$w_j \geq w_i + p_{j-n} z_{j-n} - (1 - x_{ij})Q + (Q + d_i - p_i - p_{j-n})x_{ji} \quad \forall i \in V_d; j \in V_p \quad (2.24)$$

$$w_j \geq w_i - d_j + p_j y_j - (1 - x_{ij})Q + (Q + d_j - p_{i-n} - p_j)x_{ji} \quad \forall i \in V_p; j \in V_d \quad (2.25)$$

$$w_j \geq w_i + p_{j-n} z_{j-n} - (1 - x_{ij})Q + (Q + p_{i-n} - p_{j-n})x_{ji} \quad \forall i \in V_p; j \in V_p \quad (2.26)$$

Moreover, in order to work also for the depot, before applying the lifting process, the vertex 0 is duplicated, 0 being the beginning of the route and $2n+1$ the end, which causes minor changes on constraints from (2.23) to (2.26). The new constraints are shown in the following.

Other liftings (duplicates depot in 2n+1)

$$w_j \geq w_0 - d_j + p_j y_j - (1 - x_{0j})Q \quad \forall j \in V_d \quad (2.27)$$

$$w_j \geq w_0 + p_{j-n} z_{j-n} - (1 - x_{0j})Q \quad \forall j \in V_p \quad (2.28)$$

$$w_0 \geq w_i + (d_i - p_i)x_{0i} - (1 - x_{0i})Q \quad \forall i \in V_d \quad (2.29)$$

$$w_0 \geq w_i + p_{i-n} x_{0i} - (1 - x_{0i})Q \quad \forall i \in V_p \quad (2.30)$$

$$w_j \geq w_{2n+1} - (1 - x_{j,2n+1})(Q + d_j) - p_j(x_{j,2n+1} - y_j) \quad \forall j \in V_d \quad (2.31)$$

$$w_j \geq w_{2n+1} - (1 - x_{j,2n+1})Q - p_{j-n}(x_{j,2n+1} - z_{j-n}) \quad \forall j \in V_p \quad (2.32)$$

$$w_{2n+1} \geq w_i - (1 - x_{i,2n+1})Q \quad \forall i \in V \quad (2.33)$$

Chapter 3

Methodology for the SVRPDSP

In this chapter we describe the methodology for the SVRPDSP. Since the metaheuristics have the same solution representation and share some procedures we first describe them. In the first section, we provide the solution representation used in our heuristic approaches. Further we describe our constructive heuristics in details, followed by a description of our repair and improvement heuristics. Next, the neighborhood structures used are presented. In the following we describe the metaheuristic approaches implemented for the SVRPDSP, including a Hybrid Evolutionary Algorithm, which uses a data mining strategy to track good attributes of solutions and guide the crossover and mutation operators; and a Variable Neighborhood Descent. In addition we present a Branch&Cut algorithm for the formulation of Gribkovskaia, Laporte and Shyshou [14] and a novel formulation for the SVRPDSP.

3.1 Solution representation

As in [6] a solution is represented by an array of integers, denoting the order in which the customers are visited. The depot is represented by the number 0 and appears twice to indicate where the route begins and where it ends. All customers that appears between the two 0's are in the route and the remaining are customers that will not be visited. As one can notice, only pickup customers can appear out of the route as only their demands are optional.

Since in our benchmark instances all customers have both demands, we assume that each customer i is represented by two different integers, i and $i + n$, respectively for the demand and pickup service, where n is the number of customers. Therefore solutions as the one presented in Figure 3.1 are possible and are represented as show in the same figure. Notice that as customers 7 and 8 are not being visited they appear

after the second depot in the representation, and that the demands of customer 1|5 are being performed simultaneously, while customer 2|6 has its pickup demand being performed in a second visit.

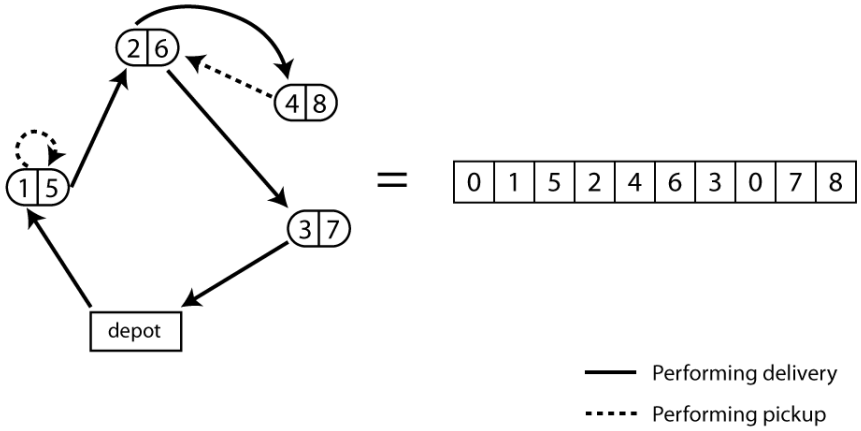


Figure 3.1: Example of solution for the SVRPDSP and its representation.

3.2 Constructive Heuristics

Four constructive algorithms were implemented, where the first two usually generate good quality solutions and the remaining two generate medium or even poor quality solutions. The only case in this work where poor quality solutions may be useful is in the EA metaheuristic, where there must be diversity among the individuals of the population.

3.2.1 TspBased

This heuristic is based on the one proposed in [6]. In this method, at first, a Traveling Salesman Problem (TSP) considering only the delivery customers is solved. The basic idea is to route these customers in the best way possible minimizing the transportation cost. Notice that the solution of this TSP is a valid solution for the SVRPDSP, since $D \leq Q$, and serving the pickup customers is optional. The following TSP formulation is used and solved using the software IBM ILOG CPLEX 12.3.

$$Min \sum_{i \in V_d \cup 0} \sum_{j \in V_d \cup 0} c_{ij} x_{ij} \tag{3.1}$$

Subject to:

$$\sum_{i \in V_d \cup 0} x_{ij} = 1 \qquad \forall j \in V_d \cup 0 \tag{3.2}$$

$$\sum_{j \in V_d \cup 0} x_{ij} = 1 \qquad \forall i \in V_d \cup 0 \tag{3.3}$$

$$\sum_{j \in V_d \cup 0} (f_{ik} - f_{ki}) = 1 \qquad \forall k \in V_d \tag{3.4}$$

$$f_{ij} \leq |V_d| x_{ij} \qquad \forall i, j \in V_d \cup 0; i \neq j \tag{3.5}$$

$$x_{ij} \in \{0, 1\}; f_{ij} \geq 0 \qquad \forall i, j \in V_d \cup 0 \tag{3.6}$$

For this formulation consider $x_{ij} = 1$ if arc (i, j) is being used and 0 otherwise. Variables f are flow variables from network flow problems and f_{ij} defines the flow of the arc (i, j). The objective function (3.1) minimizes the total routing cost. Constraints (3.2) and (3.3) specify that the vehicle arrives and leaves exactly once each customer. The set of constraints (3.4) defines a continuous flow throughout the route. Inequalities (3.5) specify that if arc (i, j) is used its flow must be within the total number of customers plus the depot, otherwise set the flow as zero. Finally (3.6) are non-negativity and binary restrictions.

The optimal route of the TSP is used as base for inserting the pickup customers, possibly improving the current solution, which at this point only has the delivery customers being served. The insertion of the pickups is performed iteratively. At each iteration, all possible positions to insert all the remaining pickups are evaluated and the best candidates and their positions to insert stored in a *Restricted Candidate List* (RCL) with size $rclSize$. We emphasize that only feasible and profitable insertions are considered candidates to be stored in the RCL, therefore all candidates in the RCL would improve the current solution if inserted. Figure 3.2 shows a flowchart for this heuristic.

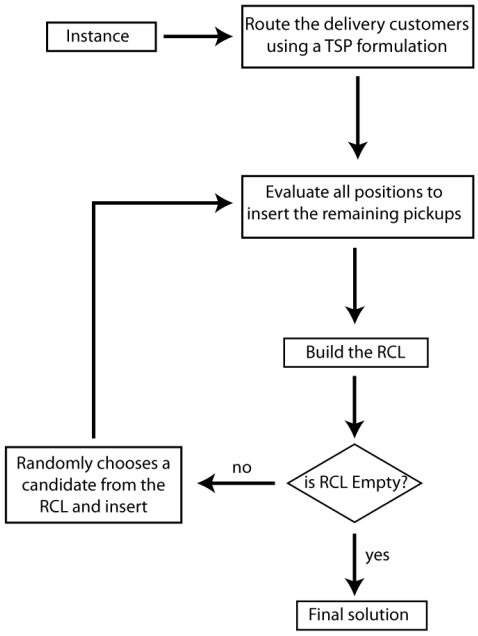


Figure 3.2: Flowchart of the constructive heuristic tspBased

The difference between this heuristic and the one presented in [6] is the addition of a RCL, which allows different solutions to be generated when $rclSize > 1$.

3.2.2 TspKnapsackBased

This constructive heuristic is very similar to the previous one, differing only by not considering all pickup customers in the evaluation that generates the RCL. Basically we consider only the pickup customers with the best cost-benefit and in order to discover which are these customers we solve the well known Knapsack Problem, considering the pickup customers as the items of the knapsack, their pickup demand as the weights and their revenues as the values of the items. If all p_i are integers, a pseudo-polynomial dynamic programming may be used, but for matters of generality we prefer to use a method suitable for all cases. Many other methods could be used, however the SVRPDSP instances are rather small for the Knapsack Problem, and the following simple ILP works well and is solved using the software IBM ILOG CPLEX 12.3.

$$Max \sum_{i \in V_p} r_i x_i \tag{3.7}$$

Subject to:

$$\sum_{i \in V_p} p_i x_i \leq Q \tag{3.8}$$

$$x_i \in \{0,1\} \qquad \forall i \in V_p \tag{3.9}$$

3.2.3 nearestNeighbor

At first the solution contains only the depot, and one by one the customers are inserted into the route. The nearest customers are inserted first. It differs from the classical nearest neighborhood heuristic by considering a RCL with size equal to $0.1n$, where n is the number of customers of a given instance. Through tests it was observed that, for this problem, this algorithm yields better results the greedier it is and the value $0.1n$ has proven to balance well the quality and the generation of different solutions. Is is important to emphasise that the importance of this algorithm in the present work is only for generating diversity in the EA population, not necessarily good quality solutions, so further tests with the RCL size were not done.

3.2.4 cheapestInsertion

The solution has, initially, only the depot, and at each iteration a new customer is inserted in the route on a good position considering the cost-benefit criterion. For instance, if a pickup customer k is inserted between customers i and j, the impact in the solution cost is $c_{ik} + c_{kj} - c_{ij} + r_k$. Notice that for delivery customers the impact is $c_{ik} + c_{kj} - c_{ij}$. It uses an RCL of size $0.1n$ as in the *nearestNeighbor* heuristic.

3.3 Repair and improvement heuristics

Since solutions are frequently shaken and combined during the proposed metaheuristics, it is not uncommon that unfeasible solutions are generated. Instead of simply reject such solutions, a repair heuristic is used, what possibly improves the exploration of the search space.

The proposed repair method works in two phases. At first, all the delivery customers not being served are inserted in the best possible position of the current route, according to the cost value, ensuring that the constraint of serving all delivery demands is satisfied. Notice that, in a feasible route, any position is feasible to insert a delivery

customer. In case the solution is still unfeasible, the route is analyzed to find where there are pickups overweighting the vehicle, removing them from the route and thus ensuring feasibility.

As one can notice, pickup customers are frequently removed from the route in the process. But if they are removed from a position in the route because of overweighting the vehicle, maybe there is a feasible and profitable position later in the route, when more space is left by the deliveries performed. Therefore, after repairing a solution, this condition is verified and the solution may be improved.

The improvement procedure takes advantage of a certain property of the problem. For instance, in case a pickup demand is served at the position i in a given feasible route, it means that at any position greater than i this demand can be fulfilled without making the route unfeasible, since if there was enough space to perform the pickup at that point, there will be, for certain, space for it at any time after, as the delivery load never increases along the route. Therefore if a customers' pickup demand is performed before its delivery demand, this solution is never the optimal, since both demands can be fulfilled simultaneously with cost zero. In such cases the pickup service is moved to be made just after the delivery service of the same customer. One can easily notice that this property is only valid if there are customers with both demands, therefore if an instance is only composed by customers with either delivery or pickup demands, this procedure will not make any difference.

3.4 Neighborhood Structures

All of the proposed metaheuristics, including the Hybrid Evolutionary Algorithm uses multiple neighborhood structures. Basically, we have chosen three classical structures, 2-opt, swap and k-or-opt, which are described in the following.

3.4.1 2-Opt

This neighborhood structure consists in removing a segment of the route, and reinserting it reversed. The two cutting points are randomly chosen, what implies that the segment does not have a fixed size. An example is shown in Figure 3.3.

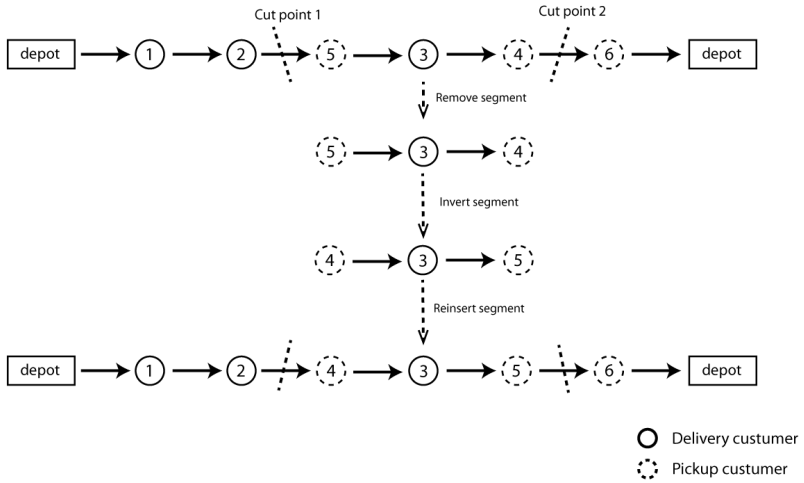


Figure 3.3: Example of a 2-opt neighbor generation.

3.4.2 Swap

Simply swaps two randomly chosen customers from a solution. Figure 3.4 shows an example of how it works. Notice that pickup customers not being served may be included in the route since any customer in a solution can be chosen, even the depot.

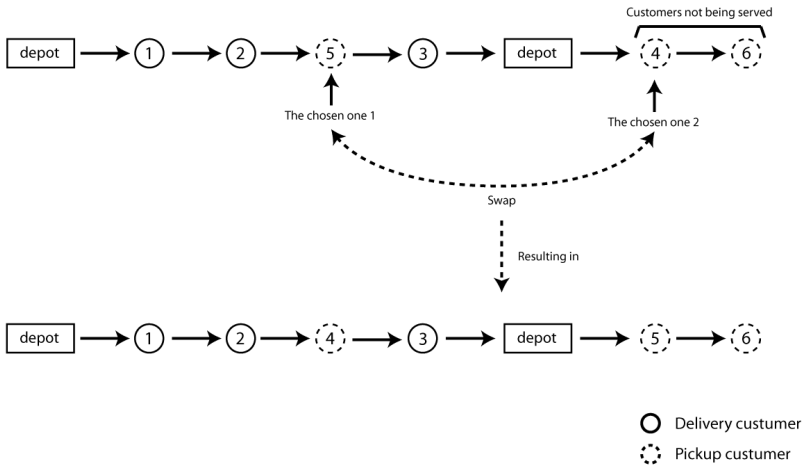


Figure 3.4: Example of a Swap neighbor generation.

3.4.3 k-or-opt

At first, k consecutive customers are removed from the route. Then they are reinserted in the best possible position. Notice that the best are inserted first and that only customers within the route are considered, therefore pickups not being served does

not take part on this neighborhood structure. Figure 3.5 shows an example of this neighborhood with $k = 2$.

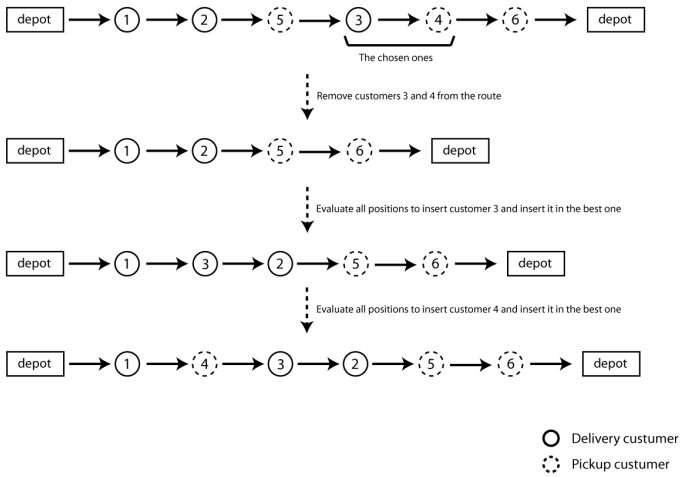


Figure 3.5: Example of a k-or-opt neighbor generation with $k = 2$.

3.5 Evolutionary Algorithm

Basically this Evolutionary Algorithm has five distinct phases, which are the initialization, crossover, mutation, intensification and diversification. The pseudocode 1 shows the steps of the algorithm, which are described in details in the following.

Algoritmo 1 Evolutionary Algorithm pseudocode

```
1:  procedure EVOLUTIONATY(popSize, numIt)
2:      //Initialization
3:      itsWithoutImpr ← 0
4:      intensity ← 1          // When a new best solution is found it is reset to 1
5:      pop ← ∅          // population
6:      bestSol ← any with cost = ∞          // best solution
7:      for i ← 1 to popSize do
8:          constructive ← random constructive heuristic
9:          rclSize ← random from {1, 2, 3}
10:         Generate individual using constructive and rclSize
11:         pop ← pop ∪ individual
12:         Update bestSol if necessary
13:     end for
```

```
14:  Compute patternsList list
15:
16:  //EA main loop
17:  for i ← 1 to numIt do
18:      //Crossover
19:      childPop ← ∅
20:      for j ← 1 to popSize/2 do
21:          parents ← random 2 individuals from pop
22:          //Do this for each of the 2 parents
23:          child_1 ← force a pattern of parents[0] into the route of parents[1]
24:          child_2 ← force a pattern of parents[1] into the route of parents[0]
25:          if child_1 is feasible then
26:              childPop ← childPop ∪ child_1
27:              Update patternsList list using child_1
28:              Update bestSol if necessary
29:          end if
30:          if child_2 is feasible then
31:              childPop ← childPop ∪ child_2
32:              Update patternsList list using child_2
33:              Update bestSol if necessary
34:          end if
35:      end for
36:      newPop ← best popSize/2 from childPop ∪ pop
37:      Add popSize/4 from tournament(pop,childPop)
38:      Add popSize/4 random from childPop ∪ pop
39:
40:      //Mutation
41:      for j ← 1 to popSize/2 do
42:          individual ← random from pop
43:          newMut ← mutate individual
44:          Update patternsList list using newMut
45:          Update bestSol if necessary
46:      end for
```

```
47:         //Intensification
48:         for j ← 1 to popSize/5 do
49:             ind ← random individual from pop
50:             for i ← 0 to intensity do
51:                 Shake ind
52:             end for
53:             ind ← LocalSearch to ind
54:             Update patternsList list using ind
55:             Update bestSol if necessary
56:         end for
57:
58:         //Diversification
59:         if i mod 10 = 0  then          // Every 10 iterations
60:             for j ← popSize − 1 to popSize/2 do          // Replace the worst
61:                 newInd ← random by constructive
62:                 Update patternsList list using newInd
63:             end for
64:         end if
65:
66:         if itsWithoutImpr = 5 then
67:             itsWithoutImpr ← 0
68:             intensity + +
69:         else
70:             itsWithoutImpr + +[1]
71:         end if
72:     end for
73: end procedure
```

In the **Initialization phase**, the first population is set. Each individual is generated by randomly selecting a constructive algorithm among the four types available, described in details in section 3.2. In case the chosen algorithm is either *tspBased* or *tspKnapsackBased*, the *rclSize* must be set. If it is the first time this type of constructive is called, the *rclSize* is set as 1 (greedy solution), otherwise it is set as 2. As equal elements are not allowed at any time in the population the constructives are called until the population has been fully generated with mutually different elements. As one can notice, it may be necessary many calls to the constructive algorithms to assure

this constraint. This can decrease its performance and then, should be avoided. In order to do so, if a new element is not successfully inserted into the population within 3 iterations, the value of $rclSize$ is set as 3 increasing the chances of the constructives *tspbased* or *tspKnapsackBased* generate a different element.

The second phase is the Crossover procedure, in which elements of the population are combined to generate new solutions so as to improve the quality of the individuals and promote, at some degree, diversification.

In order to produce better individuals the crossover procedure needs a criteria to evaluate the features of a given individual and decide which ones are the best. Therefore, before continuing explaining the crossover procedure we firstly go through the details of how our EA keeps track of good attributes of a solution.

Basically we use a Data Mining strategy, in which every new individual has its route analyzed to extract patterns (sequence of customers) within a given range $[minPatternSize, maxPatternSize]$. Each pattern found is stored in a structure called *patternsList* along with the frequency it has appeared in the solutions already analyzed. In addition to these information, we also keep record of the average cost of the route in which the pattern was found so as to improve the robustness of the evaluation criteria that decides how good a pattern is. Therefore we have two types of data to evaluate a pattern: frequence and average cost. Good patterns have high frequency and low average cost. Since cost value is usually much higher than the frequency value, this data must be normalized. Lets call $nFrequency$ the normalized frequency value, $nAvgCost$ the normalized average cost. Therefore we define $qualityIndex = (1 - nAvgCost) + nFrequency$, as the value used to evaluate the patterns, since it considers both measures. The closer to 2 the better. This is not the first time an approach combining a heuristic and a data mining algorithm is proposed for a vehicle routing problem. In [28], Santos *et al* proposed 4 approaches for a single vehicle routing problem, including one that combines a Genetic Algorithm with the data mining algorithm Apriori. Our approach is not based on their approach and is fairly different from the algorithm they developed.

