FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Optimization Algorithms for the Inventory Routing Problem

António Silva Fernandes

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Luís Filipe Ribeiro dos Santos Guimarães

27 de junho de 2016

Optimization Algorithms for the Inventory Routing Problem

António Silva Fernandes

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

27 de junho de 2016

Resumo

As cadeias de abastecimento são cada vez mais um ponto fulcral na redução de custos e consequente aumento da competitividade das empresas. A integração de toda a cadeia desde a produção ao consumidor final pode representar uma estratégia de redução de custos para todos os intervenientes na cadeia.

Grande parte dos problemas de investigação operacional, relativos ao tema em questão, apresentam-se com uma estrutura ou grau de dificuldade que nao permite uma resolução rápida e fácil. Desenvolve-se assim, a necessidade de encontrar métodos de apoio à decisão capazes de entregar soluções rápidas e eficazes.

O problema de rotas e inventário (IRP) apresenta-se com algum destaque na literatura, tendo em conta os proveitos conseguidos com o desenvolvimento de novas técnicas de abordagem. Tendo em conta que este problema integra duas grandes problemáticas de Investigação Operacional, apresenta-se com um elevado grau de complexidade.

Neste contexto, por forma a tentar dar resposta à necessidade de soluções alternativas a presente dissertação, apresenta-se como estudo acerca do problema de inventigação operacional - Problema de Rotas e Inventário. É apresentado um estudo intenso do estado da arte, remetendo para os vários métodos de resolução para problemas de otimização, mencionando alguns algoritmos exatos, mas dando mais enfase nos algoritmos aproximados, nomeadamente nas metaheuristicas. É proposto um modelo matemático para a resolução do problema de rotas e inventário com multi-veículos, multi-períodos e janelas temporais para efetuar as entregas. É considerado um horizonte temporal finito no qual se considera uma estrutura de um fornecedor e vários clientes sem rotas bi-partidas, assumindo que todos os clientes vêm a sua procura (préviamente conhecida) satisfeita sem entregas posteriores. São também apresentadas quatro variações do algoritmo GRASP com intuito de tentar resolver o problema em questão para uma versão reduzida de apenas um produto e um veículo. É desenvolvida uma variação do GRASP, o *Reactive* GRASP, de modo a melhor calibrar o algoritmo para cada instancia em estudo e uma aplicação de *Simulated Annealing*, tendo em vista a melhoria da pesquisa local em cada ciclo GRASP. Os resultados são avaliados recorrendo a intâncias da literatura comparando a solução obtida com um algoritmo di tipo Branch-and-Cut.

Abstract

Nowadays economic setting is characterized by high rates of competitiveness and and constant struggle for market shares. Supply chains represent an area of interest when talking about cost reduction policies as it may present enough flexibility to achieve considerable savings.

Metaheuristics tackle this demand for optimization as it returns good quality solutions taking into account the trade-off between solution quality and computational time.

This thesis presents an approximate algorithm to solve the Inventory Routing Problem. The problem consists on integrating the scheduling of the visits to each costumer while considering the trade-off between inventory costs and routing costs.

A mathematical model is proposed for the multi-vehicle multi-product inventory routing version with time-windows within a finite time horizon. It is considered a single supplier that serves multiple customers with no split deliveries, where demand has to be met without backlogging.

Four GRASP variations are developed to tackle a reduced version of the IRP with one vehicle and one product. The GRASP algorithm is modified into Reactive GRASP to better calibrate the GRASP procedure to each case. Furthermore, a Simulated Annealing is implemented to better search the widen the search-space in the local search phase. The final solutions achieved with the algorithms are compared with benchmark instances solved with a Branch-and-Cut algorithm.

Acknowledgments

I would like to thank my Supervisor professor Luís Guimarães for believing in me and in the project until the very end. I am truly thankful for all the teachings and hours spent with me. I shall cherish the knowledge wherever the future takes me.

I would also like to thank my fellow colleagues who accompanied me during this project. To Cristina thanks for putting up with me and for pretending to listen to all of my daydreams. To João Aires and Edgar Bicho for the nights spend with such inspiring philosophies and for all the help. To João Gomes, without your friendship and early help i would have taken much more time.

Finally, to my mother, thanks for the everyday teachings, thanks for making me who i am today, thanks for all the efforts, this thesis is for you!