Since the number of insertions and searches through the *patternsList* is high, this structure is implemented as a hash table to optimize these operations. Figure 3.6 shows how this structure is organized. To describe the hash function consider $P = (c_1, c_2, ...c_m)$ as a pattern, or in other words, a segment of a route; c_i is a customer and m the size of the pattern. The hash function used is $hashValue_P = c_1 10^1 + c_2 10^2 + .. + c_m 10^m$. This hash function balances well the number of collisions and the number of keys in the hash table, although further tests were not performed.

Once explained the *patternsList*, we can now detail the **crossover phase**. At

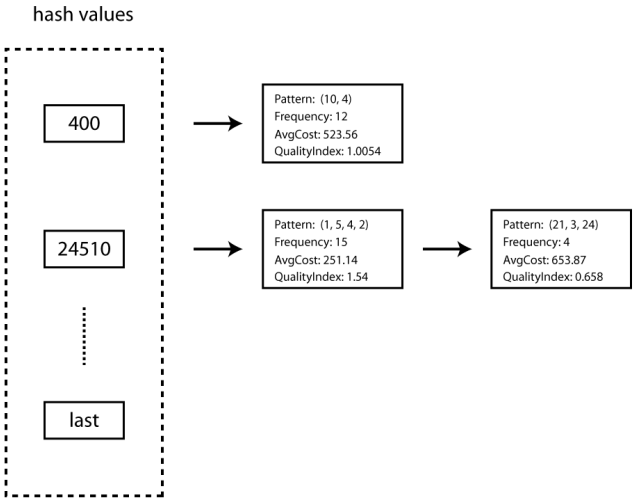


Figure 3.6: Example the structure of *patternsList*.

first two elements of the population are randomly selected and the patterns of both parents extracted along with their respective $qualityIndex$ value. Let us refer to the parents as P_1 and P_2. One pattern of P_1 is chosen, giving more probability for the ones with higher $qualityIndex$ values, and it is forced to appear in the best possible position of P_2's route generating a new individual.Then, we do the opposite, a pattern of P_2 is chosen to be forced into P_1's route. Therefore for each crossover iteration 2 children are generated. It is important to notice that in order to efficiently force a pattern, we evaluate every possibility to force a given pattern into the route and choose the best one considering the cost of the new route. However we consider only the positions where one of the customers in the chosen pattern is present. For instance, consider the route $R = (0, 1, 2, 4, 5, 3, 0)$ and the pattern $P = (3, 2)$. There are two different possibilities to force the pattern. The first one would produce the route $R^1 = (0, 1, 3, 2, 4, 5, 0)$ and the second $R^2 = (0, 1, 4, 5, 3, 2, 0)$. The crossover procedure is performed $popSize/2$ times in each iteration. Figure 3.7 illustrates one iteration of the process.

After the child population has been completely generated, it is time to combine it with the parent population to decide which individuals will be held for the next generation. The new population is half composed by the best elements of both populations. A quarter is selected through tournaments, where the best of two elements (one from each population) wins and is inserted in the new population. The other quarter is composed of randomly selected individuals. This three stage process enhance the diversity, which is a very important factor for the success of this kind of algorithm.

In the **mutation phase**, one pattern is selected from the *patternsList* to be forced into a randomly selected individual of the population, giving more chances the

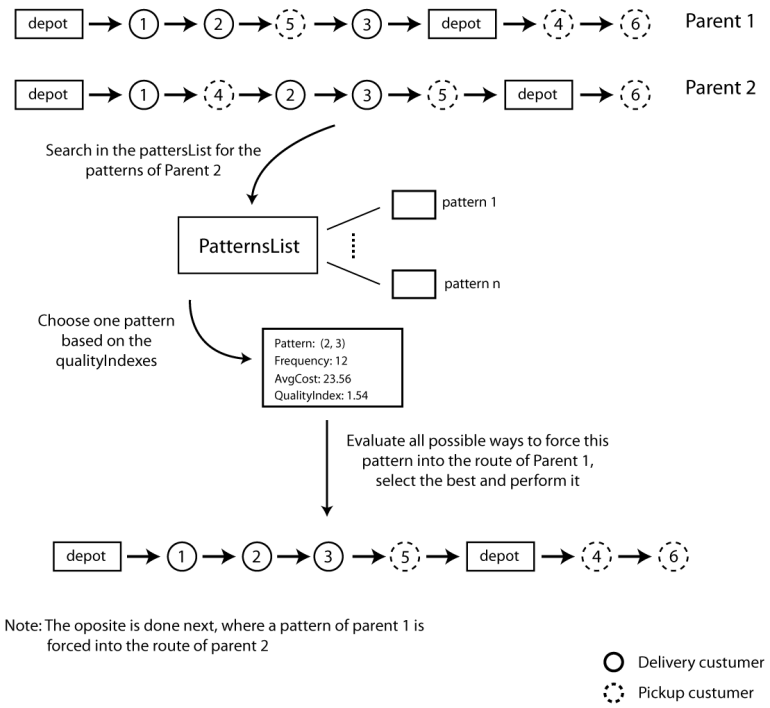


Figure 3.7: Example of a crossover iteration.

higher the value of $qualityIndex$. In case the new individual is better than the original, it replaces the latter. It is important to emphasize that, as in the crossover phase, the function which forces the patterns to appear in a given individual never generates unfeasible solutions, since it uses the repair procedure, described in section 3.3.

In the **intensification phase** an individual is randomly selected and its route shaken *intensity* times, using one of five different structures to shake a route, also randomly selected, which are 2-Opt, Swap and k-or-opt, with $k = \{1, 2, 3\}$, described in 3.4. Initially $intensity = 1$ and at each 5 iterations without improvement on the best solution, its value is increased by one. When the best solution is updated the value is reset to 1. A local search (Algorithm 2) procedure is then applied to the shaken individual. This procedure randomly select a neighborhood structure and completely explores it, returning the local minima. It uses the same neighborhood structures used by the shake method.

Finally the **diversification phase** simply replaces half of the population (the worst individuals) by new ones, generated by the constructive algorithms at each 10 iterations.

This Evolutionary Algorithm is an improved version of the one we published in [5]. The current version uses the $qualityIndex$, which considers both frequency and

Algoritmo 2 Local Search pseudocode

```
1: procedure localSearch(individual)
2:     neighborhood ← randomly select from {2-opt, swap, 2-Or-Opt, 3-Or-Opt, 4-Or-Opt}
3:     newind ← local optima of neighborhood using individual
4:     return newind
5: end procedure
```

average cost of the patterns. The earlier version used only the frequency. Besides that we propose and include the repair and the improvement procedures, which were not present in the previous EA.

3.6 Variable Neighborhood Descent

A Variable Neighborhood Descent (VND) [16] was also implemented and its steps are shown in pseudocode 3. Notice that in order to set the initial solution we run our constructive algorithms *tspBased* with $rclSize = 1$ and *tspKnapsackBased* with $rclSize = 1, 2$ and select the best solution generated. These configurations were chosen since they are the ones which generate the best solutions. Computational results to support this statement will be presented in section 5.2.

At each iteration the solution is shaken and one of five neighborhood structures is completely explored, returning the local optima. In case this solution is better than the best found it replaces the latter. The neighborhood structures used are the same as the ones presented in section 3.4, except that the neighborhood is completely explored. The neighborhoods are explored sequentially: if one cannot improve the current solution, the next one is tried; everytime a solution is improved, the search starts again from the first neighborhood. The procedure stops when the fifth neighborhood is explored and the best solution found is not updated.

Algoritmo 3 Variable Neighborhood Descent pseudocode

```
1: procedure VND(individual)
2:     bestSol ←tspBased(rclSize = 1)
3:     updateBestSol(tspKnapsackBased(rclSize = 1))
4:     for i ← 1 to 5 do
5:         updateBestSol(tspKnapsackBased(rclSize = 2))
6:     end for
7:     neighborhood ← 1
8:     while neighborhood < 6 do
9:         solAux ← bestSol
10:        switch neighborhood
11:            case 1
12:                solAux ← 2-opt-complete(bestSol)
13:            case 2
14:                solAux ← swap-complete(bestSol)
15:            case 3
16:                solAux ← 1-or-opt-complete(bestSol)
17:            case 4
18:                solAux ← 2-or-opt-complete(bestSol)
19:            case 5
20:                solAux ← 3-or-opt-complete(bestSol)
21:        if solAux.cost < bestSol.cost then
22:            bestSol ← solAux
23:            neighborhood ← 1
24:        else
25:            neighborhood + +
26:        end if
27:    end while
28: end procedure
```

3.7 A Branch&Cut for the Gribkovskaia-Laporte-Shyshou formulation

Gribkovskaia, Laporte and Shyshou present a formulation for the SVRPDSP in [14], which is described in section 2.1.2. By performing some preliminary tests we identi-

fied that this formulation is much less efficient than the one proposed by Sural and Bookbinder [30]. It is clear that the subtour elimination constraints play a important role in the efficiency of this formulation. These constraints we are referring to are the following.

$$\sum_{(i,j)\in S} x_{ij} \leq |S| - 1 \qquad \begin{array}{r} \forall S \subset \{V\} \ ; \\ 2 \leq |S| \leq 2n \ ; \\ S \not\supseteq \{0, ..., n\} \end{array} \tag{3.10}$$

One can easily notice that the number of these constraints grows exponentially to the number of customers, therefore they have a considerable impact on the efficiency of the model, even running it in commercial and very optimized softwares as IBM ILOG CPLEX.

We propose a Branch&Cut algorithm to add the subtour elimination constraints to the model as they are needed, instead of adding them all at once. This could significantly increase the performance of the model. For that end we need to be able to identify whether a given solution violates (3.10) or not, i.e., has subcycles or not. Notice that during the Branch&Cut there are two kinds of nodes: integers and fractionals. A node is integer if all x variables have integer values; and fractional if at least one x is fractional. While nodes of the former type are feasible solutions for the SVRPDSP, the latter represents only partial solutions, since some binary variables assume linear values.

Identifying subtours in integers nodes is fairly simple. Consider a given integer node n_i and that the instance being solved has n customers. The formulation even without the subtour elimination constraints enforces that a route must begin and end at the depot, therefore we are able to know how many customers are in this route by tracing it using the values of variables x. In case any delivery customer is not visited in this route we have at least two subtours and, conveniently, the route we traced to come to this conclusion is one of them. The other subtours (if there are) can be found using the same process, only changing the initial node to one that is not part of any subtour already identified.

In the other hand, the process to identify whether a fractional node has subtours is not that simple. Crowder and Padberg presents a method in [7] to identify subtours in fractional nodes of a TSP problem, which we used for the same purpose in the SVRPDSP. From now on we are going to refer to this method as the Shrink process.

Basically the Shrink process works by merging vertices to find out if there are and which are the subtours. It is important to emphasize that the method must be applied

to an undirected graph. Therefore, since a solution for the SVRPDSP is represented by a directed graph, the first step is to convert it to an undirected graph.

At each iteration it is verified if there are any edges with value greater than or equal to one. In case there are, one is selected and the vertices at the ends of the edge are merged. The new vertex shares the same edges of its former vertices, therefore in case the merged vertexes were connected to a common vertex other than them, the two edges become one and their values are summed. A subtour can be identified if an edge with value greater than one is generated after shrinking. In such case there is a subtour connecting the vertices that are part of both ends. This is valid since after shrinking these vertices, the sum of the edges connecting the resulting vertex to the others will result in a value less than 2, which is a behavior that does not occur in a solution without subtours: the sum must be at least 2, since the route must arrive in at least one customer of this subset and also leave this subset at least once. Figure 3.8 depicts an example of the process. Notice that, besides the subtour constraints, this fractional solution satisfy all other constraints, including the flow constraints, as the in and out degree of each vertex sums 1. Notice that after shrinking vertices 1 and 2 an edge with value 1.5 is generated. Therefore there is a subtour connecting vertexes 1,2 and 3. This information can be used to create a cut constraint for this node, using the equations 3.10 where $S = \{1, 2, 3\}$. Furthermore notice that vertexes 4,5 and 6 are also connected by a subtour. The process continues until is no longer possible to shrink the graph.

The Branch&Cut algorithm implemented uses these two methods to analyze each node generated in the branching tree, be it an integer node or a fractional one. In case the node has subtours, a cut for each subtour is generated using equations 3.10 and added to the model. Therefore, the optimal solution found in the end of the run will not contain subtours even without adding all possible subtour elimination constraints.

3.8 A novel MILP formulation

Baldacci, Hadjiconstantinou and Mingozzi in [1] proposed a two-commodity network flow formulation for the Capacitated Vehicle Routing Problem (CVRP), which was based on a TSP formulation introduced by Finke, Claus and Gunn [10]. In this TSP formulation a commodity A is delivered during the tour and a commodity B is collected. To visit n customers, the salesman departs with n units of A and 0 of B. At each customer one unit of A is delivered and one of B is collected. Then, the salesman ends the route with 0 units of A and n of B. Notice that at any arc in the tour the salesman

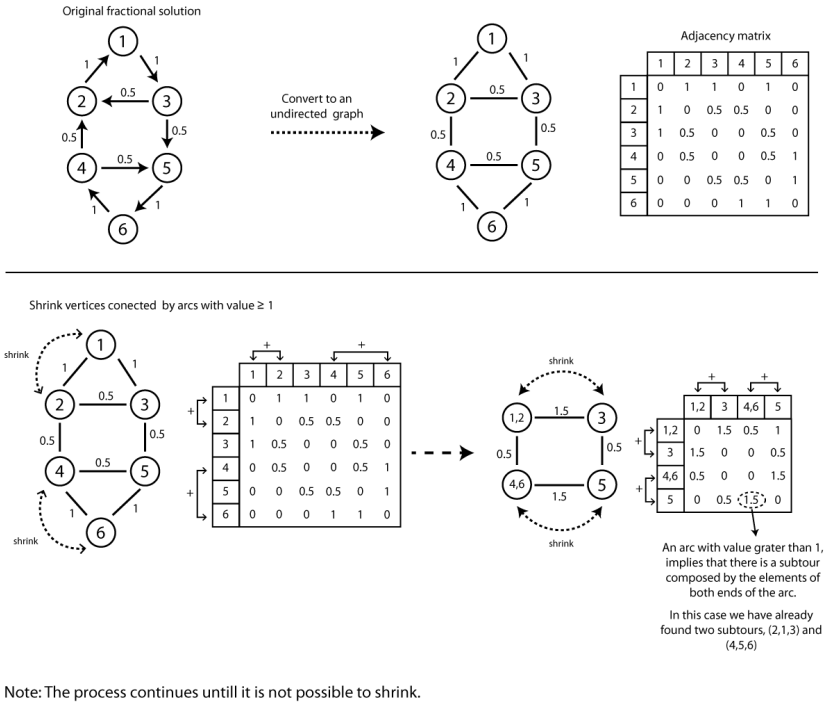


Figure 3.8: Example of the shrink process.

carries a total of exactly n units.

Similarly, for the CVRP, considering a demand d_i for each customer i, let $D = \sum_i d_i$ be the vehicle capacity. The vehicle departs fully loaded, with D units of commodity A, drops the demand at each customer visited and ends the route with 0 units. While doing the tour, it collects d_i units of commodity B at each customer, ending up with D units of commodity B. Notice that while the load of commodity A decreases, commodity B increases by the same amount. Then, commodity B may be viewed as the free space on vehicle, since on each arc the total load of both commodities combined is exactly D. Let f_{ij} and g_{ji} be respectively the vehicle load of commodity A and the free space when travelling through arc (i, j). Notice that the indices in g are reversed, as the free space (or commodity B) increases while following the same tour, but in the reverse order.

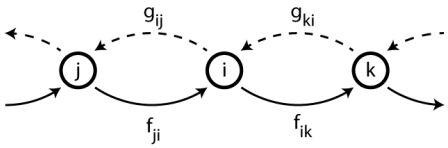


Figure 3.9: Route passing through customers j; i; k.

Figure 3.9 shows a general example of a subtour using variables f and g, passing through three customers. Notice that equations (3.11) and (3.12) are valid, and we can prove the following lemma from them:

$$f_{ji} - f_{ik} = d_i \tag{3.11}$$

$$g_{ki} - g_{ij} = d_i \tag{3.12}$$

Lemma 3.8.1. *For each customer i, the difference between the total value of the incoming and outgoing arcs are exactly $2d_i$.*

Proof. Without lost of generality consider the customer i of Figure 3.9. Arcs not used in the tour have $f = g = 0$, then the incoming arcs sum $f_{ji} + g_{ki}$ and the outgoing arcs sum $f_{ik} + g_{ij}$. Using (3.11) and (3.12) we have:

$(f_{ji} + g_{ki})$ - $(f_{ik} + g_{ij}) = f_{ji} - f_{ik} + g_{ki} - g_{ij} = 2d_i$ □

This property is used in the following formulation we propose for the SVRDSP.

3.8.1 The proposed four-commodity network flow formulation

As in the formulations of Sural-Bookbinder (section 2.1.1) and Gribkovskaia-Laporte--Shyshou (section 2.1.2) we use the variable x to control which arcs are being used in the route. Therefore let $x_{ij} = 1$ if arc (i, j) is used and 0 otherwise. Inspired by the two-commodity network flow formulation for the CVRP previously mentioned we use variables f and g to control the load on the vehicle on each arc. Since we have delivery and pickup demands, we actually propose a four-commodity network flow formulation. Then let f^d_{ij} and g^d_{ji} be the delivery load and free space, respectively, when travelling through arc (i, j), and f^p_{ij} and g^p_{ji} be the pickup load and free space, respectively. It is important to notice that free space here is considered separately for delivery and pickup demands, i.e., pickup demands actually uses the free space of delivery load and vice-versa, although the delivery demands can be different from the pickup demands.

Variables f^d are similar to the y variables of the Sural-Bookbinder formulation, but represents the load of an *arc* being traversed, not upon departure of a vertex. They can be related by the expression $y_i = \sum_{(i,j)\in A} f^d_{ij}$. Similarly, f^p variables are related to the z variables of the Sural-Bookbinder formulation. Therefore we use variables f and g to control the flow of load, avoiding subtours.

Then, we propose the following novel formulation for the SVRPDSP, which we named as four-commodity network flow formulation.

The four-commodity network flow formulation for the SVRPDSP

$$Min \sum_{(i,j)\in V} c_{ij}x_{ij} - \sum_{(i,j)\in V} r_j x_{ij} \tag{3.13}$$

Subject to:

$$\sum_{i\in V} x_{ij} = 1, \qquad \forall j \in V_d \tag{3.14}$$

$$\sum_{i\in V} x_{ij} \leq 1, \qquad \forall j \in V_p \tag{3.15}$$

$$\sum_{j\in V} x_{ij} - \sum_{k\in V} x_{ki} = 0, \qquad \forall i \in V \tag{3.16}$$

$$f^d_{ij} + f^p_{ij} \leq Q x_{ij}, \qquad \forall i,j \in V \tag{3.17}$$

$$f^d_{ij} + g^d_{ji} = Q x_{ij}, \qquad \forall i,j \in V \tag{3.18}$$

$$f^p_{ij} + g^p_{ji} = Q x_{ij}, \qquad \forall i,j \in V \tag{3.19}$$

$$\sum_{j\in V} \left(f^d_{ji} + g^d_{ij}\right) - \sum_{j\in V} \left(f^d_{ij} + g^d_{ji}\right) = 2d_i, \qquad \forall i \in V_d \tag{3.20}$$

$$\sum_{j\in V} \left(f^p_{ji} + g^p_{ij}\right) - \sum_{j\in V} \left(f^p_{ij} + g^p_{ji}\right) = 2p_i \sum_{j\in V} x_{ij}, \qquad \forall i \in V_p \tag{3.21}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i,j \in V \tag{3.22}$$

$$f^d_{ij}, f^p_{ij}, g^d_{ji}, g^p_{ji} \geq 0, \qquad \forall i,j \in V \tag{3.23}$$

The objective function (3.13) minimizes the total route cost deducting the revenue generated by collecting pickups. The route must visit exactly once each delivery customer (3.14) and may visit at most once the pickup customers (3.15). By constraints (3.16) each arrival produces a departure. The set of constraints (3.17) limits the combined amount of delivery and pickup load on each arc: the vehicle capacity if the arc is used, and zero otherwise. Constraints (3.18) and (3.19) define g's variables in terms of f's variables. Constraints (3.20) and (3.21) assures Lemma 3.8.1 respectively on delivery and pickup customers. For pickup customers the lemma holds only for the ones visited; for the non visited the load flow is zero. Finally, constraints (3.22) and (3.23) define the variables. In this formulation, subtour elimination is implicitly considered in the load flow variables f and g.

3.8.1.1 Improvements on the formulation

Coefficients of constraints (3.17) may be slightly improved if node i is a delivery customer and/or node j is a pickup customer. If i is a delivery customer, then the total load when leaving i cannot exceed $Q - d_i$, since d_i units of demand were dropped at customer i. Similarly, if j is a pickup customer, the total load just before arriving j

cannot exceed $Q - p_j$, as p_j units will be collected at customer j. Combining both cases, we have the following improvement in the coefficients.

Coefficient Improvement on capacity constraints (CI)

$$f_{ij}^d + f_{ij}^p \leq (Q - max(d_i, p_j))x_{ij}, \qquad \forall i, j \in V \tag{3.17.1}$$

Consider a customer i having delivery and pickup demands at the same time. In such case, as explained in section 2.1, this customer would be split into two customers, one with each demand. Therefore there are two different ways of performing both demands simultaneously (in a single visit), first the delivery demand and then the pickup or the opposite. As one can notice, it makes no difference the order these demands are fulfilled, disconsidering feasibility. However the model considers each possibility as a different solution with same cost value.

In our test problems all customers have both demands, so the number of solutions with this kind of symmetry is exponential, thus we propose the following set of constraints (3.24) to remove this symmetry by not allowing a pickup demand to be performed just before the delivery demand of the same customer. Then, if they are to be served simultaneously, the route in the model visits the split customers serving the delivery and then the pickup demand consecutively, not the opposite. Notice that, because of the vehicle capacity, we cannot force the opposite, since being feasible to perform the delivery and then the pickup does not imply that the opposite would also be feasible. This is the case when the pickup demand collected occupies the free space left by the delivery goods dropped.

Anti-symmetry constraints (AS)

$$x_{i+n,i} = 0 \qquad \forall i \in V_d \tag{3.24}$$

Furthermore, even when the services are made in two different visits, it is never sub-optimal to serve the delivery demand first. If the pickup demand can be collected in the first visit, it certainly can be collected in the second, as the delivery load never increases, and the pickup, once collected, is never dropped during the route. Then, if there is enough space to collect it in the first visit, there will be for certain in the second visit. This is not necessarily true in the opposite way. The following set of constraints, together with (3.24), assures that the pickup service of a given customer is never done before its delivery service. Since the delivery load decreases along the route and the pickup load tends to increase, we use variables f^d and f^p, which denote, respectively,

the delivery and pickup load in a given arc, to force that if a pickup is served, the pickup load (f^p) when leaving the delivery customer must be less than or equal to the pickup load when leaving the respective pickup customer, therefore the pickup cannot be performed before the respective delivery demand of the same customer. In order to tight even more these constraints we proposed the following constraints, which force that the delivery load (f^d) in the vehicle when leaving the delivery customer must be greater than or equal to the pickup load when leaving the respective delivery customer.

Additional constraints (AC)

$$\sum_{j \in V} f^p_{ij} \leq \sum_{j \in V} f^p_{i+n,j} + (1 - \sum_{j \in V} x_{i+n,j})M \qquad \forall i \in V_d \qquad (3.25)$$

$$\sum_{j \in V} f^d_{ij} \geq \sum_{j \in V} f^d_{i+n,j} \qquad \forall i \in V_d \qquad (3.26)$$

Finally, the same Knapsack constraints proposed for the Süral-Bookbinder formulation can be used in our formulation, to constrain the maximum pickup load that the vehicle can carry. We repeat then in the following for convenience:

Knapsack constraints (KC)

$$\sum_{i \in V} x_{ij} - q_i = 0, \qquad \forall i \in V_p \qquad (3.27)$$

$$\sum_{i \in V_p} p_i q_i \leq Q \qquad (3.28)$$

Chapter 4

Methodology for the MVRPDSP

This chapter presents the methodology used for the MVRPDSP. Similarly to the methodology for the single vehicle version, we begin by describing our solution representation, which is used by our hybrid constructive heuristic. Further we present our hybrid constructive heuristic and finally the two mathematical formulations, along with some improvements.

4.1 Solution representation

A solution for the MVRPDSP is a set of routes. Each route is represented by an array of integers representing the customers in the route in the same way as in the solution representation of the SVRPDSP, although pickup customers not being served are not appended after the second depot, which represents the end of the route. Pickup customers not being served are stored in a separated array of integers.

As for the SVRPDSP in our benchmark instances for the MVRPDSP all customers have both demands, therefore we assume that each customer i is represented by two different integers, i and $i + n$, respectively for the demand and pickup service, where n is the number of customers. Figure 4.1 depicts an example of a solution for the MVRPDSP and how it is represented.

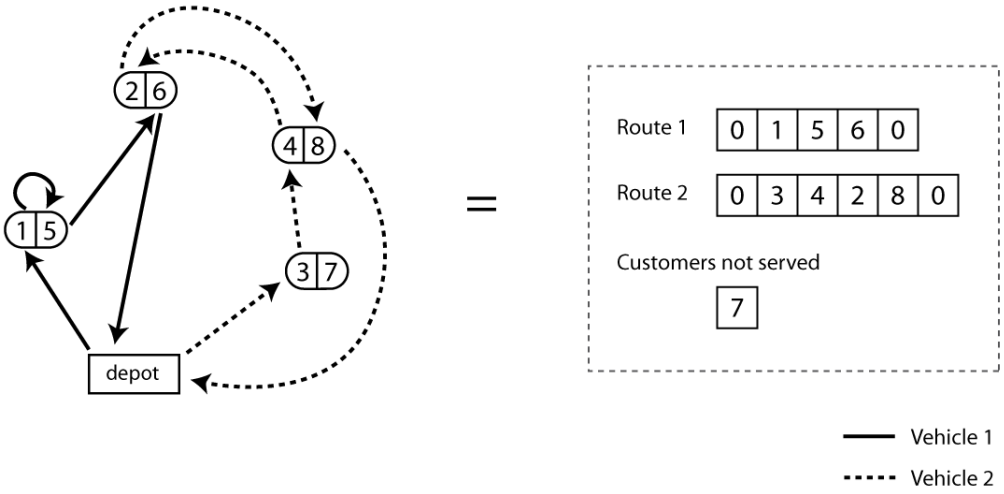


Figure 4.1: Example of solution for the MVRPDSP and its representation.

4.2 Constructive Heuristic

The proposed hybrid constructive heuristic is a cluster-first heuristic. Thus it can be separated into two distinct phases: clustering and routing. The former is responsible for clustering the customers, while the later route the customers within each cluster. In order to generate good quality solutions we propose solving the clustering phase by using an adapted Capacitated p-Median mathematical model and the routing phase by using the constructive heuristic *tspKnapsackBased*, the same presented in section 3.2 for the SVRPDSP. It is important to emphasize that, in case a customer has both demands, it is split into two different customers, one with each demand and the cost to travel between the two is set as zero. Figure 4.2 shows a framework of the heuristic. Each component is detailed in the following.

4.2.1 Clustering phase

This phase is responsible for clustering the customers minimizing the distance between the customers within the clusters. For that end we use a Capacitated p-Median formulation, which will define the clusters. The concept of facility in such model is only used as a base to generate the clusters minimizing the distance between the customers that are part of each cluster. Furthermore, it is not guaranteed that the optimal solution for this formulation will group the customers in the same way they appear in the optimal solution of the MVRPDSP.

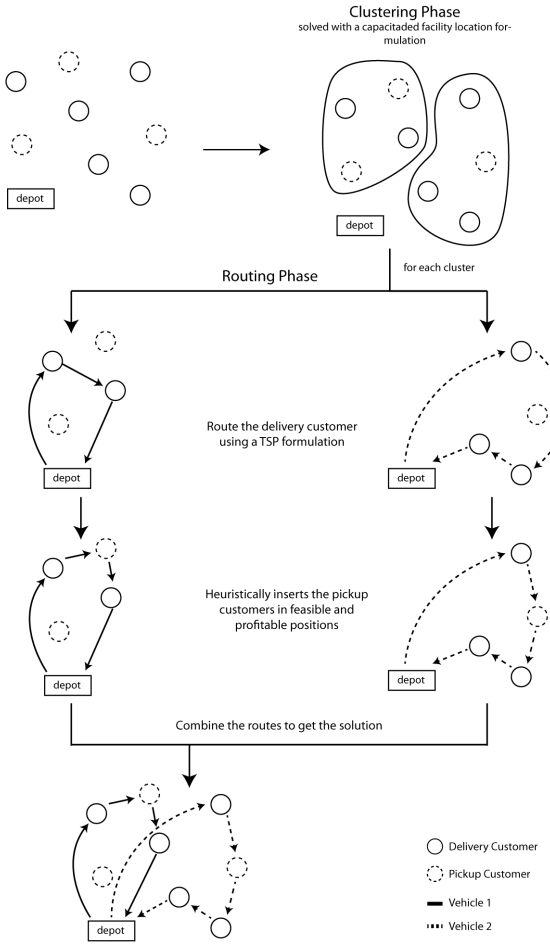


Figure 4.2: Example of how the constructive heuristic for the MVRPDSP works

We use the following Capacitated p-Median formulation in order to define the clusters:

$$Min \sum_{i \in V_d} \sum_{j \in V_d} c_{ij} x_{ij} \tag{4.1}$$

Subject to:

$$\sum_{i \in V_d} x_{ij} = 1 \qquad \forall j \in V_d \tag{4.2}$$

$$\sum_{j \in V_d} d_j x_{ij} \leq Q y_i \qquad \forall i \in V_d \tag{4.3}$$

$$\sum_{i \in V_d} y_i = |K| \tag{4.4}$$

$$x_{ij}, y_i = \{0, 1\} \qquad \forall i, j \in V_d \tag{4.5}$$

Let $x_{ij} = 1$ if customer j is served by a facility opened at customer i and 0

otherwise. Also consider $y_i = 1$ if a facility is opened at customer i and 0 otherwise. Constraints (4.2) specify that every customer must be served by one facility. Of course the capacity of each facility, which is the vehicle capacity Q, cannot be exceed (4.3), which will guarantee that there is a way to route the customers, within a cluster without overweighing the vehicle. Constraints (4.4) specify that the number of clusters in the solution will be exactly the same amount of vehicles in the instance. Note that this formulation only considers delivery customers, since their demand is mandatory, thus we must guarantee that they will be served. This also speed up the runtime needed to solve the model. In addition, as our test instances contains only customers with both demands and considering that they will be split into two different customers we simply assign pickup customers to the cluster where their respective delivery customers were assigned.

Notice that in a certain degree this procedure can be considered heuristic, since its not guaranteed that this formulation will generate the optimal cluster formation and it does not generate a solution for the problem, only the cluster formations. Our intention in using an exact approach in this phase is mostly to find a feasible solution, since extra effort to generate a feasible solution or to repair it is not necessary. Nevertheless, this process apparently generates good cluster formations. Of course that the downside of such approach can be the runtime needed to optimally solve the model for bigger instances. When the runtime proves to be an issue a feasible solution is acceptable and a time limit for the model can be assigned.

4.2.2 Routing phase

Once the clusters have been formed, we must route their customers in order to generate a solution for the MVRPDSP. The routing procedure is based on the one presented in section 3.2 for the constructive *tspKnapsackBased*. In order to take advantage of this heuristic in our problem, we consider each cluster as an instance of the SVRPDSP, which is obviously valid. Therefore, we use the constructive heuristic $tspKnapsackBased$ of the SVRPDSP, explained in section 3.2 to route each cluster. This constructive was chosen since it has proven to be the best among the four presented.

Applying the described routing procedure to each cluster will generate a set of routes, which is a solution for the MVRPDSP.

4.3 MILP formulations

Since the MVRPDSP is a novel problem, in this section we are going to propose two MILP formulations. The first one is based on the formulation proposed by Süral and Bookbinder for the SVRPDSP in [30]. The second one, is an adaptation of our four-commodity network flow formulation for the SVRPDSP, presented in section 3.8. We have not adapted the formulation proposed by Gribkovskaia, Laporte and Shyshou in [14] due to the exponential number of subtour elimination constraints, which greatly impact in the efficiency of the model.

4.3.1 Süral-Bookbinder based formulation

We adapted the formulation proposed by Süral and Bookbinder [30] for the SVRPDSP to the MVRPDSP. This formulation uses three sets of decision variables. Let $x_{ij}^k = 1$ if customer i immediately precedes customer j $(i \neq j)$ in route k and 0 otherwise. As in [30], in order to define the subtour elimination constraints, we define y_i as the total delivery load of a vehicle when departing from customer i. Similarly, we define z_i as the total pickup load of a vehicle when arriving at customer i. Notice that there is no reference of which vehicle these variables are related to. This is due to the fact that at most one vehicle will visit a given customer, therefore there is no need to directly relate these variables to a specific vehicle. Then our mathematical model is the following:

Süral-Bookbinder based formulation for the MVRPDSP

$$Min \sum_{i\in V}\sum_{j\in V}\sum_{k\in K} c_{ij}x_{ij}^k - \sum_{i\in V}\sum_{j\in V_p}\sum_{k\in K} r_j x_{ij}^k \tag{4.6}$$

Subject to:

$$\sum_{k\in K}\sum_{i\in V} x_{ij}^k = 1 \qquad \forall j \in V_d \tag{4.7}$$

$$\sum_{i\in V}\sum_{k\in K} x_{ij}^k \leq 1 \qquad \forall j \in V_p \tag{4.8}$$

$$\sum_{j\in V\setminus\{0\}} x_{0j}^k \leq 1 \qquad \forall k \in K \tag{4.9}$$

$$\sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{ji}^k = 0 \qquad \forall i \in V, k \in K \tag{4.10}$$

$$y_i - y_j \geq d_j - D(1 - \sum_{k\in K} x_{ij}^k) \qquad \forall i,j \in V \setminus \{0\} \tag{4.11}$$

$$z_j - z_i \geq p_i - Q(1 - \sum_{k\in K} x_{ij}^k) \qquad \forall i,j \in V \setminus \{0\} \tag{4.12}$$

$$(y_i + d_i) + (z_i + p_i) \leq Q \qquad \forall i \in V \setminus \{0\} \tag{4.13}$$

$$x_{ij}^k = \{0,1\} \qquad \forall i,j \in V, k \in K \tag{4.14}$$

$$y_i, z_i \geq 0 \qquad \forall i,j \in V \tag{4.15}$$

The objective is to minimize the function (4.6), that represents the total cost of the solution, which is the sum of the routing cost of each route minus the revenue generated by all pickups collected. Constraints (4.7) specify that all delivery customers must be visited exactly once and by only one vehicle. The set of constraints (4.8) define that pickup customers can be visited at most once. Constraints (4.9) define that at most $|K|$ vehicles depart from the depot, limiting each vehicle $k \in K$ to depart from depot only once. Equations (4.10) are flow conservation constraints, which specify that each arrival forces a departure of the same vehicle. The inequalities (4.11) and (4.12) are adaptations of the *MTZ* subtour elimination constraints proposed in [19] and adapted to the SVRPDSP. The capacity of a vehicle must not be exceeded at any time (4.13). Finally (4.14) and (4.15) are binary non-negativity restrictions.

Since this formulation is based on the model of Süral-Bookbinder [30] we adapted all the improvements presented in their paper which are presented in the following.

Lifted MTZ subtour elimination constraints:

$$y_i - y_j + [min\{Q - d_i - d_j, Q - p_j - d_i\}]^+ \sum_{k \in K} x_{ji}^k \geq d_j - \min\{D, Q - p_j\}(1 - \sum_{k \in K} x_{ij}^k) \quad \forall i, j \in V \setminus \{0\} \quad (4.11.1)$$

$$z_j - z_i + [min\{P, Q - d_i\} - p_i - p_j]^+ \sum_{k \in K} x_{ji}^k \geq p_i - \min\{P, Q - d_i\}(1 - \sum_{k \in K} x_{ji}^k) \quad \forall i, j \in V \setminus \{0\} \quad (4.12.1)$$

Lifted capacity constraints using logical inequalities:

$$\sum_{k \in K} \sum_{i} x_{ij}^k - q_j = 0 \qquad \forall j \in V \qquad (4.16)$$

$$y_j + z_j + \sum_{i \in V_d} \sum_{k \in K} d_i x_{ij}^k \leq Q - d_j \qquad \forall j \in V_d \qquad (4.13.3)$$

$$y_i + z_i + \sum_{j \in V_p} \sum_{k \in K} p_j x_{ij}^k \leq (Q - p_i) q_i \qquad \forall i \in V_p \qquad (4.13.4)$$

Knapsack constraints (KC):

$$\sum_{i \in V} \sum_{j \in V_p} p_j x_{ij}^k \leq Q \qquad \forall k \in K \qquad (4.17)$$

$$(4.18)$$

Constraints (4.11.1) and (4.12.1) are lifted *MTZ* subtour elimination constraints. The sets of constraints (4.16), (4.13.3) and (4.13.4) are lifted logical inequalities. Constraints (4.17) are inequalities adapted from the knapsack problem.

In addition to these adapted improvements we also propose another two to remove symmetry from the problem. They are presented in the following.

4.3.1.1 Anti-symmetry constraints

If a customer i has delivery and pickup demands it is split in i (delivery) and $i + n$ (pickup). If both demands are to be served simultaneously, the vertices are visited consecutively in the route, no matter in which order. In our test problems all customers have both demands, so the number of solutions with this symmetry is exponential. To avoid this symmetry, the set of anti-symmetry constraints (3.24) proposed for our four-commodity network flow formulation for the SVRPDSP can be adapted to the MVRPDSP:

$$x_{i+n,i}^k = 0 \qquad \forall i \in V_d, k \in K \qquad (4.19)$$

Similarly, there is symmetry regarding the order of the routes in a solution. Consider an instance with $|K| = 2$, $V_d = \{1, 2, 3\}$, $V_d = \{4, 5, 6\}$ and assume that the set of routes $\{\{0, 1, 4, 2, 0\}, \{0, 3, 6, 0\}\}$ is a feasible solution for such instance. Clearly, the

solution $\{\{0,3,6,0\},\{0,1,4,2,0\}\}$ is also feasible, and in practice they are the same, as the fleet is homogeneous. However, the model considers them as different solutions. Therefore, we propose the following constraints, which avoids this symmetry by forcing the routes to be ordered by the first vertex of the route in increasing order. For instance, in our example, by adding these constraints, the solution $\{\{0,1,4,2,0\},\{0,3,6,0\}\}$ would still be feasible, however $\{\{0,3,6,0\},\{0,1,4,2,0\}\}$ would be considered unfeasible.

$$x_{0,j}^{k} \leq \sum_{i=0}^{j-1} x_{0,i}^{k-1} \qquad \forall j \in V_d, k \in K \backslash \{0\} \tag{4.20}$$

4.3.2 Four-commodity network flow formulation

We adapted our four-commodity network flow formulation, originally proposed for the SVRPDSP, to the MVRPDSP. In this formulation let $x_{ij}^{k} = 1$ if customer i immediately precedes customer j $(i \neq j)$ in route k and 0 otherwise. As in the single vehicle version, let f_{ij}^{d} and g_{ji}^{d} be the delivery load and free space, respectively, when travelling through arc (i,j), and f_{ij}^{p} and g_{ji}^{p} be the pickup load and free space, respectively. Therefore, the four-commodity network flow formulation for the MVRPDSP is as in the following.

The four-commodity network flow based formulation for the MVRPDSP

$$Min \sum_{(i,j)\in V}\sum_{k\in K} c_{ij}x_{ij}^k - \sum_{(i,j)\in V}\sum_{k\in K} r_j x_{ij}^k \tag{4.21}$$

Subject to:

$$\sum_{i\in V}\sum_{k\in K} x_{ij}^k = 1, \qquad \forall j \in V_d \tag{4.22}$$

$$\sum_{i\in V}\sum_{k\in K} x_{ij}^k \leq 1, \qquad \forall j \in V_p \tag{4.23}$$

$$\sum_{j\in V} x_{ij}^k - x_{ji}^k = 0, \qquad \forall i \in V; k \in K \tag{4.24}$$

$$\sum_{j\in V} x_{0j}^k \leq 1 \qquad \forall k \in K \tag{4.25}$$

$$f_{ij}^d + f_{ij}^p \leq Q \sum_{k\in K} x_{ij}^k, \qquad \forall i,j \in V; i \neq j \tag{4.26}$$

$$f_{ij}^d + g_{ji}^d = Q \sum_{k\in K} x_{ij}^k, \qquad \forall i,j \in V; i \neq j \tag{4.27}$$

$$f_{ij}^p + g_{ji}^p = Q \sum_{k\in K} x_{ij}^k, \qquad \forall i,j \in V; i \neq j \tag{4.28}$$

$$\sum_{j\in V}\left(f_{ji}^d - f_{ij}^d\right) + \sum_{j\in V}\left(g_{ji}^d - g_{ij}^d\right) = 2d_i, \qquad \forall i \in V_d \tag{4.29}$$

$$\sum_{j\in V}\left(f_{ji}^p - f_{ij}^p\right) + \sum_{j\in V}\left(g_{ji}^p - g_{ij}^p\right) = 2p_i \sum_{j\in V}\sum_{k\in K} x_{ij}^k, \qquad \forall i \in V_p \tag{4.30}$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall i,j \in V; k \in K \tag{4.31}$$

$$f_{ij}^d, f_{ij}^p, g_{ji}^d, g_{ji}^p \geq 0, \qquad \forall i,j \in V \tag{4.32}$$

The objective function (4.21) minimizes the total routing cost. Delivery customers must be visited exactly once (4.22) and pickup customers may be visited at most once (4.23). The set of constraints (4.24) specify that each arrival produces a departure of the same vehicle. The set of equations (4.25) define that each of the $|K|$ vehicles may leave the depot only once, thus limiting the number of vehicles. Constraints (4.26) limits the combined amount of delivery and pickup load on each arc: the vehicle capacity if the arc is used, and zero otherwise. Constraints (4.27) and (4.28) define g's variables in terms of f's variables. Inequalities (4.29) and (4.30) assures Lemma 3.8.1 respectively on delivery and pickup customers. For pickup customers the lemma holds only for the ones visited; for the non visited the load flow is zero. Finally, constraints (4.31) and (4.32) define the variables. As in the formulation for the SVRPDSP, subtour elimination is implicitly considered in the load flow variables f and g.

In addition we adapted the improvements of the formulation for the single vehicle

version, except the improvement named *Additional constraints*, since they assume that f^d only decreases and f^p only increases along the route. As the MVRPDSP does not have a single route, this property is not necessarily valid.