António Fernandes

"With great power,
comes great responsibility"

Ben Parker

Contents

List of Figures

List of Tables

Abbreviations

VMI	Vendor Managed Inventory
IRP	Inventory Routing Problem
VRP	Vehicle Routing Problem
EVA	European Vending Association
MMIRP	Multi-Product Multi-Vehicle Inventory Routing Problem
MMIRPTW	Multi-Product-Multi-Vehicle-Inventory-Routing- Problem with Time-Windows
(VMIRP)	Vendor Managed Inventory Routing Problem
CP	Constraint Programming
ACO	Ant-Colony Optimization
GRASP	Greedy randomized Adaptive Search Procedure
LS	Local Search
ILS	Iterated Local Search
TS	Tabu Search
GA	Genetic Algorithm
SS	Scatter Search
PSO	Particle Swarm Optimization
TSP	Traveling Salesman Problem
VRPBTW	Vehicle Routing Problem with Backhauls and Time Windows
ML	Maximum-level
OU	Order-up-to-level
RGRASP	Reactive GRASP
SA	Simulated Annealing
GRASPSA	GRASP with SA
RGRASPSA	RGRASP with SA
CL	Candidates List
RCL	Restricted Candidates List

Chapter 1

Introduction

This document describes the thesis "Optimization Algorithms for the Inventory Routing Problem". In the first section the context and motivation that created the need for the study as well as the proposed objectives for the work are introduced. Chapter two presents the Literature Review related to previously done work concerning the approached subjects. In the third chapter is proposed a model to the IRP problem. The fourth chapter focuses on describing the developed work presenting the developed optimization algorithms. Chapter five will scrutinize the computational results presenting some previous conclusions. In the last chapter, final considerations are made regarding the developed work as well as future work.

1.1 Context and Motivation

In the last few years many European Countries have been facing economic crisis and in order to overcome it many cost-cutting policies have been developed and applied by many companies seeking to keep market shares, reduce operational costs and increase or maintain profits. Supply chains reflect how competitiveness responsibilities do not fall on only one entity as it represents a chain of operators who may depend on each other to survive. Among Supply Chain costs, logistics costs represent a big slice of many companies budget, and in order to tackle this issue, many entities apply what is called Vendor-Managed Inventory (VMI). VMI defines a collaboration between supplier and customer allowing the supplier to control when and how much to supply each customer. This embodies a win-win relation for both supplier and customer as it allows the supplier to better manage inventory while granting a better coordination of its entire fleet enabling an optimization of routes and removing the responsibility of scheduling deliveries from the customers equation. In order to accomplish a doable VMI, companies must have a solid Inventory and Routing plan.

Many are the industries that depend mainly in logistics operations, as such, this thesis will be developed as general as possible in order to be as adaptable to the many industries settings. Nonetheless a special focus will be given on solving problems arising in the Vending-Machines industry. This motivation comes from the fact that usual vending-machines companies operate

on a VMI strategy (Rusdiansyah and bi Tsao (2005)), being responsible to distribute and manage the stock in a large number of machines spread over a wide area. Using the example of the Portuguese group 'A Super 2000' which has over 2000 machines spread throughout the Portugal northern region, it is possible to understand the logistic challenges implied by such a large number of vending-machines (customer) and amount of products to deliver. According to The European Vending Association (EVA), there were in 2014, 295 million consumers who use machines at least once a week and 3.77 million vending machines in Europe allowing a turnover of €11.3 billion annually. These figures mean that cost-cutting policies can allow for large yearly savings, thus making this an economically interesting problem. As such the problem modeling will be oriented in a way that it may easily be applied to part of the supply-chain of such enterprise focusing on the single-depot multiple-customer model.

The Inventory Routing Problem (IRP) dates back thirty years (Coelho et al. (2013)) since Bell's seminal work (Bell et al. (1983)) and is one of the most demanding problems in operations management, integrating inventory management and vehicle routing decisions in order to minimize global logistics costs while satisfying customers demands. It allows to explore the trade-off between transportation/routing costs and inventory holding costs. This problem represents a much more realistic problem when compared with solutions that deal with both problems separately. This complex optimization problem started to be defined when the VRP community focused on developing heuristic solution approaches for routing problems considering inventory costs.

There have been many different approaches seeking to solve large amount of problems found in operational research. Among these are metaheuristics which start by in a first phase defining an initial solution leading to a second stage of improvement of that same initial solution using the most various methods. In order to find an initial solution one can use many different methods like branch-and-bound method, or its adaptation branch-and-cut, which have been found to be the most used ones in problems as IRP. In a second stage there is a need to improve the initial solution and just like in the first stage, many are the methods used, it is on the second stage that most authors diverge as it represents the most important component of the optimization problems. Exact algorithms, metaheuristics, matheuritics and many others can be used to improve this initial solution. This thesis will focus on developing a metaheuristic, even though it does not guarantee to find a global optimal solution it may present a good feasible solution in a shorter time then when compared with other methods.

1.2 Thesis Objectives

The thesis at hand proposes to tackle more realistic situations, considering multiple vehicles with heterogeneous holding capacities, delivering multiple products within time windows, this will permit a larger adaptation to the various logistics entities as vehicles have limited capacity to hold inventory and customers have limitations in availability to receive it.

In order to accomplish its proposed objectives this thesis aims to engage the IRP through the development of optimization algorithms and its comparison to state-of-the-art methods, in

particular the objective is to model the complete Multi-Product-Multi-Vehicle-Inventory-Routing-Problem with Time-Windows (MMIRPTW) and implement a metaheuristic-based algorithm capable of delivering reasonable solutions within a time-span as short as possible for a realistic scenario able to be used in practical cases.

Chapter 2

Literature Review

This chapter aims to analyze the state-of-the-art surrounding Optimization problems with focus on Inventory Routing problems. It will start by scrutinizing the different methods for solving optimization problems, giving examples of exact methods and approximated methods focusing on metaheuristics, which is the approach followed in the developed work, closing with different approaches to solve the problem at hand.

2.1 Optimization Problems

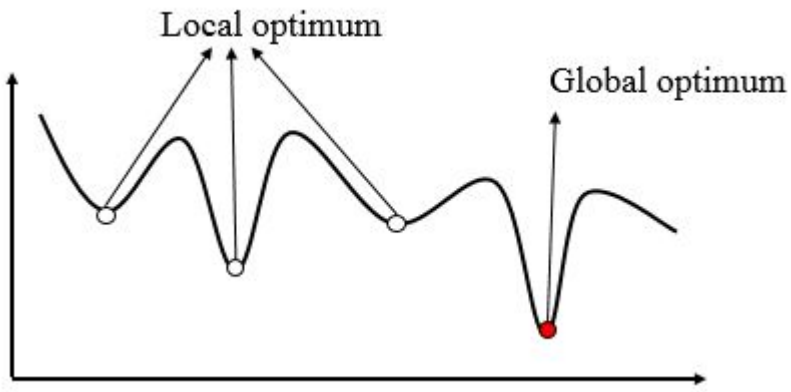
An optimization problem can be defined by an objective function and it's restrictions and has the goal of finding the best solution from all feasible solutions. To solve optimization problems there are many methods to be taken into account. In a first stage these can be divided into Exact Algorithms and Approximate Algorithms (Talbi (2009)), where the first ones obtain optimal solutions and the latter obtains high-quality feasible solutions in a reasonable computational time not guaranteeing the optimality of this solution.

Optimization problems may be solved using Exact methods or approximate methods. The first ones reach their optimal solutions guaranteeing their optimality. For bigger and more complex problems approximate algorithms may represent the wisest course of action, approximate methods do not guarantee a global optimal solution nor the optimality of the found solution but are able to find a high-quality feasionable solution in a reasonable computational time (Talbi (2009)).

Optimization problem can be classified into P or NP-hard problems, taking into account the size and structure of the instances and computational efforts needed to solve such problems, easier problems (P class) are often solved with exact algorithms which can deliver optimal solutions in a short computational time, more intricate problems with bigger or more complicated structured instances require an approximation approach which does not guarantee optimality.

When studying optimization problems, one must take into consideration the definitions of local and global optima (Figure 2.1):

Figure 2.1: Local and Global Optima



Definition 1. *The neighborhood $N(x)$ of x is the amount of solutions found in the search-space at a distance ε with $\varepsilon > 0$*

Definition 2. *A local optimum represents a solution $x \in S$, with S representing the search-space, which presents better quality then all its neighbors $s' \in N(x)$.*

Definition 3. *A global optimum represents a solution $x \in S$ which presents better quality then all other solutions in the entire search space, $s \leq s', \forall s' \in S$*

Having internalized such definitions it becomes obvious the need for algorithms capable of searching the search-space without getting caught in local optima.