The adapted improvements are presented in the following.

Lifted capacity constraints (LC)

$$f_{ij}^d + f_{ij}^p \leq (Q - max(d_i, p_j)) \sum_{k \in K} x_{ij}^k, \qquad \forall i, j \in V \qquad (4.26.1)$$

Constraints (4.26.1) can be easily proven to be valid for the MVRPDSP, since the sets of variables f^d and f^p are the same as in the SVRPDSP, and the term $\sum_{k \in K} x_{ij}^k$ indicates whether arc (i, j) is in the route or not, as x_{ij} do in the SVRPDSP.

Knapsack constraints (KC)

$$\sum_{i \in V \setminus \{0\}} \sum_{j \in V_p} p_i x_{ij}^k \leq Q \qquad \forall k \in K \qquad (4.33)$$

In the case of the Knapsack constraints we opted to remove the former set of variables q, since we must constrain the maximum possible pickup load per route.

Besides these improvements we also included the anti-symmetry constraints presented in section 4.3.1 for our first formulation of the MVRPDSP, which are shown in the following.

Anti-symmetry constraints (AS)

$$x_{i+n,i}^k = 0 \qquad \forall i \in V_d; k \in K \qquad (4.34)$$

$$x_{0,j}^k \leq \sum_{i=0}^{j-1} x_{0,i}^{k-1} \qquad \forall j \in V_d, k \in K \setminus \{0\} \qquad (4.35)$$

Chapter 5

Computational Tests

In this chapter the computational results of our approaches for both problems are presented, comparing them with the results of other approaches from the literature. We begin by describing the set of instances used for each problem. Further, before presenting the computational results of our approaches, we describe how the parameter calibration of our algorithms was performed. For that end, we present statistical analyses to support our decisions. Finally, we present the results obtained by our approaches and compare them with the others from the literature.

5.1 Benchmark instances

The set of benchmark instances used to test the SVRPDSP approaches are the ones proposed by Gribkovskaia, Laporte and Shyshou in [14]. This set was derived from 17 instances of the Capacitated Vehicle Routing Problem (CVRP) available at the VPRLIB [31]. Since the CRVP instances only have delivery customers they generated the pickup demands based on the delivery demands. For the revenues they generated values proportional to the average cost of the instance, multiplied by a factor ω. Four values were used for ω (0.2, 0.5, 1.0 and 2.0), therefore a total of 68 instances were generated for the SVRPDSP. The capacity of the vehicle is equal to the sum of the delivery demands. These instances are named in the format `num_B_type` where `num` is the number of customers plus the depot and `type` is the ω used when generating the instances, named `p_two`, `half`, `one` and `two` respectively for 0.2, 0.5, 1.0 and 2.0. The number of customers in these instances ranges from 15 to 110 customers, however, since all customers have both demands in these instances, in all methods they are, in practice, split into two different customers, thus the minimum number of customers of these instances is actually 30 and the maximum 220.

Since the MVRPDSP is a novel problem, there is no set of instances in the literature. Therefore, we decided to adapted some benchmark instances from the SVRPDSP to the multiple vehicle version. Let $D = \sum_{i \in V_d} d_i$. In order to suit these instances to this problem we assigned the capacity of each vehicle as $\lfloor \alpha D \rfloor$, where $\alpha = \{0.2, 0.4, 0.6\}$. Moreover, in real world contexts, the number of vehicles is often a critical factor. It is much more expensive to acquire a new vehicle than to perform the deliveries and pick-ups with fewer vehicles and with a higher cost. Thus, we decided to use as few vehicles as possible. Therefore, for each instance we calculate the minimum number of vehicles needed to perform all delivery demands. This is the number of bins in the optimal solution of the well known Bin Packing Problem considering the delivery customers as the item set and the vehicle capacity as the capacity of each bin. We selected the instances of size 16, 26, 36, 45, 51 and 72 to transform into instances of the MVRPDSP, generating a total of 72 instances. These instances are named `NNN_B_TYPE_C_ALFA` where `NNN_B_TYPE` is the original name of the instance for the SVRPDSP and `ALFA` refers to the α used to set the capacity of each vehicle in our multiple vehicle problem.

5.2 Parameter calibration

We are going to present the parameter calibration of our approaches following the order in which they were described, therefore we begin with the SVRPDSP algorithms.

As stated in section 3.2, the constructives *tspBased* and *tspKnapsackBased* have a fixed-size RCL, with a parameter called $rclSize$, which defines the size of the RCL. By performing preliminary tests we noticed that the greedier the heuristic, the best. Thus we decided to consider only the set of values $\{1, 2, 3\}$ to calibrate this parameter. For a total of 52 instances we run each constructive heuristic five times for each $rclSize$ value, except for the $rclSize = 1$, which was run only once as it is totally greed.

Consider $best_i^j$ as the best solution value found in 5 runs of the constructive j, where $j = \{t(tspbased), tk(tspKnapsackbased)\}$, and with $rclSize = i$ for a given instance. In addition consider $best = min(best_i^j)$ and $worst = max(best_i^j)$ $\forall i \in \{1, 2, 3\}; j \in \{t, tk\}$. Then, we normalized the best values using the following formula resulting in values in the interval $0 : 1$.

$$norm_i^j = (best_i^j - best)/(worst - best + \epsilon^1) \tag{5.1}$$

Having these values normalized we can perform an ANOVA (ANalysis Of VAriance) to statistically analyse which parameter value is the best one. The software

[1] $\epsilon = 10^{-9}$, for preventing division by zero

Minitab 16 was used for that end. The result of this analysis can be seen in Figure 5.1. According to the results none of the configurations is statistically the best with 95% of confidence, since all intervals intercept at least one of the others. However, the configurations $tspKnapsackBased(rclSize = 1)$ or $tspKnapsackBased(rclSize = 2)$ are clearly better than the ones of the constructive *tspBased*. Since the former is greedy and it only needs to be run once, the best choice could be using both configurations.

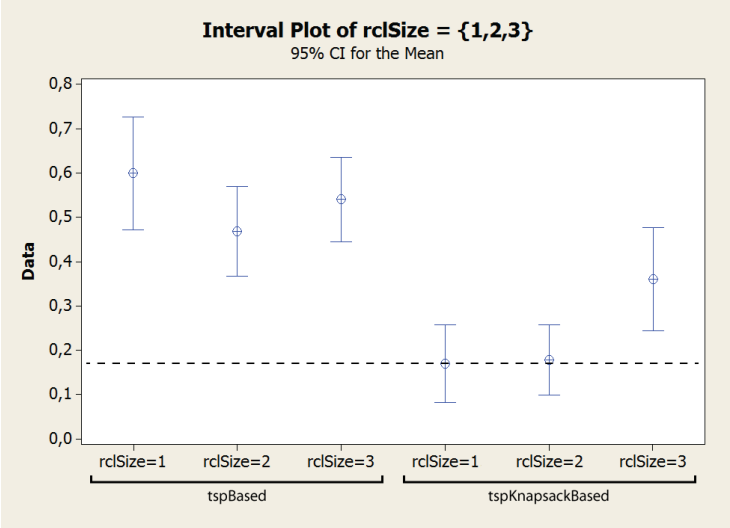


Figure 5.1: ANOVA of the constructive heuristics tspBased and tspKnapsackBased

Despite these results we noticed that, although the configuration $tspBased(rclSize = 1)$ seems to be the worst of them, for some instances it has found values much better than the other ones. To support our claim, we present a graph, in Figure 5.2, showing the *gap* values for the 52 instances tested. Notice that these *gap* values were calculated by the following equation 5.2 and that LB is the theoretical lower bound proposed by [14], which is calculated by subtracting the value of the optimal solution of the TSP (for the delivery customers) by the value of the optimal solution of the Knapsack Problem (for the pickup customers). This lower bound is valid since the optimal solution of the TSP will route the delivery customers in the best possible way and the Knapsack Problem will select the best set of pickups according to the revenue they generate. According to Figure 5.2, the $tspBased(rclSize = 1)$ has found better solutions for 12 instances out of the 52 instances tested, including `023_B_p_two`, in which the difference is very high.

$$gap = \left| \frac{solution - LB}{LB} \right| \times 100 \tag{5.2}$$

Notice that, in Figure 5.2, in order to simplify, we only present the results of the

best configurations and the $tspBased(rclSize = 1)$. Since this configuration is greedy and the impact in the time efficiency is very low, we decided to use this configuration in addition to the other two previously mentioned. Therefore in our approches we use the configurations $tspBased(rclSize = 1)$, $tspKnapsackBased(rclSize = 1)$ and $tspKnapsackBased(rclSize = 2)$

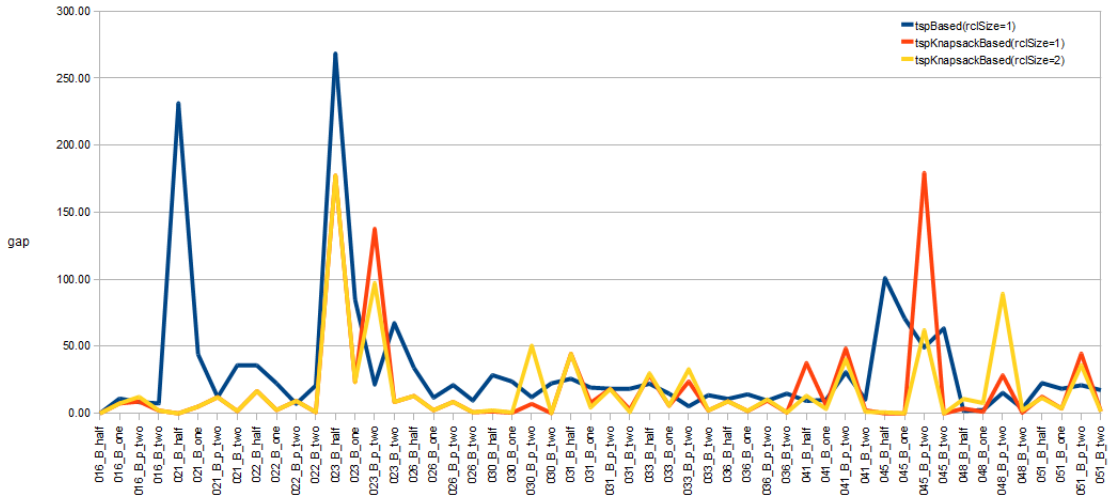


Figure 5.2: Graph showing the *gap* values for the constructive heuristics $tspBased(rclSize = 1)$ and $tspKnapsackBased(rclSize = \{1, 2\})$

We performed a similar analysis on the parameters of the EA, which are the population size (*popSize*) and the number of iterations (*numIt*). However, instead of normalizing the values we used the *gap* values, since for the set of instances tested the *gap* values were in about the same scale. For a total of 16 instances, we tested the set of values $\{10, 20, 30\}$ for each parameter running 5 times each configuration for each instance. Figure 5.3 shows the results of the ANOVA performed using the *gap* values.

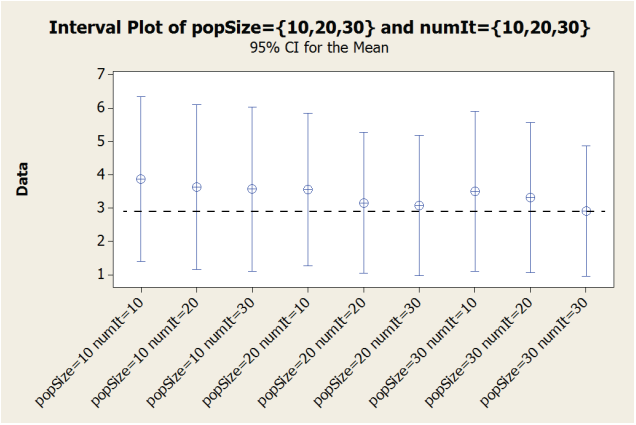


Figure 5.3: ANOVA of the EA parameters

According to the results of the ANOVA, the algorithm is robust and changing its parameters does not have a significant impact in the quality of the solutions generated. Therefore, we decided to choose a configuration that balance the quality and the running time. Although the configuration $popSize = 30, numIt = 30$ has the best average, we chose the configuration $popSize = 20, numIt = 20$ which has almost the same average, but runs faster.

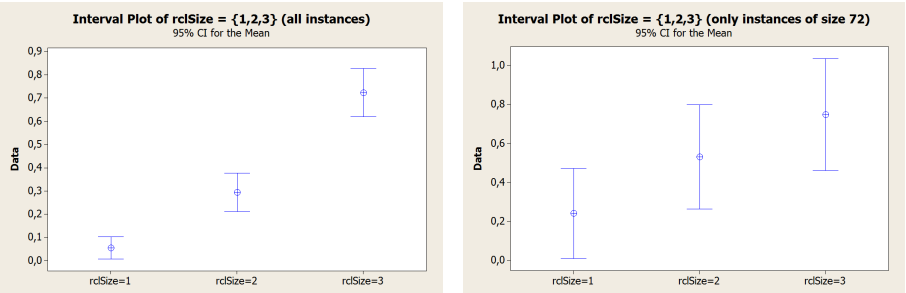
Regarding the VND, since it has no parameters, there is no analysis to be performed.

The only approach for the MVRPDSP in which we must perform an analysis to calibrate its parameters is the hybrid constructive heuristic. In its routing phase the heuristic $tspKnapsackBased$ from the SVRPDSP is used. Notice that this heuristic has a parameter $rclSize$ and as in the SVRPDSP it must be calibrated. For that end we ran 30 times the hybrid constructive for all instances using each $rclSize$ value from the set of values $\{1, 2, 3\}$. The best solution values were normalized and an ANOVA was performed on these values. The results of such analysis are shown in Figure 5.4a. Clearly the $rclSize = 1$ is the best with 95% of confidence. However, since for the SVRPDSP the $rclSize = 2$ proved to be also a good choice, we decided to perform an ANOVA only considering our biggest instances, with size 72. The results can be seen in Figure 5.4b and shows that the difference between the $rclSize = 1$ and $rclSize = 2$ is much smaller and the $rclSize = 1$ is no longer proved to be the best. This behavior could be due to the fact that in bigger instances the solutions tend to have more customers per route, which greatly increases the number of possibilities to insert the pickup customers in the routing phase. Therefore, we foresee that for instances with a greater number of customers the value $rclSize = 2$ may be equally a good choice or even a better one. Nevertheless, since the constructive is not time consuming we decided to use both, $rclSize = 1$ and $rclSize = 2$, then choose the best solution generated.

5.3 Experimental results for the SVRPDSP

In this section we present the experimental results of the SVRPDSP approaches comparing them with the results of the literature.

All results were obtained in a $Intel(R)Core^{TM}$ i7 3.07GHz machine with 6Gb RAM, running the operating system Ubuntu 12.04. All codes were written in C++ and the mathematical formulations were solved using the software *IBM ILOG CPLEX 12.3* under an academic license.



(a) ANOVA for all instances (b) ANOVA for instances of size 72

Figure 5.4: Graphs for the Analysis of Variance of the $rclSize$ of the hybrid constructive for the MVRPDSP

In the following subsections we present the computational results of our metaheuristic approaches and MILP formulations, respectively.

5.3.1 Metaheuristics

At first we ran the EA and VND 10 times for each of the 68 benchmark instances described in section 5.1 with a time limit of 3 hours. Notice that this time limit is only reached by our biggest instances, with number of customers up to 72. Individual results for the best and average solutions found by these algorithm are shown in Tables 5.1 and 5.2, respectively. These results are compared to the results of the preliminary version of our EA presented in [5] and to the GVNS proposed in [6]. Values highlighted in bold face define the best value for the instance. Solutions whose objective value are equal to the lower bound are proved optimal, and are marked with '*'. Noticed that we do not present the results of the Tabu Search proposed by Gribkovskaia, Laporte and Shyshou in [14]. This is due to the lack of individual results in their paper, only average results per instance type are shown.

According to Table 5.1 for more than half of the instances the EA has found better or equal solutions than the other algorithms of the literature, including 15 new solutions, 7 proved optimal solutions, including the 3 found by the GVNS. Although the VND is not a match for the EA nor for the GVNS it has found better or equal solution than the other algorithm for about 1/3 of the instances, including 5 of the optimal solutions found by the EA. These results also show that our EA is substantially better than the preliminary implementation yielding much better results for most instances. The fact that in the cases of instances with sizes 76, 101 and 111 the EA has found poorer results, could be due to the time limit set for the algorithm, which was not set in the previous EA, neither in the GVNS.

Table 5.1: Best solutions of our approaches compared to the ones from the literature

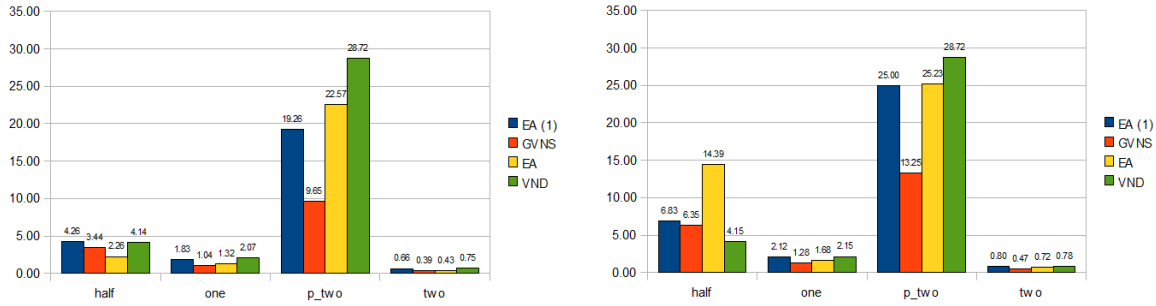
Instance	EA [5]	GVNS [6]	EA	VND	Lower Bound
016_B_half	**36.60***	39.86	**36.60***	**36.60***	36.60
016_B_one	**-150.73**	**-150.73**	**-150.73**	-145.44	-155.41
016_B_p_two	135.11	**132.28**	133.8	142.03	130.99
016_B_two	**-536.13**	**-536.13**	-536.06	-530.84	-540.81
021_B_half	**-20.16***	-18.78	**-20.16***	**-20.16***	-20.16
021_B_one	-301.93	**-307.80**	**-307.80**	-300	-316.09
021_B_p_two	146.75	**137.59**	**137.59**	147.03	132.21
021_B_two	**-901.28**	**-901.28**	**-901.28**	-893.48	-909.57
022_B_half	**-62.63**	**-62.63**	-62.4	-56.86	-64.97
022_B_one	**-421.04**	**-421.04**	**-421.04**	**-421.04**	-429.15
022_B_p_two	124.25	**123.59**	124.25	124.33	116.01
022_B_two	**-1,149.38**	**-1,149.38**	**-1,149.38**	**-1,149.38**	-1,157.49
023_B_half	**-80.95**	**-80.95**	-80.01	-61.67	-94.06
023_B_one	**-698.35**	**-698.35**	-697.41	-640.23	-711.46
023_B_p_two	274.31	269.86	**269.08**	291.07	260.88
023_B_two	**-1,933.10**	**-1,933.10**	-1932.16	-1874.98	-1,946.21
026_B_half	-91.95	-87.87	**-92.41**	**-92.41**	-92.47
026_B_one	**-497.17**	**-497.17**	**-497.17**	-495.87	-504.40
026_B_p_two	146.51	146.9	**146.11**	146.34	139.67
026_B_two	**-1,334.26**	**-1,334.26**	**-1,334.26**	-1332.96	-1,341.49
030_B_half	-357.21	**-378.64**	**-378.64**	**-378.64**	-382.80
030_B_one	-1120.33	**-1,152.07**	**-1,152.07**	**-1,152.07**	-1,156.23
030_B_p_two	**82.29**	**82.29**	**82.29**	85.49	81.33
030_B_two	-2667.25	**-2,699.00**	**-2,699.00**	**-2,699.00**	-2,703.16
031_B_half	-85.56	-87.06	**-89.15**	-85.66	-91.79
031_B_one	-505.46	**-511.07**	**-511.07**	-507.82	-514.05
031_B_p_two	**123.15**	**123.15**	**123.15**	131.07	115.52
031_B_two	-1350.61	-1354.49	**-1,355.59**	-1352.34	-1,358.57
033_B_half	-137.29	**-150.73**	**-150.73**	**-150.73**	-157.09
033_B_one	-759.25	**-765.79**	**-765.79**	-753.55	-778.21
033_B_p_two	198.44	199.17	**194.61**	198.42	188.44
033_B_two	-2001.44	**-2,007.98**	-2007.28	-1995.74	-2,020.40
036_B_half	-113.44	-127.06	**-128.26**	**-128.26**	-128.53
036_B_one	-608.73	-616.12	**-624.41**	**-624.41**	-624.67
036_B_p_two	139.71	**130.42**	132.6	133.56	121.94
036_B_two	-1596.84	-1608.35	**-1,616.64**	**-1,616.64**	-1,616.90
041_B_half	-184.07	-183.86	**-185.75**	-181.85	-186.35
041_B_one	-755.69	**-760.52**	-758.06	-757.26	-767.97
041_B_p_two	111.66	**109.48**	111.69	131.86	100.89
041_B_two	-1922.74	**-1,923.86**	-1922.71	-1920.6	-1,931.31
045_B_half	-489.47	**-491.15***	**-491.15***	**-491.15***	-491.15
045_B_one	-1647.59	**-1,648.51***	**-1,648.51***	**-1,648.51***	-1,648.51
045_B_p_two	202.70	199.82	**198.04***	201.22	198.04
045_B_two	**-3,963.32***	**-3,963.32***	**-3,963.32***	**-3,963.32***	-3,963.32
048_B_half	-36786.80	-37200.62	-36786.8	**-37,205.50**	-37,753.00
048_B_one	-105863.00	-107058.78	**-107,059.00***	-106423	-107,059.00
048_B_p_two	-3830.83	**-4,244.69**	-3830.83	-4140.1	-4,797.03
048_B_two	-247332.00	**-247,946.25**	-247865	-247663	-248,298.00
051_B_half	-308.53	**-310.24**	-308.53	-305.94	-320.61
051_B_one	-1083.76	-1086.52	**-1,088.26**	-1077.35	-1,098.86
051_B_p_two	135.34	126.7	**123.81**	131.14	116.58
051_B_two	-2640.20	**-2,645.35**	-2643.46	-2633.79	-2,655.30
072_B_half	-388.84	-406.57	**-408.09**	-406.87	-409.78
072_B_one	-998.14	-1024.04	-984.2	**-1,024.33**	-1,027.24
072_B_p_two	-19.05	**-36.03**	-33.43	-30.16	-39.24
072_B_two	-2232.95	-2259.01	-2247.83	**-2,259.30**	-2,262.21
076_B_half	-553.10	**-579.25**	-553.1	-576.66	-579.52
076_B_one	-1741.31	**-1,758.92**	-1731.91	-1752.71	-1,759.19
076_B_p_two	34.04	**23.62**	51.95	57.55	14.33
076_B_two	-4099.12	**-4,118.23**	-4102.38	-4112.02	-4,118.50
101_B_half	**-911.68**	-906.79	-909.32	-910.73	-922.71
101_B_one	-2538.99	**-2,541.35**	-2538.99	-2538.99	-2,552.38
101_B_p_two	-47.99	**-53.89**	-49.55	-49.55	-66.59
101_B_two	**-5,800.66**	**-5,800.66**	-5798.3	-5798.3	-5,811.69
111_B_half	-1380.63	**-1,384.06**	-1375.41	-1380.63	-1,386.67
111_B_one	-3314.79	**-3,315.78**	-3309.57	-3314.79	-3,320.83
111_B_p_two	-251.14	**-252.84**	-249.66	-249.73	-257.18
111_B_two	-7182.88	**-7,186.31**	-7177.66	-7182.88	-7,188.92

Table 5.2: Average solutions of our approaches compared to the ones from the literature

Instance	EA [5]	GVNS [6]	EA	VND	Lower Bound
016_B_half	**36.60***	40.61	**36.60***	**36.60***	36.60
016_B_one	-150.07	**-150.73**	-148.947	-145.44	-155.41
016_B_p_two	138.66	**133.43**	140.406	142.03	130.99
016_B_two	-534.64	**-536.13**	-532.799	-530.84	-540.81
021_B_half	-17.81	-13.964	**-20.16***	**-20.16***	-20.16
021_B_one	-300.73	**-307.21**	-304.535	-300	-316.09
021_B_p_two	147.55	**141.17**	144.587	147.03	132.21
021_B_two	-896.20	**-900.69**	-895.391	-893.48	-909.57
022_B_half	**-59.45**	-58.591	-57.414	-56.86	-64.97
022_B_one	**-421.04**	**-421.04**	**-421.04**	**-421.04**	-429.15
022_B_p_two	126.56	124.859	**124.29**	124.33	116.01
022_B_two	**-1,149.38**	**-1,149.38**	-1149.37	**-1,149.38**	-1,157.49
023_B_half	-80.46	**-80.76**	-79.829	-61.67	-94.06
023_B_one	-698.05	**-698.26**	-688.259	-640.23	-711.46
023_B_p_two	282.52	**273.23**	273.718	291.07	260.88
023_B_two	-1931.65	**-1,932.91**	-1926.52	-1874.98	-1,946.21
026_B_half	-85.31	-85.526	**-92.41**	**-92.41**	-92.47
026_B_one	-497.04	**-497.17**	-495.909	-495.87	-504.40
026_B_p_two	147.12	147.179	**146.14**	146.34	139.67
026_B_two	**-1,334.26**	**-1,334.26**	-1333.48	-1332.96	-1,341.49
030_B_half	-356.21	-370.694	**-378.64**	**-378.64**	-382.80
030_B_one	-1114.56	-1146.639	**-1,152.07**	**-1,152.07**	-1,156.23
030_B_p_two	86.84	85.694	**82.29**	85.49	81.33
030_B_two	-2662.89	-2697.256	**-2,699.00**	**-2,699.00**	-2,703.16
031_B_half	-80.82	-84.896	**-88.39**	-85.507	-91.79
031_B_one	-502.40	-508.038	**-510.65**	-507.686	-514.05
031_B_p_two	124.71	**123.39**	124.345	131.07	115.52
031_B_two	-1345.89	-1352.73	**-1,354.83**	-1352.21	-1,358.57
033_B_half	-133.96	-146.008	-146.469	**-150.73**	-157.09
033_B_one	-755.00	-761.796	**-765.10**	-753.55	-778.21
033_B_p_two	199.48	202.084	**195.60**	198.42	188.44
033_B_two	-1996.78	-2003.986	**-2,005.94**	-1995.74	-2,020.40
036_B_half	-112.86	-124.648	**-128.26**	**-128.26**	-128.53
036_B_one	-605.02	-610.631	**-624.41**	**-624.41**	-624.67
036_B_p_two	143.81	136.406	**132.93**	133.56	121.94
036_B_two	-1596.75	-1604.649	**-1,616.64**	**-1,616.64**	-1,616.90
041_B_half	-173.69	-179.359	**-185.75**	-181.85	-186.35
041_B_one	-753.35	**-758.60**	-756.292	-749.98	-767.97
041_B_p_two	**112.19**	115.164	112.887	131.86	100.89
041_B_two	-1917.54	**-1,921.50**	-1921.02	-1913.59	-1,931.31
045_B_half	-481.15	-490.524	**-491.15***	**-491.15***	-491.15
045_B_one	-1645.76	-1647.884	-1647.63	**-1,648.51***	-1,648.51
045_B_p_two	208.25	201.85	**198.57**	201.22	198.04
045_B_two	-3960.49	-3962.694	-3962.41	**-3,963.32***	-3,963.32
048_B_half	-36786.80	-37200.62	-36786.8	**-37,205.50**	-37,753.00
048_B_one	-105439.00	-106682.877	**-106,796.00**	-106423	-107,059.00
048_B_p_two	-3457.70	-4122.465	-3829.53	**-4,140.10**	-4,797.03
048_B_two	-247027.00	**-247,871.39**	-247762	-247663	-248,298.00
051_B_half	-306.20	**-307.35**	-306.2033	-305.94	-320.61
051_B_one	-1083.18	-1085.089	**-1,086.76**	-1077.19	-1,098.86
051_B_p_two	140.92	127.992	**127.71**	131.14	116.58
051_B_two	-2639.07	**-2,640.95**	-2636.52	-2633.63	-2,655.30
072_B_half	-383.50	-404.964	-406.355	**-406.87**	-409.78
072_B_one	-994.55	**-1,023.07**	-978.857	-1020.47	-1,027.24
072_B_p_two	-12.38	**-35.85**	-32.3133	-30.16	-39.24
072_B_two	-2228.19	**-2,259.01**	-2197.82	-2255.44	-2,262.21
076_B_half	-548.85	-576.189	-548.8533	**-576.62**	-579.52
076_B_one	-1730.49	**-1,753.27**	-1713.61	-1752.35	-1,759.19
076_B_p_two	39.59	**26.95**	55.3033	57.55	14.33
076_B_two	-4097.67	**-4,113.12**	-4076.64	-4111.66	-4,118.50
101_B_half	-910.00	-900.822	-909.32	**-910.73**	-922.71
101_B_one	**-2,538.99**	-2536.941	**-2,538.99**	**-2,538.99**	-2,552.38
101_B_p_two	-44.59	-47.439	**-49.55**	**-49.55**	-66.59
101_B_two	**-5,798.69**	-5793.329	-5798.3	-5798.3	-5,811.69
111_B_half	**-1,380.63**	-1380.258	-1375.41	**-1,380.63**	-1,386.67
111_B_one	**-3,314.79**	-3312.665	-3309.57	**-3,314.79**	-3,320.83
111_B_p_two	-248.28	**-251.35**	-249.66	-249.73	-257.18
111_B_two	**-7,182.88**	-7181.598	-7177.66	**-7,182.88**	-7,188.92

Table 5.2 presents the average solution values for the algorithms of the literature along with the ones of our approaches. Notice that the EA has the best average values for a total of 28 instances, against 27 of the GVNS and 25 of the VND. We emphasize that for 5 instances the VND has found the optimal solution in each of the 10 runs, against 3 of the EA and none of the GVNS.

Figures 5.5a and 5.5b shows the average *gap* values per instance type, considering the best and the average values, respectively. These *gap* values are related to the theoretical lower bound of the problem. Notice in Figure 5.5a that the EA is clearly better for instances of type $half$. Although for the other types the GVNS has the best average *gap* values, the EA yielded very close averages. It may seem that for instances of type `p_two` the GVNS outmatches our approaches by far. However by analysing the individual results we noticed that the instance `076_B_half` yielded very high gap values for both, the EA and VND. Therefore, in order to better compare the approaches by the average *gap* value per instance type, we have excluded the results of this instance from all approaches. The new values are shown in Table 5.3. According to this table the average of the best *gap* values of the EA is very close to the GVNS, different from what Figure 5.5a seemed to indicate.



(a) Considering the best *gap* values

(b) Considering the average *gap* values

Figure 5.5: Average *gap* values per instance type

Table 5.3: Comparison of the Algorithms with the literature considering the average $gap(\%)$ for instances with type p_two excluding instance 076_B_half

	EA [5]	GVNS [6]	EA	VND
best	11.87	6.20	7.58	11.67
avgerage	15.55	8.58	8.93	11.67

5.3.2 MILP formulations

Regarding the exact approaches, at first we tested the Süral-Bookbinder formulation with the best combination of the improvements, which according to the authors is LS, LI and KC. From now on we are going to refer to this configuration as N. Then we tested the formulation providing it an initial solution (IS). To that end, we ran once the constructive heuristics $tspBased(rclSize = 1)$ and $tspKnapsackBased(rclSize = 1)$, five times the $tspKnapsackBased(rclSize = 2)$ and choose the best solution, providing it to the model as a start point. A time limit of 3 hours was established for each run.

The results for this formulation are presented in Table 5.4. Notice that the last two columns refer to the deviation from each configuration to the best solution found. According to these results the configuration N has proved only one optimal solution and for the majority of the instances the time limit was reached. In addition for 48 out of the 68 instances, N has found poorer results than IS, with an average deviation of 23.24%, which means that IS is the best configuration in average, evidencing its efficiency. In this average results we exclude instance `072_B_p_two` because of its atypical behavior of a very high deviation.

Furthermore, the configurations N and the IS did not find the optimal solution nor have reached the time limit for 15 and 25 instances, respectively. In such cases, a memory overflow error caused the solver to prematurely stop. Although this situation has occurred more for the configuration IS, in 20 of these cases, IS has found a solution better than N.

Regarding the formulation proposed by Gribkovskaia, Laporte and Shyshou, presented in section 2.1.2, one can easily notice that the number of subtour elimination constraints of such formulations is almost 2^n where n is the number of customers, therefore for bigger instances it is impracticable to run the model with all these constraints.

Since the model with all subtour elimination constraints is impracticable for bigger instances, in further analysis we are going to consider only the results obtained using Branch&Cut.

Further we ran the proposed four-commodity network flow formulation for the same set of 68 benchmark instances. To decide which combination of the improvements generates the best results we performed experimental tests with all combinations on 20 instances. These tests have shown that the best configuration is $\{L, AS, KC\}$, which we are going to refer as N from now on. Notice that the chosen configuration does not include the improvement AC. We observed that when these constraints are included into the model the quality of the final solutions are reduced, when not the optimal, and the computational time needed to find the optimal is increased. In addition, we also

tested the model providing an initial solution (IS) using the constructives *tspBased* and *tspKnapsackBased* in the same way as done for the other formulations.

Table 5.5 shows the results of this model. The last two columns refers to the percentage deviation from each configuration to the best solution found. According to the results, the configuration N has proved 31 optimal solutions (those with gap 0) but could not find a single feasible solution for instances with more than 51 customers. However providing the model with an initial solution has shown to improve the time need to prove the same 31 optimal solutions found by the normal version, while also finding 8 new optimal solutions, summing up a total of 39 optimals. The average deviation of N is 4.10% against 0.08% of IS. It is important to notice that although the average of N seems relatively small, we are not including in this value the 21 instances in which this configuration was not able to find a single feasible solution.

In order to compare the solutions of the three models we selected the results of the configurations with initial solution from the Süral-Bookbinder (SB) formulation and from our four-commodity network flow formulation ($4C$). Regarding the formulation Gribkovskaia-Laporte-Shyshou (GLS) we decided to select the results of the Branch&Cut algorithm, since this formulation is impracticable for bigger instances given the exponential number of subtour elimination constraints, as previously stated.

Table 5.7 compare the best solutions found by these formulations considering the aforementioned configurations and the computational time. Solutions highlighted in bold face indicate the best value found among the formulations and solution values followed by $*$ indicates that the model has proved that value as the optimal. The last three columns define the percentage deviation values from the best solution found. Notice that the computational time of $4C$ is much much smaller than the other formulations for most of the instances.

Table 5.4: Results of the Süral-Bookbinder formulation for the SVRPDSP

Instance	normal (N)			Initial solution (IS)			Deviation (%)	
	Solution value	gap	Total time	Solution value	GAP	Total time	N	IS
016_B_half	36.60	12.80	10800	36.60	4.95	10800	0.00	0.00
016_B_one	-150.73	5.43	10800	-150.73	5.19	10800	0.00	0.00
016_B_p_two	**132.28**	0.00	2687	**132.28**	0.00	1248	0.00	0.00
016_B_two	-536.13	1.64	10800	-536.13	1.42	10800	0.00	0.00
021_B_half	-18.63	44.44	10800	-20.16	39.87	7441	7.59	0.00
021_B_one	-299.75	9.24	8215	-307.80	6.80	7777	2.62	0.00
021_B_p_two	137.59	5.92	10800	137.59	5.92	10800	0.00	0.00
021_B_two	-848.74	8.12	5900	-893.48	3.28	5180	5.01	0.00
022_B_half	-56.86	27.43	10800	-62.63	20.51	10800	9.21	0.00
022_B_one	-421.04	3.57	10800	-421.04	3.13	10800	0.00	0.00
022_B_p_two	123.59	18.95	10800	124.33	15.84	9862	0.00	0.60
022_B_two	-1,149.25	1.18	10800	-1,149.38	1.12	10800	0.01	0.00
023_B_half	-80.95	25.17	10800	-80.95	24.12	3905	0.00	0.00
023_B_one	-698.35	3.40	10800	-698.35	3.63	10800	0.00	0.00
023_B_p_two	274.07	17.94	1419	276.60	22.93	1325	0.00	0.92
023_B_two	-1,933.10	1.34	10800	-1,933.10	1.25	10800	0.00	0.00
026_B_half	-85.24	28.50	10800	-92.41	17.42	10800	7.76	0.00
026_B_one	-497.17	7.21	10800	-497.17	6.96	10800	0.00	0.00
026_B_p_two	146.11	21.57	10800	146.90	21.51	5981	0.00	0.54
026_B_two	-1,327.85	3.28	10800	-1,334.26	2.77	9414	0.48	0.00
030_B_half	-357.23	26.27	7382	-378.64	20.87	8574	5.65	0.00
030_B_one	-1,098.86	12.49	8016	-1,152.07	7.58	7689	4.62	0.00
030_B_p_two	107.27	714.82	7932	87.13	628.31	9073	23.11	0.00
030_B_two	-2,650.22	5.47	7612	-2,699.00	3.68	6422	1.81	0.00
031_B_half	-86.00	29.74	10800	-84.16	31.03	9041	0.00	2.14
031_B_one	-505.22	9.79	4695	-506.02	9.46	9030	0.16	0.00
031_B_p_two	123.15	15.80	10800	123.26	14.41	5127	0.00	0.09
031_B_two	-1,245.26	13.19	1070	-1,338.64	6.65	5365	6.98	0.00
033_B_half	-125.69	48.66	10800	-134.42	45.81	4716	6.49	0.00
033_B_one	-716.12	17.50	10800	-742.36	13.81	10800	3.53	0.00
033_B_p_two	203.75	116.75	10800	205.48	79.10	10800	0.00	0.85
033_B_two	-2,001.06	5.20	10800	-1,995.71	5.33	10800	0.00	0.27
036_B_half	-49.11	69.59	965	-120.93	20.39	3373	59.39	0.00
036_B_one	-615.98	5.04	10800	-613.70	4.48	10800	0.00	0.37
036_B_p_two	158.46	49.83	10800	139.59	20.85	8471	13.52	0.00
036_B_two	-1,603.50	2.17	10800	-1,605.92	2.34	907	0.15	0.00
041_B_half	-170.11	27.00	10800	-185.75	19.77	1760	8.42	0.00
041_B_one	-730.16	11.70	10800	-760.52	8.03	10800	3.99	0.00
041_B_p_two	121.17	31.65	10800	123.03	39.41	10800	0.00	1.54
041_B_two	-1,862.40	7.88	10800	-1,916.25	5.10	10800	2.81	0.00
045_B_half	-491.05	28.49	10800	-489.41	29.81	2490	0.00	0.33
045_B_one	-1,639.05	11.09	10800	-1,648.51	10.42	10800	0.57	0.00
045_B_p_two	204.14	1,465.50	10800	200.41	2,207.77	10800	1.86	0.00
045_B_two	-3,719.73	10.43	9208	-3,963.32	4.51	10800	6.15	0.00
048_B_half	-34,990.00	21.69	10800	-36,453.90	19.01	10800	4.02	0.00
048_B_one	-103,635.00	11.98	10800	-106,526.00	9.25	10800	2.71	0.00
048_B_p_two	-3,154.38	72.00	10800	-3,826.48	65.54	10800	17.56	0.00
048_B_two	-242,686.00	7.44	10800	-247,766.00	5.38	10800	2.05	0.00
051_B_half	-289.83	20.28	10800	-310.02	14.74	10800	6.51	0.00
051_B_one	-1,074.46	6.12	10800	-1,080.81	4.80	10800	0.59	0.00
051_B_p_two	141.18	89.61	10800	141.13	86.45	10800	0.04	0.00
051_B_two	-2,637.33	2.34	10800	-2,640.41	2.07	10800	0.12	0.00
072_B_half	-78.09	83.37	10800	-361.00	23.11	9773	78.37	0.00
072_B_one	-220.91	79.69	9231	-989.06	8.99	9844	77.66	0.00
072_B_p_two	214.87	312.72	4308	-0.39	99.61	10800	55,194.87	0.00
072_B_two	-2,042.93	12.02	9753	-2,203.27	5.13	10800	7.28	0.00
076_B_half	-512.69	21.48	10800	-573.05	12.03	10800	10.53	0.00
076_B_one	-1,696.56	8.23	10800	-1,737.32	6.04	3170	2.35	0.00
076_B_p_two	100.81	387.04	10800	41.83	214.28	10800	141.00	0.00
076_B_two	-3,935.32	7.20	10800	-4,101.85	3.24	10800	4.06	0.00
101_B_half	316.19	131.93	10800	-909.32	8.23	10800	134.77	0.00
101_B_one	-2,172.82	17.06	10800	-2,538.99	3.03	10800	14.42	0.00
101_B_p_two	233.89	305.26	10800	-49.55	56.99	10800	572.03	0.00
101_B_two	-5,149.83	12.42	10800	-5,798.30	1.38	10800	11.18	0.00
111_B_half	-459.98	70.74	10800	-1,369.99	12.69	10800	66.42	0.00
111_B_one	-2,318.50	33.86	10800	-3,277.19	6.52	10800	29.25	0.00
111_B_p_two	117.12	126.90	10800	-249.79	42.86	10800	146.89	0.00
111_B_two	-3,942.18	46.54	9918	-7,173.87	2.72	10800	45.05	0.00

Table 5.5: Results of the proposed four-commodity network flow formulation for the SVRPDSP