2.2 Exact algorithms

Tree search algorithms such as Branch and X family and A* algorithms, Constraint Programming and Dynamic programming, search for a solution within the entire (interesting) search space, usually by subdividing the initial problem into smaller and simpler ones. Unlike approximated algorithms, exact algorithms obtain optimal solution guaranteeing optimality. When dealing with exact approaches, one must take into account some characteristics of the given problem such as instances size and also their structure. According to Talbi (2009) exact algorithms may be able to solve small sized NP-hard problems or even solve big sized ones depending on the structure of the instances, but generally exact algorithms are time-expensive, restraining the solutions of NP-hard problems (Jourdan et al. (2009)).

- **Dynamic Programming-** Is based on a recurrent formula that divides the problems in smaller ones. It uses a memory-based data structure to avoid recalculating the same sub-problems(Talbi (2009)). A Shortest Path Problem with Time Windows and Linear Node Costs was solved calling upon Dynamic Programming in Ioachim et al. (1998), solving a problem sized up to 700 nodes and 80,000 arcs highly outperforming an approach based on partial discretization of the time windows.

- **Branch and X-** This family of algorithms enumerates all solutions of a given optimization problem searching the search-space for the best feasible solution. Going through all possible solutions may become impracticable as the algorithms search tends to grow exponentially, this drawback is minimized with the use of upper and lower-bounds combined with the current best solution (Clausen (1999)). As seen in Desaulniers et al. (2015) the branch-and-x family of algorithms can represent a good approach for solving NP-hard problems, finding optimal solutions in a reasonable amount of computational time. Archetti et al. (2007) developed the first exact approach for the vendor managed inventory routing problem (VMIRP), using a a branch-and-cut algorithm adopting a best bound first strategy, solving up to 50 customers in 3 periods of time, and 30 customers for a time horizon equal to 6.

- **Constraint-programming-** Consists in a logic program built around tree search concepts. It relies on logical relations between variables to embed constraints between them (e.g. $x \leq 0$), and the optimization problem, reducing the search-space taking into account that all the constraints must be met to be able to find feasible solutions. Often found used in symbiosis with branch-and-bound, it presents a highly complementary nature Pesant et al. (1998). Constraint Programming has proven to be a good approach for many supply-chain problems such as in the traveling salesman problem with time windows developed in Pesant et al. (1998) who presents a CP approach that though it needs a great computational time, was able to reach to better solutions than some heuristics found in the literature. Another one is Aminu and Eglese (2006) who applied constraint programming to the Chinese postman problem with time-windows (CPPTW) that does not need to convert the initial problem into a vehicle routing problem with time windows (VRPTW) as usually approached in the literature, using a transformation of arc routing problems into node routing problems it was possible to achieve promising results.

2.3 Approximated algorithms

When dealing with approximate algorithms it is possible to consider a subdivision between approximation algorithms and Heuristics. As approximate, these do not guarantee a global optimum but, are able to state a provable solution while allowing high quality a trade-off between computational time and inherent costs for large size instances and may be applied to a large variety of problems. This provable solution is defined by performance guarantees Klein and Young (2010), which make possible to find an approximation factor ($optimalsolution \leq \varepsilon * s$) from which the found solution can be located from the optimum (Talbi (2009)).

Heuristics can still be divided into specific heuristics and metaheuristics, where the first ones are designed, as the name recalls, for specific problems and instances where the latter represents the focus of the last two decades in heuristics studies as it is considered a class of upper-lever algorithms, taking into account the flexibility of applications allowed and the quality of the solutions found.

Two of the most important concepts necessary to develop a metaheuristic are diversification and intensification. In diversification the goal its to explore the entire search-space in search of "good" regions while intensification focuses on exploring such region more thoroughly in search for the optimum. Hybrid metaheuristics methods represent one of the most promising approaches to tackle this trade-off as in these metaheuristics try to include both concepts in the same algorithm, presenting a good solution diversification and an intense search of each select region. When considering Metaheuristics, one must take into account many classification criteria (Talbi (2009)) such as:

- **Deterministic Vs Stochastic** - where the main differences lies in having pre-determined decisions or having probability embed into the search. In other words in a deterministic algorithm, if the initial solution stays the same, the algorithm will always return the same answer, while stochastic algorithms may return different results.