Instance	normal(N)			Initial solution (IS)			Deviation (%)	
	Value	gap	Total time	Value	gap	Total time	N	IS
016_B_half	**36.60**	5.00	9	**36.60**	2.50	3	0.00	0.00
016_B_one	**-150.73**	0.00	21	**-150.73**	0.00	9	0.00	0.00
016_B_p_two	**132.28**	0.00	15	**132.28**	0.00	9	0.00	0.00
016_B_two	**-536.13**	0.00	19	**-536.13**	0.00	8	0.00	0.00
021_B_half	**-20.16**	0.00	96	**-20.16**	0.00	11	0.00	0.00
021_B_one	**-307.80**	0.00	59	**-307.80**	0.00	22	0.00	0.00
021_B_p_two	**137.59**	0.00	107	**137.59**	0.00	70	0.00	0.00
021_B_two	**-901.28**	0.00	57	**-901.28**	0.00	16	0.00	0.00
022_B_half	**-62.63**	0.00	877	**-62.63**	0.00	192	0.00	0.00
022_B_one	**-421.04**	0.00	170	**-421.04**	0.00	140	0.00	0.00
022_B_p_two	**123.59**	0.00	632	**123.59**	0.00	173	0.00	0.00
022_B_two	**-1,149.38**	0.00	115	**-1,149.38**	0.00	124	0.00	0.00
023_B_half	**-80.95**	0.00	194	**-80.95**	0.00	229	0.00	0.00
023_B_one	**-698.35**	0.00	178	**-698.35**	0.00	167	0.00	0.00
023_B_p_two	**269.02**	0.00	190	**269.02**	0.00	377	0.00	0.00
023_B_two	**-1,933.10**	0.00	106	**-1,933.10**	0.00	159	0.00	0.00
026_B_half	**-92.41**	0.00	562	**-92.41**	0.00	127	0.00	0.00
026_B_one	**-497.17**	0.00	435	**-497.17**	0.00	341	0.00	0.00
026_B_p_two	**145.15**	0.00	751	**145.15**	0.00	206	0.00	0.00
026_B_two	**-1,334.26**	0.00	267	**-1,334.26**	0.00	112	0.00	0.00
030_B_half	-378.64	2.77	10800	-378.64	2.82	10800	0.00	0.00
030_B_one	-1,152.07	1.18	10800	**-1,152.07**	0.00	10579	0.00	0.00
030_B_p_two	82.29	27.51	10800	85.49	59.25	10800	0.00	3.89
030_B_two	-2,699.00	0.40	10800	**-2,699.00**	0.00	9450	0.00	0.00
031_B_half	**-91.20**	0.00	6476	**-91.20**	0.00	2262	0.00	0.00
031_B_one	**-511.07**	0.00	1472	**-511.07**	0.00	367	0.00	0.00
031_B_p_two	**121.17**	0.00	1814	**121.17**	0.00	436	0.00	0.00
031_B_two	**-1,355.59**	0.00	507	**-1,355.59**	0.00	893	0.00	0.00
033_B_half	**-150.73**	0.00	6900	**-150.73**	0.00	8588	0.00	0.00
033_B_one	-764.73	1.13	10800	**-765.79**	0.00	9011	0.14	0.00
033_B_p_two	194.61	3.24	10800	194.76	5.37	10800	0.00	0.08
033_B_two	-2,006.70	0.39	10800	**-2,007.98**	0.00	3638	0.06	0.00
036_B_half	**-128.26**	0.00	6769	**-128.26**	0.00	759	0.00	0.00
036_B_one	**-624.41**	0.00	542	**-624.41**	0.00	1064	0.00	0.00
036_B_p_two	**133.13**	0.00	6884	134.57	8.91	10800	0.00	1.08
036_B_two	**-1,616.64**	0.00	1332	**-1,616.64**	0.00	921	0.00	0.00
041_B_half	-167.89	12.01	10800	**-185.75**	0.00	923	9.62	0.00
041_B_one	**-760.62**	0.00	7945	**-760.62**	0.00	1820	0.00	0.00
041_B_p_two	109.54	8.72	10800	**109.48**	0.00	10468	0.05	0.00
041_B_two	**-1,923.96**	0.00	7691	**-1,923.96**	0.00	9753	0.00	0.00
045_B_half	-261.05	55.00	10808	-489.41	13.73	10800	46.66	0.00
045_B_one	—	0.00	0	-1,646.77	3.03	10800	—	0.00
045_B_p_two	346.24	216.42	10800	201.06	77.15	10800	72.21	0.00
045_B_two	-3,932.70	2.43	10800	-3,961.58	1.64	10800	0.73	0.00
048_B_half	-20,722.60	46.50	10800	**-37,205.50**	0.00	5702	44.30	0.00
048_B_one	—	0.00	0	-106,881.00	0.20	10800	—	0.00
048_B_p_two	—	0.00	0	**-4,244.69**	0.00	9757	—	0.00
048_B_two	—	0.00	0	-247,766.00	0.65	10800	—	0.00
051_B_half	—	0.00	0	-310.02	3.80	10800	—	0.00
051_B_one	-1,084.35	1.79	10800	-1,078.55	2.22	10800	0.00	0.53
051_B_p_two	166.58	61.02	10800	140.41	33.48	10800	18.64	0.00
051_B_two	-2,631.90	1.10	10800	-2,635.00	0.81	10800	0.12	0.00
072_B_half	—	0.00	0	-316.67	30.33	10800	—	0.00
072_B_one	—	0.00	0	-926.29	12.98	10800	—	0.00
072_B_p_two	—	0.00	0	-16.48	79.04	10800	—	0.00
072_B_two	—	0.00	0	-1,769.06	23.65	10800	—	0.00
076_B_half	—	0.00	0	-562.87	10.30	10800	—	0.00
076_B_one	—	0.00	0	-1,742.54	3.59	10800	—	0.00
076_B_p_two	—	0.00	0	56.34	269.69	10800	—	0.00
076_B_two	—	0.00	0	-4,101.85	1.55	10800	—	0.00
101_B_half	—	0.00	0	-909.32	6.59	10800	—	0.00
101_B_one	—	0.00	0	-2,538.99	2.45	10800	—	0.00
101_B_p_two	—	0.00	0	-49.55	55.95	10800	—	0.00
101_B_two	—	0.00	0	-5,798.30	1.08	10800	—	0.00
111_B_half	—	0.00	0	-1,322.56	14.19	10800	—	0.00
111_B_one	—	0.00	0	-3,256.72	6.34	10800	—	0.00
111_B_p_two	—	0.00	0	-249.41	39.34	10800	—	0.00
111_B_two	—	0.00	0	-7,124.81	3.00	10800	—	0.00

Table 5.6 summarizes the number of best solutions, optimal solutions and proved optimals found by each formulation. The optimals column refers to all optimal solutions found, even when the model could not prove the optimality. In such case we know that the value is optimal by the results of the other formulations.

According to Table 5.6, one can easily notice that the number of proved optimals of our four-commodity network flow formulation is much higher than the others. While SB has proved only 1 optimal solution and GLS, 8, our $4C$ formulation has proved a total of 38 optimals. The formulations SB and GLS have found the optimal value for 19 and 24 instances, respectively, although most of them were not proved by these models. For example, we can see in Table 5.7 that the SB and GLS formulations have found the optimal solution for the instance `016_B_one`, but could not prove its optimality. Comparing the average deviation from the solutions of the formulations to the best value found, all models have about the same average value of 1.5%, not including in the calculation the deviations of instance `072_B_p_two` which have deviation values very high for the formulations SB and GLS, which is not the behavior for the other instances. Regarding the computational time, for instances with number of customers up to 26 our approach clearly outmatches by far the other formulations. For example, for these benchmarks the average time of our $4C$ formulation is 124 seconds, against 8546 and 7869 seconds of SB and GLS, respectively. In extremes cases such as for instance `016_B_one` our formulation required only 9 seconds to find the optimal and prove its optimality, while the SB and GLS have spent the 3 hours without proving the optimality of the solution found.

Table 5.6: Comparison of the number of best solutions, optimals and proved optimals found by the exact approaches for the SVRPDSP

Formulation	best solutions	optimals	proved optimals
SB	38	19	1
GLS	42	24	8
4C	49	38	38

An interesting fact is that the GLS formulation is more efficient than the other ones for instances of type `p_two`. We have not performed further analysis to check the cause of these behavior, however we foresee that the fact that these instances have the least revenue values accounts for these results.

In addition we have compared the lower bound values of the three formulations, obtained by solving the linear relaxation of the formulation. In order to properly analyse these values, we have normalized them in the interval of $0 : 1$. Figure 5.6 shows the normalized lower bound values for all 68 instances. In general our proposed $4C$ formulation has the best lower bound values. Only for some instances of type `p_two` the

Süral-Bookbinder model has the best values. One interesting fact is that the relation between the lower bound of the formulations from the literature is highly impacted by the relation between revenue and transportation costs: as the value of ω increases from 0.2 to 2.0, i.e., the revenues generated by the pickups collected increases, the linear bound of the Süral-Bookbinder formulation becomes worse and the linear bound of the Gribkovskaia-Laporte-Shyshou becomes better. Furthermore, we performed an ANOVA on the normalized values, presented in Figure 5.7. According to the result of this analysis the lower bound of the $4C$ formulation is statistically the best by far, with 95% of confidence. This proves that our formulation is the strongest among them. Having a strong linear lower bound allows the proposed $4C$ formulation to prove optimality for several instances, while the others fail.

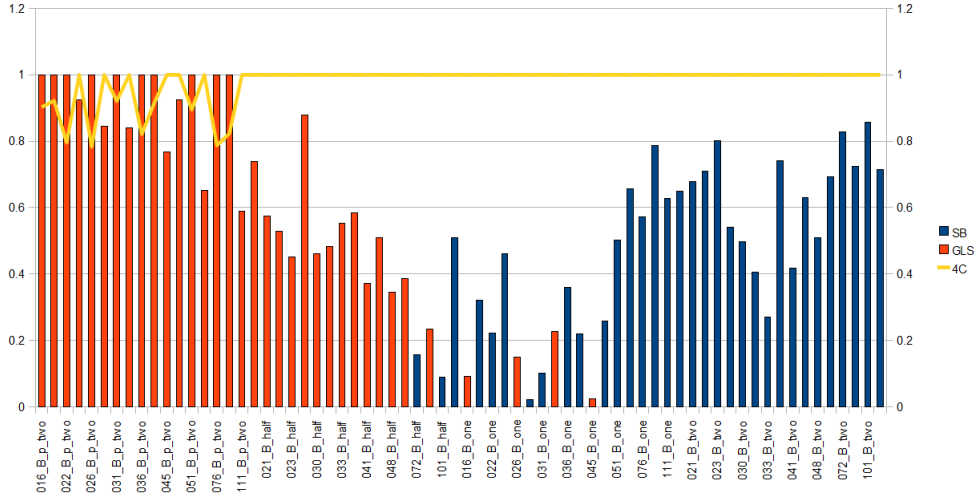


Figure 5.6: Graph of the normalized lower bound values of the three formulations for the SVRPDSP.

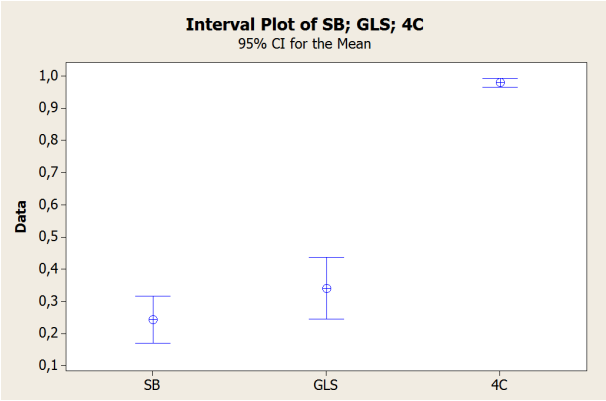


Figure 5.7: ANOVA of the normalized lower bound values of the three formulations for the SVRPDSP.

Table 5.7: Comparison of the solution values and computational times from the formulations for the SVRPDSP

Instance	SB (initial solution)		GLS (Branch&Cut)		4C (initial solution)		Deviation (%)		
	solution	total time	solution	total time	solution	total time	SB	GLS	4C
016_B_half	**36.60**	10800	**36,60***	196	**36.60***	3	0	0,00	0,00
016_B_one	**-150.73**	10800	**-150,73**	10800	**-150.73***	9	0,00	0,00	0,00
016_B_p_two	**132.28***	1248	**132,28***	1	**132.28***	9	0,00	0,00	0,00
016_B_two	**-536.13**	10800	**-536,13**	10800	**-536.13***	8	0,00	0,00	0,00
021_B_half	**-20.16**	7441	**-20,16**	10800	**-20.16***	11	0,00	0,00	0,00
021_B_one	**-307.80**	7777	**-307,80**	10800	**-307.80***	22	0,00	0,00	0,00
021_B_p_two	**137.59**	10800	**137,59***	9	**137.59***	70	0,00	0,00	0,00
021_B_two	-893.48	5180	**-901,28**	10800	**-901.28***	16	0,87	0,00	0,00
022_B_half	**-62.63**	10800	-56,86	10800	**-62.63***	192	0,00	9,21	0,00
022_B_one	**-421.04**	10800	-420,96	5689	**-421.04***	140	0,00	0,02	0,00
022_B_p_two	124.33	9862	**123,59***	57	**123.59***	173	0,60	0,00	0,00
022_B_two	**-1,149.38**	10800	-1.149,30	10800	**-1,149.38***	124	0,00	0,01	0,00
023_B_half	**-80.95**	3905	**-80,95**	10800	**-80.95***	229	0,00	0,00	0,00
023_B_one	**-698.35**	10800	**-698,35**	10800	**-698.35***	167	0,00	0,00	0,00
023_B_p_two	276.60	1325	**269,02**	10800	**269.02***	377	2,82	0,00	0,00
023_B_two	**-1,933.10**	10800	**-1.933,10**	10800	**-1,933.10***	159	0,00	0,00	0,00
026_B_half	**-92.41**	10800	**-92,41**	10800	**-92.41***	127	0,00	0,00	0,00
026_B_one	**-497.17**	10800	**-497,17**	10800	**-497.17***	341	0,00	0,00	0,00
026_B_p_two	146.90	5981	**145,15***	231	**145.15***	206	1,21	0,00	0,00
026_B_two	**-1,334.26**	9414	**-1.334,26**	10800	**-1,334.26***	112	0,00	0,00	0,00
030_B_half	**-378.64**	8574	**-378,64**	10800	**-378.64**	10800	0,00	0,00	0,00
030_B_one	**-1,152.07**	7689	**-1.152,07**	10800	**-1,152.07***	10579	0,00	0,00	0,00
030_B_p_two	87.13	9073	**82,29**	10800	85.49	10800	5,88	0,00	3,89
030_B_two	**-2,699.00**	6422	**-2.699,00**	10800	**-2,699.00***	9450	0,00	0,00	0,00
031_B_half	-84.16	9041	-87,49	10800	**-91.20***	2262	7,72	4,07	0,00
031_B_one	-506.02	9030	**-508,27**	10800	**-508.27***	367	0,44	0,00	0,00
031_B_p_two	123.26	5127	**121,17***	232	**121.17***	436	1,72	0,00	0,00
031_B_two	-1,338.64	5365	-1.352,79	10800	**-1,355.59***	893	1,25	0,21	0,00
033_B_half	-134.42	4716	-146,16	10800	**-150.73***	8588	10,82	3,03	0,00
033_B_one	-742.36	10800	-753,88	10800	**-765.79***	9011	3,06	1,56	0,00
033_B_p_two	205.48	10800	**194,61**	10800	194.76	10800	5,59	0,00	0,08
033_B_two	-1,995.71	10800	-2.001,44	10800	**-2,007.98***	3638	0,61	0,33	0,00
036_B_half	-120.93	3373	-127,06	10800	**-128.26***	759	5,71	0,94	0,00
036_B_one	-613.70	10800	-616,19	10800	**-624.41***	1064	1,72	1,32	0,00
036_B_p_two	139.59	8471	**130,42***	546	134.57	10800	7,03	0,00	3,18
036_B_two	-1,605.92	907	-1.612,42	10800	**-1,616.64***	921	0,66	0,26	0,00
041_B_half	**-185.75**	1760	-181,81	10800	**-185.75***	923	0,00	2,12	0,00
041_B_one	-760.52	10800	-759,38	10800	**-760.62***	1820	0,01	0,16	0,00
041_B_p_two	123.03	10800	**109,48***	1172	**109.48***	10468	12,38	0,00	0,00
041_B_two	-1,916.25	10800	-1.923,25	10800	**-1,923.96***	9753	0,40	0,04	0,00
045_B_half	-489.41	2490	**-491,15**	10800	-489.41	10800	0,35	0,00	0,35
045_B_one	**-1,648.51**	10800	**-1.648,51**	10800	-1,646.77	10800	0,00	0,00	0,11
045_B_p_two	**200.41**	10800	208,37	10800	201.06	10800	0,00	3,97	0,32
045_B_two	**-3,963.32**	10800	**-3.963,32**	10800	-3,961.58	10800	0,00	0,00	0,04
048_B_half	-36,453.90	10800	-36.657,20	10800	**-37,205.50***	5702	2,02	1,47	0,00
048_B_one	-106,526.00	10800	-106.526,00	10800	**-106,881.00**	10800	0,33	0,33	0,00
048_B_p_two	-3,826.48	10800	**-4.244,69**	10800	**-4,244.69***	9757	9,85	0,00	0,00
048_B_two	**-247,766.00**	10800	**-247.766,00**	10800	**-247,766.00**	10800	0,00	0,00	0,00
051_B_half	**-310.02**	10800	**-310,02**	10800	**-310.02**	10800	0,00	0,00	0,00
051_B_one	-1,080.81	10800	**-1.087,38**	10800	-1,078.55	10800	0,60	0,00	0,81
051_B_p_two	141.13	10800	**127,83**	10800	140.41	10800	10,40	0,00	9,84
051_B_two	**-2,640.41**	10800	-2.622,11	10800	-2,635.00	10800	0,00	0,69	0,20
072_B_half	**-361.00**	9773	-213,60	10800	-316.67	10800	0,00	40,83	12,28
072_B_one	**-989.06**	9844	-771,98	10800	-926.29	10800	0,00	21,95	6,35
072_B_p_two	-0.39	10800	66,11	10800	**-16.48**	10800	97,63	501,15	0,00
072_B_two	**-2,203.27**	10800	-2.083,35	10800	-1,769.06	10800	0,00	5,44	19,71
076_B_half	**-573.05**	10800	-565,17	10800	-562.87	10800	0,00	1,38	1,78
076_B_one	-1,737.32	3170	**-1.742,54**	10800	**-1,742.54**	10800	0,30	0,00	0,00
076_B_p_two	41.83	10800	**39,93**	10800	56.34	10800	4,76	0,00	41,10
076_B_two	**-4,101.85**	10800	**-4.101,85**	10800	**-4,101.85**	10800	0,00	0,00	0,00
101_B_half	-909.32	10800	**-912,77**	10800	-909.32	10800	0,38	0,00	0,38
101_B_one	**-2,538.99**	10800	**-2.538,99**	10800	**-2,538.99**	10800	0,00	0,00	0,00
101_B_p_two	**-49.55**	10800	**-49,55**	10800	**-49.55**	10800	0,00	0,00	0,00
101_B_two	**-5,798.30**	10800	**-5.798,30**	10800	**-5,798.30**	10800	0,00	0,00	0,00
111_B_half	**-1,369.99**	10800	-1.354,63	10800	-1,322.56	10800	0,00	1,12	3,46
111_B_one	**-3,277.19**	10800	-3.256,72	10800	-3,256.72	10800	0,00	0,62	0,62
111_B_p_two	**-249.79**	10800	**-249,79**	10800	-249.41	10800	0,00	0,00	0,15
111_B_two	**-7,173.87**	10800	-7.124,81	10800	-7,124.81	10800	0,00	0,68	0,68

Finally, since we have proved several optimal solutions with our proposed $4C$ formulation we decided to check the optimality of the solutions found by the metaheuristics. Thus Table 5.8 presents the results of the best heuristic approaches for instances where the optimal has been proved. According to these results the GVNS has found more optimal solutions, totalizing 23, against 21 of the EA and 12 of the VND. However notice that our EA has found the optimal in all 10 runs for 12 instances, against only 6 of the GVNS. In addition, an interesting fact is that the VND has found its 12 optimal solutions in all 10 runs, which indicates that this algorithm is robust.

Table 5.8: Comparison between the solutions found by the metaheuristics and the optimals proved by the formulations.

Instance	GVNS [6]		EA		VND		optimal value
	best	average	best	average	best	average	
016_B_half	39.86	40.61	**36.60**	**36.60**	**36.60**	**36.60**	36.60
016_B_one	**-150.73**	**-150.73**	**-150.73**	-145.44	-145.44	-145.44	-150.73
016_B_p_two	**132.28**	133.43	133.80	142.03	142.03	142.03	132.28
016_B_two	**-536.13**	**-536.13**	-536.06	-530.84	-530.84	-530.84	-536.13
021_B_half	-18.78	-13.96	**-20.16**	**-20.16**	**-20.16**	**-20.16**	-20.16
021_B_one	**-307.80**	-307.21	**-307.80**	-300.00	-300.00	-300.00	-307.80
021_B_p_two	**137.59**	141.17	**137.59**	147.03	147.03	147.03	137.59
021_B_two	**-901.28**	-900.69	**-901.28**	-893.48	-893.48	-893.48	-901.28
022_B_half	**-62.63**	-58.59	-62.40	-56.86	-56.86	-56.86	-62.63
022_B_one	**-421.04**	**-421.04**	**-421.04**	**-421.04**	**-421.04**	**-421.04**	-421.04
022_B_p_two	**123.59**	124.86	124.25	124.33	124.33	124.33	123.59
022_B_two	**-1,149.38**	**-1,149.38**	**-1,149.38**	**-1,149.38**	**-1,149.38**	**-1,149.38**	-1149.38
023_B_half	**-80.95**	-80.76	-80.01	-61.67	-61.67	-61.67	-80.95
023_B_one	**-698.35**	-698.26	-697.41	-640.23	-640.23	-640.23	-698.35
023_B_p_two	269.86	273.23	269.08	291.07	291.07	291.07	269.02
023_B_two	**-1,933.10**	-1932.91	-1932.16	-1874.98	-1874.98	-1874.98	-1933.10
026_B_half	-87.87	-85.53	**-92.41**	**-92.41**	**-92.41**	**-92.41**	-92.41
026_B_one	**-497.17**	**-497.17**	**-497.17**	-495.87	-495.87	-495.87	-497.17
026_B_p_two	146.90	147.18	146.11	146.34	146.34	146.34	145.15
026_B_two	**-1,334.26**	**-1,334.26**	**-1,334.26**	-1332.96	-1332.96	-1332.96	-1334.26
030_B_one	**-1,152.07**	-1146.64	**-1,152.07**	**-1,152.07**	**-1,152.07**	**-1,152.07**	-1152.07
030_B_two	**-2,699.00**	-2697.26	**-2,699.00**	**-2,699.00**	**-2,699.00**	**-2,699.00**	-2699.00
031_B_half	-87.06	-84.90	-89.15	-85.66	-85.66	-85.51	-91.20
031_B_one	**-511.07**	-508.04	**-511.07**	-507.82	-507.82	-507.69	-511.07
031_B_p_two	123.15	123.39	123.15	131.07	131.07	131.07	121.17
031_B_two	-1354.49	-1352.73	**-1,355.59**	-1352.34	-1352.34	-1352.21	-1355.59
033_B_half	**-150.73**	-146.01	**-150.73**	**-150.73**	**-150.73**	**-150.73**	-150.73
033_B_one	**-765.79**	-761.80	**-765.79**	-753.55	-753.55	-753.55	-765.79
033_B_two	**-2,007.98**	-2003.99	-2007.28	-1995.74	-1995.74	-1995.74	-2007.98
036_B_half	-127.06	-124.65	**-128.26**	**-128.26**	**-128.26**	**-128.26**	-128.26
036_B_one	-616.12	-610.63	**-624.41**	**-624.41**	**-624.41**	**-624.41**	-624.41
036_B_two	-1608.35	-1604.65	**-1,616.64**	**-1,616.64**	**-1,616.64**	**-1,616.64**	-1616.64
041_B_half	-183.86	-179.36	**-185.75**	-181.85	-181.85	-181.85	-185.75
041_B_one	-760.52	-758.60	-758.06	-757.26	-757.26	-749.98	-760.62
041_B_p_two	**109.48**	115.16	111.69	131.86	131.86	131.86	109.48
041_B_two	-1923.86	-1921.50	-1922.71	-1920.60	-1920.60	-1913.59	-1923.96
048_B_half	-37200.62	-37200.62	-36786.80	**-37,205.50**	**-37,205.50**	**-37,205.50**	-37205.50
048_B_p_two	**-4,244.69**	-4122.47	-3830.83	-4140.10	-4140.10	-4140.10	-4244.69
Total	23	6	21	10	12	12	38

5.4 Experimental results for the MVRPDSP

We present in this section the results obtained by our proposed approaches for the MVRPDSP. Notice that since there is no literature regarding this problem, we are only going to compare the results among our approaches.

Initial tests with the first formulation (the Süral-Bookbinder based formulation) showed that it consumes a lot of memory and, consequently, its execution stops prematurely in several cases, delivering only a feasible solution or none for the most difficult scenarios. For many instances this time was less than one hour. Since the second formulation (the four-commodity based formulation) is able to run for much longer without throwing a memory overflow error we decided to assign a time limit of an hour for both models, in order to fairly compare them.

At first we tested the performance of the first formulation proposed in section 4.3.1 with its improvements, but no initial solution provided (here called *normal* - N) and then the same formulation providing an initial feasible solution (IS) using the hybrid construcive heuristic presented in section 4.2. This initial solution was generated by running once the constructive with $rclSize = 1$, then 30 times with $rclSize = 2$ and selecting the best solution value. The results are presented in Table 5.9. Values highlighted in bold face indicate optimal solutions. The last two columns refers to the deviation of the solution found by the constructive and by the IS from the best solution found. We did not consider the deviation of the configuration N since it was not able to find a single feasible solution for a total of 41 instances.

For instances of type `C_0.2`, except one, the model without initial solution was not able to find a single feasible solution when no initial solution was provided. This behavior could be due to the number of vehicles in such instances, which is higher than in the types `C_0.4` and `C_0.6`. In addition notice that the model found the optimal solution for only the 4 easiest instances, those with least number of customers and vehicles with highest capacity.

When we provided the model with an initial solution the results were significantly improved, although no other optimal solutions were found. In some rare cases the solution found by IS is worse than the one found by N, for example for instances `026_B_half_C_0.2` and `026_B_one_C_0.2`. According to the deviation columns in Table 5.9 for 24 instances the constructive has found the best solution, which could indicate that the constructive generates solution with high quality. In addition, according to the last column, one can notice that for 66 out of the 72 instances the configuration IS has found the best results. An interesting fact is that the constructive has found the same 4 optimal solutions found by the model, which is another indication that it

can generate very good quality solutions. This shows that not only this formulation needs a feasible solution to start with, but the constructive heuristic proposed is a good one to provide the initial solutions.

The second trial was performed with the proposed four-commodity network flow formulation for the MVRPDSP presented in section 4.3.2. We used the same procedure done to test the first formulation, where, at first the model was tested without an initial solution (N), then with an initial solution (IS) provided by the constructive. The results of these tests are presented in Table 5.10. Values highlighted in bold face indicate optimal solutions. The last two columns refers to the deviation of the solution foud by the constructive and by the IS from the best solution found. Notice that as for the previous formulation we do not consider the deviation of the configuration N. However for this formulation, the number of instances where the model was not able to find a single feasible solution was higher, totalizing 56, which accounts for more than 2/3 of the total of instances. This result was not expected since the four-commodity network flow formulation for the SVRPDSP presented the best results. However when providing an initial solution the quality of the solutions found by the models has been improved. Also notice in the last column that for 32 instances the model was able to improve the solution generated by the constructive heuristic.

This formulation reached the 1 hour time limit for all instances (except the 4 optimals which was found in matter of seconds). The first formulation does not reach this limit for most of the instances nor find the optimal, which indicate that the solver was interrupted by memory error. Then, we can foresee that the $4C$-based formulation can still improve the solutions if more time is given, while the other cannot proceed, unless a machine with more memory is used.

Finally, Table 5.11 compare the results of both formulations and the results of the constructive heuristic. Since in general the quality of the solutions were improved by providing the models with an initial solution, we only include these results. Notice that the last three columns refer to the percentage deviation to the best solution found.

According to the results shown in Table 5.11 the average deviation of the formulation SB in relation to the best solutions found is 1.69%, against 2.59% of $4C$ and 5.62% of the constructive. Notice that for these calculations we have removed instances with deviation values highly different from the behavior of the other instances. For instance, we have not included the deviation values of instances `036_B_half_C_0.2`, `045_B_half_C_0.2`, and `072_B_p_two_C_0.6`. Notice that this data indicates that none of the formulations outmatches the other. Also notice that the solutions of the constructive did not have a very high average deviation and for 19 instances it has found the best value, including 4 proved optimals, pointing out that the constructive

can generate good quality solutions.

In addition, as for the SVRPDSP, we have performed an analysis in the lower bound values of the formulations, however we are not going to present any graphs nor tables in these case, since for all 72 instances the $4C$ model had the best lower bound value, thus there is no variation. In addition, we have calculated that the average deviation of the lower bound of SB from $4C$ is of about 42%. We have excluded from this average the deviation values of instances `045_B_p_two_C_0.2` and `016_B_half_C_0.4`, which had very high deviation values of 2047.70% and 860.43%, respectively.

Table 5.9: Results of the proposed formulation for the MVRPDSP based on the one from Süral and Bookbinder [30] with and without initial solution

Instance	normal (N)			constructive	initial solution (IS)			dev from best(%)	
	solution	gap	total time	solution	solution	gap	total time	C	IS
016_B_p_two_C_0.6	**149.94**	**0**	2616	**149.94**	**149.94**	**0**	3471	0.00	0.00
016_B_half_C_0.6	**32.19**	**0**	857	**32.19**	**32.19**	**0**	601	0.00	0.00
016_B_one_C_0.6	**-174.27**	**0**	944	**-174.27**	**-174.27**	**0**	615	0.00	0.00
016_B_two_C_0.6	**-587.20**	**0**	768	**-587.20**	**-587.20**	**0**	712	0.00	0.00
026_B_p_two_C_0.6	337.42	193.48	401	188.06	184.57	55.75	2061	1.89	0.00
026_B_half_C_0.6	-98.77	29.13	3600	-73.54	-77.09	42.38	2585	25.54	21.95
026_B_one_C_0.6	-534.74	6.35	3600	-509.51	-516.84	10.01	2947	4.72	3.35
026_B_two_C_0.6	-1,406.23	2.14	3600	-1,381.48	-1,406.71	2.50	2021	1.79	0.00
036_B_p_two_C_0.6	294.39	152.61	936	152.78	152.78	32.60	1160	0.00	0.00
036_B_half_C_0.6	-70.64	60.59	1029	-126.79	-142.78	19.98	1488	11.20	0.00
036_B_one_C_0.6	161.24	122.79	708	-637.97	-644.34	8.73	884	0.99	0.00
036_B_two_C_0.6	-1,744.23	0.67	3600	-1,660.26	-1,741.76	0.76	2070	4.81	0.14
045_B_p_two_C_0.6	907.94	8,839.80	2460	228.89	194.36	940.90	809	17.77	0.00
045_B_half_C_0.6	-172.85	76.07	3601	-461.41	-493.10	30.85	969	6.43	0.00
045_B_one_C_0.6	273.68	114.52	891	-1,618.77	-1,650.97	13.26	772	1.95	0.00
045_B_two_C_0.6	-1,137.49	73.48	855	-3,933.57	-3,965.77	7.23	3686	0.81	0.00
051_B_p_two_C_0.6	493.30	458.12	3601	126.68	126.68	40.25	3332	0.00	0.00
051_B_half_C_0.6	-34.46	91.21	3601	-358.30	-365.16	7.05	2313	1.88	0.00
051_B_one_C_0.6	-414.62	65.56	3601	-1,168.98	-1,171.95	2.63	2250	0.25	0.00
051_B_two_C_0.6	-2,668.11	5.55	3601	-2,790.27	-2,797.13	0.99	2043	0.25	0.00
072_B_p_two_C_0.6	518.00	577.05	3601	-2.17	-8.00	92.50	4231	72.88	0.00
072_B_half_C_0.6	—	—	1111	-329.44	-379.09	21.50	4263	13.10	0.00
072_B_one_C_0.6	—	—	3600	-949.13	-949.13	14.50	3574	0.00	0.00
072_B_two_C_0.6	-525.22	77.77	3608	-2,180.54	-2,231.77	5.55	4337	2.30	0.00
016_B_p_two_C_0.4	197.81	54.18	850	190.18	187.61	41.01	1026	1.37	0.00
016_B_half_C_0.4	146.13	1,053.15	663	80.69	78.12	388.68	1156	3.29	0.00
016_B_one_C_0.4	-45.87	76.00	790	-112.00	-128.10	32.13	475	12.57	0.00
016_B_two_C_0.4	-481.85	20.56	924	-497.40	-542.79	10.45	382	8.36	0.00
026_B_p_two_C_0.4	664.31	549.40	503	202.04	195.04	68.16	1599	3.59	0.00
026_B_half_C_0.4	567.57	485.98	471	-49.09	-56.09	61.91	556	12.48	0.00
026_B_one_C_0.4	8.44	101.45	1194	-467.62	-472.85	18.74	968	1.11	0.00
026_B_two_C_0.4	-714.08	50.90	617	-1,304.71	-1,344.82	7.31	1294	2.98	0.00
036_B_p_two_C_0.4	—	—	1734	177.92	177.92	55.23	1453	0.00	0.00
036_B_half_C_0.4	—	—	3600	-90.97	-95.45	46.94	1910	4.69	0.00
036_B_one_C_0.4	—	—	1649	-606.80	-641.00	9.57	598	5.34	0.00
036_B_two_C_0.4	—	—	1655	-1,629.09	-1,698.42	3.45	2053	4.08	0.00
045_B_p_two_C_0.4	—	—	3600	311.15	250.06	3,488.42	3645	24.43	0.00
045_B_half_C_0.4	—	—	3600	-355.05	-433.35	37.79	777	18.07	0.00
045_B_one_C_0.4	—	—	3602	-1,507.57	-1,616.63	15.79	686	6.75	0.00
045_B_two_C_0.4	—	—	3602	-3,822.37	-3,958.64	7.34	1247	3.44	0.00
051_B_p_two_C_0.4	—	—	3606	162.16	162.16	88.52	905	0.00	0.00
051_B_half_C_0.4	—	—	832	-296.04	-296.04	25.30	458	0.00	0.00
051_B_one_C_0.4	—	—	1011	-1,090.51	-1,125.69	6.67	3665	3.13	0.00
051_B_two_C_0.4	—	—	870	-2,679.38	-2,679.38	5.33	664	0.00	0.00
072_B_p_two_C_0.4	—	—	3599	-16.63	-18.14	83.42	3757	8.32	0.00
072_B_half_C_0.4	—	—	1726	-381.47	-381.47	21.57	2492	0.00	0.00
072_B_one_C_0.4	—	—	3579	-998.94	-998.94	10.23	1275	0.00	0.00
072_B_two_C_0.4	—	—	3600	-2,233.91	-2,233.91	5.53	2395	0.00	0.00
016_B_p_two_C_0.2	—	—	507	286.74	282.41	93.95	606	1.53	0.00
016_B_half_C_0.2	—	—	534	177.25	172.92	410.98	694	2.50	0.00
016_B_one_C_0.2	—	—	598	-15.44	-15.44	91.11	568	0.00	0.00
016_B_two_C_0.2	-270.80	53.89	565	-400.85	-405.18	30.89	564	1.07	0.00
026_B_p_two_C_0.2	—	—	1353	347.80	347.80	186.99	514	0.00	0.00
026_B_half_C_0.2	—	—	1327	128.56	104.44	175.00	709	23.09	0.00
026_B_one_C_0.2	—	—	1341	-219.84	-269.91	53.15	629	18.55	0.00
026_B_two_C_0.2	—	—	1334	-987.17	-1,098.13	24.22	765	10.10	0.00
036_B_p_two_C_0.2	—	—	3605	284.77	277.89	142.02	948	2.48	0.00
036_B_half_C_0.2	—	—	3604	3.41	-17.55	90.46	900	119.43	0.00
036_B_one_C_0.2	—	—	3605	-507.78	-517.06	27.19	697	1.79	0.00
036_B_two_C_0.2	—	—	3604	-1,530.07	-1,569.42	10.96	954	2.51	0.00
045_B_p_two_C_0.2	—	—	3603	842.15	783.33	1,786.71	435	7.51	0.00
045_B_half_C_0.2	—	—	566	-0.30	161.38	124.71	551	0.00	53893.33
045_B_one_C_0.2	—	—	3605	-882.19	-915.25	50.05	746	3.61	0.00
045_B_two_C_0.2	—	—	3599	-3,171.12	-3,068.56	26.91	602	0.00	3.23
051_B_p_two_C_0.2	—	—	1108	306.03	306.03	258.04	627	0.00	0.00
051_B_half_C_0.2	—	—	1061	-139.63	-139.63	64.71	747	0.00	0.00
051_B_one_C_0.2	—	—	1587	-901.67	-901.67	25.26	1697	0.00	0.00
051_B_two_C_0.2	—	—	3609	-2,425.68	-2,425.68	14.22	1415	0.00	0.00
072_B_p_two_C_0.2	—	—	2372	251.86	241.61	322.11	3422	4.24	0.00
072_B_half_C_0.2	—	—	3156	-53.79	-64.98	86.57	2044	17.22	0.00
072_B_one_C_0.2	—	—	2800	-661.70	-588.88	46.93	2607	0.00	11.00
072_B_two_C_0.2	—	—	2130	-1,815.34	-1,815.34	23.18	2476	0.00	0.00

Table 5.10: Results of the proposed four-commodity network flow formulation for the MVRPDSP with and without initial solution

Instance	normal (N)			constructive	initial solution (IS)			dev from best (%)	
	solution	gap	total time	solution	solution	gap	total time	C	IS
016_B_p_two_C_0.6	**149.94**	**0**	311	**149.94**	**149.94**	**0**	155	0.00	0.00
016_B_half_C_0.6	**32.19**	**0**	478	**32.19**	**32.19**	**0**	13	0.00	0.00
016_B_one_C_0.6	**-174.27**	**0**	63	**-174.27**	**-174.27**	**0**	15	0.00	0.00
016_B_two_C_0.6	**-587.20**	**0**	25	**-587.20**	**-587.20**	**0**	12	0.00	0.00
026_B_p_two_C_0.6	260.71	93.70	3600	188.06	188.00	49.57	3600	0.03	0.00
026_B_half_C_0.6	261.41	301.65	3600	-73.54	-83.48	31.22	3600	11.91	0.00
026_B_one_C_0.6	-460.96	17.70	3600	-509.51	-515.47	8.08	3600	1.16	0.00
026_B_two_C_0.6	-995.59	30.43	3600	-1,381.48	-1,406.53	0.95	3600	1.78	0.00
036_B_p_two_C_0.6	150.19	28.21	3600	152.78	152.78	28.38	3600	1.72	1.72
036_B_half_C_0.6	—	—	3599	-126.79	-126.79	28.19	3600	0.00	0.00
036_B_one_C_0.6	—	—	3596	-637.97	-637.97	8.82	3600	0.00	0.00
036_B_two_C_0.6	-1,744.23	0.22	3600	-1,660.26	-1,660.26	5.63	3600	4.81	4.81
045_B_p_two_C_0.6	—	—	3592	228.89	190.41	535.76	3600	20.21	0.00
045_B_half_C_0.6	—	—	3592	-461.41	-495.29	27.64	3600	6.84	0.00
045_B_one_C_0.6	—	—	3592	-1,618.77	-1,652.65	11.48	3600	2.05	0.00
045_B_two_C_0.6	—	—	3592	-3,933.57	-3,967.45	6.66	3600	0.85	0.00
051_B_p_two_C_0.6	—	—	3588	126.68	122.68	23.64	3600	3.26	0.00
051_B_half_C_0.6	—	—	3589	-358.30	-358.30	7.71	3600	0.00	0.00
051_B_one_C_0.6	—	—	3589	-1,168.98	-1,168.98	2.84	3600	0.00	0.00
051_B_two_C_0.6	—	—	3589	-2,790.27	-2,794.27	1.08	3600	0.14	0.00
072_B_p_two_C_0.6	—	—	3566	-2.17	-19.40	79.89	3600	88.81	0.00
072_B_half_C_0.6	—	—	3566	-329.44	-332.26	29.61	3600	0.85	0.00
072_B_one_C_0.6	—	—	3566	-949.13	-949.13	13.57	3600	0.00	0.00
072_B_two_C_0.6	—	—	3567	-2,180.54	-2,181.95	7.18	3600	0.06	0.00
016_B_p_two_C_0.4	174.74	10.53	3600	190.18	175.27	8.63	3600	8.84	0.30
016_B_half_C_0.4	68.94	48.34	3600	80.69	65.59	30.88	3600	23.02	0.00
016_B_one_C_0.4	-142.64	9.17	3600	-112.00	-128.91	17.31	3600	21.48	9.63
016_B_two_C_0.4	-555.57	2.28	3600	-497.40	-551.47	2.53	3600	10.47	0.74
026_B_p_two_C_0.4	—	—	3598	202.04	191.24	39.57	3600	5.65	0.00
026_B_half_C_0.4	—	—	3597	-49.09	-70.36	27.84	3600	30.23	0.00
026_B_one_C_0.4	—	—	3602	-467.62	-506.33	5.06	3600	7.65	0.00
026_B_two_C_0.4	-920.28	34.73	3600	-1,304.71	-1,378.30	2.27	3600	5.34	0.00
036_B_p_two_C_0.4	—	—	3593	177.92	177.92	37.44	3600	0.00	0.00
036_B_half_C_0.4	—	—	3593	-90.97	-90.97	43.75	3600	0.00	0.00
036_B_one_C_0.4	—	—	3593	-606.80	-655.95	4.76	3600	7.49	0.00
036_B_two_C_0.4	—	—	3593	-1,629.09	-1,708.31	1.57	3600	4.64	0.00
045_B_p_two_C_0.4	—	—	3586	311.15	311.15	406.12	3600	0.00	0.00
045_B_half_C_0.4	—	—	3586	-355.05	-355.05	44.71	3600	0.00	0.00
045_B_one_C_0.4	—	—	3586	-1,507.57	-1,507.57	17.69	3600	0.00	0.00
045_B_two_C_0.4	—	—	3586	-3,822.37	-3,878.12	7.44	3600	1.44	0.00
051_B_p_two_C_0.4	—	—	3580	162.16	160.65	67.20	3600	0.94	0.00
051_B_half_C_0.4	—	—	3580	-296.04	-296.89	23.64	3600	0.29	0.00
051_B_one_C_0.4	—	—	3580	-1,090.51	-1,090.51	9.07	3600	0.00	0.00
051_B_two_C_0.4	—	—	3580	-2,679.38	-2,679.38	5.02	3600	0.00	0.00
072_B_p_two_C_0.4	—	—	3541	-16.63	-16.63	79.00	3600	0.00	0.00
072_B_half_C_0.4	—	—	3540	-381.47	-381.47	16.26	3600	0.00	0.00
072_B_one_C_0.4	—	—	3541	-998.94	-998.94	7.59	3600	0.00	0.00
072_B_two_C_0.4	—	—	3541	-2,233.91	-2,233.91	4.28	3600	0.00	0.00
016_B_p_two_C_0.2	—	—	3599	286.74	269.77	23.38	3600	6.29	0.00
016_B_half_C_0.2	—	—	3599	177.25	172.92	51.27	3600	2.50	0.00
016_B_one_C_0.2	—	—	3599	-15.44	-19.78	78.53	3600	21.94	0.00
016_B_two_C_0.2	-174.98	65.50	3600	-400.85	-400.85	20.46	3600	0.00	0.00
026_B_p_two_C_0.2	—	—	3597	347.80	347.80	42.61	3600	0.00	0.00
026_B_half_C_0.2	—	—	3597	128.56	122.41	3,606.35	3600	5.02	0.00
026_B_one_C_0.2	—	—	3598	-219.84	-219.84	49.51	3600	0.00	0.00
026_B_two_C_0.2	—	—	3595	-987.17	-1,048.07	19.98	3600	5.81	0.00
036_B_p_two_C_0.2	—	—	3590	284.77	284.77	48.01	3600	0.00	0.00
036_B_half_C_0.2	—	—	3590	3.41	3.41	103.68	3600	0.00	0.00
036_B_one_C_0.2	—	—	3590	-507.78	-507.78	17.90	3600	0.00	0.00
036_B_two_C_0.2	—	—	3590	-1,530.07	-1,530.07	8.59	3600	0.00	0.00
045_B_p_two_C_0.2	—	—	3583	842.15	842.15	302.14	3600	0.00	0.00
045_B_half_C_0.2	—	—	3583	-0.30	-0.30	99.94	3600	0.00	0.00
045_B_one_C_0.2	—	—	3583	-882.19	-884.78	46.28	3600	0.29	0.00
045_B_two_C_0.2	—	—	3583	-3,171.12	-3,171.12	20.28	3600	0.00	0.00
051_B_p_two_C_0.2	—	—	3579	306.03	306.03	88.69	3600	0.00	0.00
051_B_half_C_0.2	—	—	3579	-139.63	-139.63	56.51	3600	0.00	0.00
051_B_one_C_0.2	—	—	3579	-901.67	-901.67	20.32	3600	0.00	0.00
051_B_two_C_0.2	—	—	3579	-2,425.68	-2,427.11	11.86	3600	0.06	0.00
072_B_p_two_C_0.2	—	—	3549	251.86	251.86	2,417.44	3600	0.00	0.00
072_B_half_C_0.2	—	—	3548	-53.79	-53.79	85.74	3600	0.00	0.00
072_B_one_C_0.2	—	—	3549	-661.70	-661.70	33.74	3600	0.00	0.00
072_B_two_C_0.2	—	—	3549	-1,815.34	-1,815.34	18.85	3600	0.00	0.00

Table 5.11: Comparison of the proposed approaches for the MVRPDSP

Instance	Süral-Bookbinder based (SB) solution value	four-commodity network flow (4C) solution value	constructive Heuristic (C) solution value	Deviation from the best solution(%) SB	4C	C
016_B_p_two_C_0.6	**149.94**	**149.94**	**149.94**	0.00	0.00	0.00
016_B_half_C_0.6	**32.19**	**32.19**	**32.19**	0.00	0.00	0.00
016_B_one_C_0.6	**-174.27**	**-174.27**	**-174.27**	0.00	0.00	0.00
016_B_two_C_0.6	**-587.20**	**-587.20**	**-587.20**	0.00	0.00	0.00
026_B_p_two_C_0.6	184.57	188.00	188.06	0.00	1.86	1.89
026_B_half_C_0.6	-77.09	-83.48	-73.54	7.65	0.00	11.91
026_B_one_C_0.6	-516.84	-515.47	-509.51	0.00	0.27	1.42
026_B_two_C_0.6	-1,406.71	-1,406.53	-1,381.48	0.00	0.01	1.79
036_B_p_two_C_0.6	152.78	152.78	152.78	0.00	0.00	0.00
036_B_half_C_0.6	-142.78	-126.79	-126.79	0.00	11.20	11.20
036_B_one_C_0.6	-644.34	-637.97	-637.97	0.00	0.99	0.99
036_B_two_C_0.6	-1,741.76	-1,660.26	-1,660.26	0.00	4.68	4.68
045_B_p_two_C_0.6	194.36	190.41	228.89	2.07	0.00	20.21
045_B_half_C_0.6	-493.10	-495.29	-461.41	0.44	0.00	6.84
045_B_one_C_0.6	-1,650.97	-1,652.65	-1,618.77	0.10	0.00	2.05
045_B_two_C_0.6	-3,965.77	-3,967.45	-3,933.57	0.04	0.00	0.85
051_B_p_two_C_0.6	126.68	122.68	126.68	3.26	0.00	3.26
051_B_half_C_0.6	-365.16	-358.30	-358.30	0.00	1.88	1.88
051_B_one_C_0.6	-1,171.95	-1,168.98	-1,168.98	0.00	0.25	0.25
051_B_two_C_0.6	-2,797.13	-2,794.27	-2,790.27	0.00	0.10	0.25
072_B_p_two_C_0.6	-8.00	-19.40	-2.17	58.76	0.00	88.81
072_B_half_C_0.6	-379.09	-332.26	-329.44	0.00	12.35	13.10
072_B_one_C_0.6	-949.13	-949.13	-949.13	0.00	0.00	0.00
072_B_two_C_0.6	-2,231.77	-2,181.95	-2,180.54	0.00	2.23	2.30
016_B_p_two_C_0.4	187.61	175.27	190.18	7.04	0.00	8.51
016_B_half_C_0.4	78.12	65.59	80.69	19.10	0.00	23.02
016_B_one_C_0.4	-128.10	-128.91	-112.00	0.63	0.00	13.12
016_B_two_C_0.4	-542.79	-551.47	-497.40	1.57	0.00	9.80
026_B_p_two_C_0.4	195.04	191.24	202.04	1.99	0.00	5.65
026_B_half_C_0.4	-56.09	-70.36	-49.09	20.28	0.00	30.23
026_B_one_C_0.4	-472.85	-506.33	-467.62	6.61	0.00	7.65
026_B_two_C_0.4	-1,344.82	-1,378.30	-1,304.71	2.43	0.00	5.34
036_B_p_two_C_0.4	177.92	177.92	177.92	0.00	0.00	0.00
036_B_half_C_0.4	-95.45	-90.97	-90.97	0.00	4.69	4.69
036_B_one_C_0.4	-641.00	-655.95	-606.80	2.28	0.00	7.49
036_B_two_C_0.4	-1,698.42	-1,708.31	-1,629.09	0.58	0.00	4.64
045_B_p_two_C_0.4	250.06	311.15	311.15	0.00	24.43	24.43
045_B_half_C_0.4	-433.35	-355.05	-355.05	0.00	18.07	18.07
045_B_one_C_0.4	-1,616.63	-1,507.57	-1,507.57	0.00	6.75	6.75
045_B_two_C_0.4	-3,958.64	-3,878.12	-3,822.37	0.00	2.03	3.44
051_B_p_two_C_0.4	162.16	160.65	162.16	0.94	0.00	0.94
051_B_half_C_0.4	-296.04	-296.89	-296.04	0.29	0.00	0.29
051_B_one_C_0.4	-1,125.69	-1,090.51	-1,090.51	0.00	3.13	3.13
051_B_two_C_0.4	-2,679.38	-2,679.38	-2,679.38	0.00	0.00	0.00
072_B_p_two_C_0.4	-18.14	-16.63	-16.63	0.00	8.32	8.32
072_B_half_C_0.4	-381.47	-381.47	-381.47	0.00	0.00	0.00
072_B_one_C_0.4	-998.94	-998.94	-998.94	0.00	0.00	0.00
072_B_two_C_0.4	-2,233.91	-2,233.91	-2,233.91	0.00	0.00	0.00
016_B_p_two_C_0.2	282.41	269.77	286.74	4.69	0.00	6.29
016_B_half_C_0.2	172.92	172.92	177.25	0.00	0.00	2.50
016_B_one_C_0.2	-15.44	-19.78	-15.44	21.94	0.00	21.94
016_B_two_C_0.2	-405.18	-400.85	-400.85	0.00	1.07	1.07
026_B_p_two_C_0.2	347.80	347.80	347.80	0.00	0.00	0.00
026_B_half_C_0.2	104.44	122.41	128.56	0.00	17.21	23.09
026_B_one_C_0.2	-269.91	-219.84	-219.84	0.00	18.55	18.55
026_B_two_C_0.2	-1,098.13	-1,048.07	-987.17	0.00	4.56	10.10
036_B_p_two_C_0.2	277.89	284.77	284.77	0.00	2.48	2.48
036_B_half_C_0.2	-17.55	3.41	3.41	0.00	119.43	119.43
036_B_one_C_0.2	-517.06	-507.78	-507.78	0.00	1.79	1.79
036_B_two_C_0.2	-1,569.42	-1,530.07	-1,530.07	0.00	2.51	2.51
045_B_p_two_C_0.2	783.33	842.15	842.15	0.00	7.51	7.51
045_B_half_C_0.2	161.38	-0.30	-0.30	53,893.33	0.00	0.00
045_B_one_C_0.2	-915.25	-884.78	-882.19	0.00	3.33	3.61
045_B_two_C_0.2	-3,068.56	-3,171.12	-3,171.12	3.23	0.00	0.00
051_B_p_two_C_0.2	306.03	306.03	306.03	0.00	0.00	0.00
051_B_half_C_0.2	-139.63	-139.63	-139.63	0.00	0.00	0.00
051_B_one_C_0.2	-901.67	-901.67	-901.67	0.00	0.00	0.00
051_B_two_C_0.2	-2,425.68	-2,427.11	-2,425.68	0.06	0.00	0.06
072_B_p_two_C_0.2	241.61	251.86	251.86	0.00	4.24	4.24
072_B_half_C_0.2	-64.98	-53.79	-53.79	0.00	17.22	17.22
072_B_one_C_0.2	-588.88	-661.70	-661.70	11.00	0.00	0.00
072_B_two_C_0.2	-1,815.34	-1,815.34	-1,815.34	0.00	0.00	0.00

Chapter 6

Conclusions and Further investigations

In the present work we have presented some heuristic and exact approaches for the Single and Multiple Vehicle Routing Problems with Deliveries and Selective Pickups (SVRPDSP and MVRPDSP, respectively).

We have tested our approaches for the SVRPDSP with 68 benchmark instances proposed in the literature by [14]. Our computational tests with the EA have yielded much better results than our preliminary implementation of this algorithm and for more than half of the instances it has found better or equal solutions if compared to the algorithms of the literature, including 15 new solution and 7 proved optimals, against 3 of the GVNS proposed by [6], which were all found by the EA. Although the VND is not a match for the EA nor the GVNS it has found better or equal solutions than the other algorithms for about 1/3 of the instances, including 5 optimal solutions. Further comparison with proved optimal solutions found by the formulations have indicated that the GVNS has found greater number of optimal solutions, accounting for 23 instances, against 21 of the EA and 12 of the VND. However our EA has found in all 10 runs the optimal solution for 12 instances, against only 6 of the GVNS. An interesting fact is that in all cases where the VND has found the optimal solution, it has done so in all 10 runs. These results show that EA and VND were more robust than the GVNS.

Regarding the formulations, we have tested for all 68 instances our proposed Branch&Cut for the Gribkovskaia-Laporte-Shyshou (GLS) formulation and the novel formulation called Four-Commodity Network Flow ($4C$) along with the Süral-Bookbinder (SB) formulation presented in [30] with a time limit of 3 hours. The computational results indicates that our $4C$ formulation is the best if compared with the other two and has proved a total of 38 optimal solutions, against 8 of the Branch&Cut for the GLS and 1 of the SB. Although the GLS and SB formulations have proved

few optimal solutions, if we consider the number of optimals found by them, but not proved, the GLS has found 24 optimal values and the SB, 19. Notice that these results are still poorer than the ones onf the $4C$ formulation. In addition, comparing the lower bound values of each formulations, the $4C$ is statistically proved with 95% of confidence to have the best values, indicating that this formulation is tighter than the others. An interesting fact is that for instances of type `p_two` the formulation GLS has its performance greatly increased. We did not performed further investigations to check the cause of this behavior and propose this as a future work.

For the MVRPDSP we proposed two formulations, based on the SB and the $4C$ formulations for the SVRPDSP. Since the MVRPDSP is a novel problem, in order to test our proposed approaches we have generated a set of 72 instances based on the ones used for the single vehicle version of this problem. Further we have tested both formulations and our hybrid constructive heuristic setting a time limit of one hour. The computational results indicate that the models generate the best results when provided with an initial solution, since for many instances both formulations are not able to find a single feasible solution in the given time limit. Comparing the results of the formulations with an initial solution and the constructive we have found out that the solutions of SB have an average deviation from the best solution found by the 3 approaches of 1.69%, while $4C$ accounts for 2.59% and 5.62% of the constructive. In addition, both formulations and the constructive have proved 4 optimal solution. Therefore, regarding the two formulations the evidences indicate that both have similar performance. However the SB formulation has a downside, which is the high consume of memory that does not occur for the $4C$ formulations, thus for bigger instances we foresee that the $4C$ formulation would yield better results. Furthermore, we have notice that our hybrid constructive heuristic is able to generate good quality solutions, even optimals, since for our benchmark instances it has found 4 of them.

As future work for the SVRPDSP, we propose further investigations on our EA, for example, the impact of accepting unfeasible solutions in the population in order to expand the search space covered by the method. In addition we propose the implementation of incremental cuts for the GLS formulation, in which the formulation is relaxed and solved to optimality, at first, without any subtour elimination constraints, then if the relaxed optimal solution found has subtours, the model is solved again along with cuts for the subtours found in the relaxed optimal solution. The procedure continues until an optimal solution is found with no subtours.

The proposed 4C formulation has been proved to be the best of all formulations, even used as it is, without any integer programming solving technique. One can investigate if techniques like Lagrangean Relaxation, Benders Decomposition, Column

Generation and others are applicable.

For the MVRPDSP, we propose the investigation of further improvements on the formulations for the MVRPDSP, in special for the formulation $4C$, since it was the best for the SVRPDSP, but has not shown the same efficiency in the case of multiple vehicles. We also propose the adaptation of our EA for this problem and the implementation of other metaheuristics.

Finally in the following section we cite the publications originated so far from the research described in the present work.

6.1 Publications

Regarding the SVRPDSP we report the publication of the following papers.

IEEE Congress on Evolutionary Computation (IEEE CEC 2012)

- *Title:* Hybrid metaheuristic for the single vehicle routing problem with deliveries and selective pickups
- *Authors:* Bruck, Bruno P. ; Santos, André G.; Arroyo, José E.C.
- *Content:* In this paper we present our preliminary version of the Hybrid Evolutionary Algorithm
- *Submmited in:* January 2012
- *Presented in:* June 2012

Congreso Latino-Iberoamericano de Investigación Operativa/Simpósio Brasileiro de Pesquisa Operacional (CLAIO/SBPO 2012)

- *Title:* An Evolutionary Algorithm and a Variable Neighborhood Descent Algorithm for the Single Vehicle Problem with Deliveries and Selective Pickups
- *Authors:* Bruck, Bruno P. ; Santos, André G.
- *Content:* In this paper we present the final version of the Hybrid Evolutionary Algorithm (EA) described in section 3.5, and the Variable Neighborhood Search Algorithm (VND), described in section 3.6
- *Submmited in:* May 2012
- *Presented in:* September 2012

International Conference on Intelligent Systems Design and Applications (ISDA 2012)

- *Title:* Metaheuristics for the Single Vehicle Routing Problem with Deliveries and Selective Pickups
- *Authors:* Bruck, Bruno P. ; Santos, André G.; Arroyo, José E.C.
- *Content:* In this paper we present the final version of the Hybrid Evolutionary Algorithm described in section 3.5, a Iterated Local Search Algorithm (ILS) and a Variable Neighborhood Search Algorithm(VNS)
- *Submmited in:* September 2012
- *Presented in:* November 2012

Integer Programming and Combinatorial Optimization (IPCO 2013)

- *Title:* A novel integer programming formulation for the single vehicle routing problem with deliveries and selective pickups
- *Authors:* Bruck, Bruno P. ; Santos, André G.
- *Content:* In this paper we present our novel four-commodity network flow formulation for the SVRPDSP and the Branch&Cut algorithm for the Süral Bookbinder formulation
- *Submmited in:* To submit in October 2012

In addition we have submited the following paper reporting the results of our approaches for the MVRPDSP.

Hybrid Intelligent Systems (HIS 2012)

- *Title:* Hybrid approach for the Multiple Vehicle Routing Problem with Deliveries and Selective Pickups
- *Authors:* Bruck, Bruno P. ; Santos, André G.
- *Content:* In this paper we present our Süral-Bookbinder based formulation for the MVRPDSP along with the hybrid constructive heuristic.
- *Submited in:* Semptember 2012
- *Presented in:* December 2012

Bibliography

[1] Baldacci, R., Hadjiconstantinou, E., and Mingozzi, A. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research 52*, 5 (September 2004), 723–738.

[2] Boubahri, L., Addouche, S. A., and Mhamedi, A. E. Multi-ant colonies algorithms for the vrpspdtw. In *Proceedings of the 2011 International Conference on Communications, Computing and Control Applications* (Montreuil, France, March 2011), pp. 1–6.

[3] Brandão, J. A new tabu search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research 173*, 2 (September 2006), 540–555.

[4] Bruck, B. P., and Santos, A. G. An evolutionary algorithm and a variable neighborhood descent algorithm for the single vehicle problem with deliveries and selective pickups. In *Proceedings of the 2012 CLAIO/SBPO* (Rio de Janeiro, Brazil, September 2012).

[5] Bruck, B. P., Santos, A. G., and Arroyo, J. E. C. Hybrid metaheuristic for the single vehicle routing problem with deliveries and selective pickups. In *Proceedings of the 2012 IEEE Congress on Evolutionary Computation* (Brisbane, Australia, June 2012), pp. 910–917.

[6] Coelho, I. M. Contribuições para o problema de roteamento de veículos de rota única com entrega obrigatórias e coletas seletivas. Master's thesis, Universidade Federal Fluminense, April 2011.

[7] Crowder, H., and Padberg, M. W. Solving large-scale symmetric travelling salesman problems to optimality. *Management Science 26*, 5 (May 1980), 495–509.

[8] Dumitrescu, I., Ropke, S., Cordeau, J.-F., and Laporte, G. The traveling salesman problem with pickup and delivery: polyhedral results and a branch-and-cut algorithm. *Mathematical Programming 121*, 2 (April 2010), 269–305.

[9] Falcon, R., Li, X., Nayak, A., and Stojmenovic, I. The one-commodity traveling salesman problem with selective pickup and delivery: An ant colony approach. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation* (Ottawa, Canada, July 2010), pp. 1–8.

[10] Finke, G., and A Claus, E. G. A two-commodity network flow approach to the traveling salesman problem. *Numerantium 41* (September 1984), 167–178.

[11] Gajpal, Y., and Abad, P. L. Multi-ant colony system (macs) for a vehicle routing problem with backhauls. *European Journal of Operational Research 196*, 1 (July 2009), 102–117.

[12] Ganesh, K., and Narendran, T. T. Cloves: A cluster-and-search heuristic to solve the vehicleroutingproblem with delivery and pick-up. *European Journal of Operational Research 178*, 3 (May 2007), 699–717.

[13] Goetschalckx, M., and Jacobs-Blecha, C. The vehicle routing problem with backhauls. *European Journal of Operational Research 42*, 1 (September 1989), 39–51.

[14] Gribkovskaia, I., Laporte, G., and Shyshou, A. The single vehicle routing problem with deliveries and selective pickups. *Computers & Operations Research 35*, 9 (September 2008), 2908–2924.

[15] Gutiérrez-Jarpaa, G., Desaulniersc, G., Laported, G., and Marianove, V. A branch-and-price algorithm for the vehiclerouting problem with deliveries, selectivepickups and timewindows. *European Journal of Operational Research 206*, 2 (October 2010), 341–349.

[16] Hansen, P., and Mladenovic, N. Variable neighborhood search: Principles and applications. *European Journal of Operational Research 130*, 1 (May 2001), 449–467.

[17] Laporte, G., and Gribkovskaia, I. One-to-many-to-one single vehicle pickup and delivery problems. *The Vehicle Routing Problems: Latest Advances and New Challenges 43* (2008), 359–377.

[18] Mello, D. Mesmo com lei aprovada, logística reversa ainda não é realidade. http://exame.abril.com.br/meio-ambiente-e-energia/noticias/mesmo-com-lei-aprovada-logistica-reversa-ainda-nao-e-realidade.

[19] Miller, C. E., Tucker, A. W., and Zemlin, R. A. The single vehicle routing problem with deliveries and selective pickups. *Journal of the ACM 7*, 4 (September 1960), 326–329.

[20] Mingyong, L., and Erbao, C. An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows. *Engineering Applications of Artificial Intelligence 23*, 2 (March 2010), 188–195.

[21] Montané, F. A. T., and Galvão, R. D. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operations Research 33*, 3 (March 2006), 595–619.

[22] Nagy, G., and Salhi, S. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research 162*, 1 (April 2005), 126–141.

[23] Nemhauser, G. L., and Wolsey, L. A. *Integer and combinatorial optimization.* John Wiley & Sons, New York, 1988.

[24] Parragh, S. N., Doerner, K. F., and Hartl, R. F. A survey on pickup and delivery problems part i: Transportation between customers and depot. *Journal für Betriebswirtschaft 58*, 1 (April 2008), 21–51.

[25] Privé, J., Renaud, J., Boctor, F., and Laporte, G. Solving a vehicle-routing problem arising in soft-drink distribution. *Journal of the Operational Research Society 57*, 9 (September 2006), 1045–1052.

[26] Reimann, M., Doerner, K., and Hartl, R. F. *Insertion Based Ants for Vehicle Routing Problems with Backhauls and Time Windows*, vol. 2463. January 2002.

[27] Ropke, S., and Pisinger, D. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research 171*, 3 (June 2006), 750–775.

[28] Santos, H. G., Ochi, L. S., Marinho, E. H., and Drummond, L. M. A. Combining an evolutionary algorithm with data mining to solve a single-vehicle routing problem. *Neurocomputing 70*, 1 (December 2006), 70–77.

[29] Subramanian, A., Drummond, L. M. A., Bentes, C., and Farias, R. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research 37*, 11 (November 2010), 1899–1911.

[30] Süral, H., and Bookbinder, J. H. The single-vehicle routing problem with unrestricted backhauls. *Networks 41*, 3 (February 2003), 127–136.

[31] Vigo, D. Vrplib. `http://www.or.deis.unibo.it/research\_pages/ORinstances/VRPLIB/VRPLIB.html`.

[32] Zachariadis, E. E., and Kiranoudis, C. T. An effective local search approach for the vehicle routing problem with backhauls. *Expert Systems with Applications 39*, 3 (February 2012), 3174–3184.

[33] Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T. A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications 36*, 2 (March 2009), 1070–1081.