- **Non-nature Vs Nature inspired algorithms** - Some algorithms are modeled based on natural events such as: Ant-colony optimization which is based on the way ants communicate when searching for food. Genetic Algorithms which applies a population which just like in nature evolves through time, favoring the "fittest" . Industrial processes like the simulated annealing that transposes the metallurgy annealing term to algorithm considering a first stage of high temperature (more probability for accepting for unfeasible solutions) followed by a period of slow cooling (less and less probability of accepting disruptive solutions).

- **Memory usage Vs memory-less methods** - Many metaheuristics use information based on past search to evolve (memory usage), ACO uses a pheromone matrix to find the best path and Tabu-search memorizes a number of "tabu-moves" that should be avoided. Others use no information from the search (memoryless), Local Search usually focuses locally and by doing so is only interested in optimizing the actual values.

- **Iterative Vs Greedy** - Iterative cases try to improve a solution by performing predetermined moves at each iteration. Greedy ones such as "Greedy Randomized Adaptive Search Procedure" (GRASP) starts with no initial solution adding decision variables as it searches, reducing the space-state and focusing on the local optimum.

- **Single-solution based Vs Population-based search** - single-solution based search methods, start with one initial solution and try to improve it at each step while population-based ones start off with a population of solutions which also keeps evolving. The main direct consequence of this choice lies in the fact that by definition, single solutions-based algorithms focus on improving locally ignoring the quality of the region, which also makes it highly dependent of the initial solution. While population-based methods present a broader search of the solution-space as it evolves entire populations of solutions, which as no dependency of the initial solution.

The choice of using a Metaheuristic to solve an optimization problem must also consider numerous factors (Talbi (2009)) such as:

- Complexity of the problem which reflects it's hardness.
- The size of the instances, as NP-hard problems with small instances are viable to be solved with exact algorithms.
- Structures of the instances. With specific structures it is possible to solve large size instances.
- Search time needs to be accounted while designing the algorithm, where some focus on being fast-searching to be able to deliver the best solution possible in a relatively short time-span, and others focus on guaranteeing the optimality and disregarding the importance of reducing computational time.

It is redundant to use Metaheuristics to try to solve problems which can easily be solved to optimality or where exact algorithms present a reasonable search time, in other words Metaheuristics are "reserved" for NP-Hard class problems that cannot be reduced to smaller and/or easier problems. On the other hand polynomial problems may justify the use of Metaheuristics taking into account the search time needed to solve them with other strategies.

2.3.1 Single-Solution Based Algorithms

Single-Solution based algorithms, iteratively improve the quality of one solution at a time by generating an initial solution and searching the neighborhood for improvements which will replace the initial solution until a stopping criteria is satisfied.

The main setback in this method lies within the size of the neighborhood as the solution output will vary depending on this factor. When considered a small neighborhood the algorithm will tend to be more effective as it be able consider all possible solutions in said space, but with a wider neighborhood such will not be possible thus deteriorating this method's effectiveness. Single-solutions Metaheuristics focus on intensification of the solutions, which means that for a given neighborhood, the algorithm will scrutinize the nearby solutions trying to find to local optimum, which may not represent the global optimum, in other words, if by itself, this method depends mostly of the quality of the initial solution location in search-space. In order to overcome this set-back many studies in the literature suggest the use of disturbing methods to change the neighborhood in order to find a new improved local optimum. The most common examples of Single-solution based Metaheuristics are:

- **Local Search (LS)-** consists in cruising the search space by iteratively performing modifications (which can be achieve by many different methods) to the initial solution trying to find an optimum. This deterministic method focuses on intensification of the search-space which means it does not necessarily take into account the quality of the region in which it performs

the search. Wang et al. (2016) was able to approach a machine reassignment problem (MRP) reassigning processes to improve the use of machines while satisfying many constraints. In he's work it is proposed an effective multi-neighborhood local search (MNLS) which uses three different types of neighborhoods: one process shift, two processes swap and three processes swap; employing a partitioned technique to simplify and reduce the neighborhoods sizes. After finding the the best possible solution with MNLS the algorithm disturbs the solution in order to form a different solution as initial solution for the next iteration. For this disruption the algorithm uses a Tabu Search (TS) methodology, to avoid local optimums and improve the diversification. In fact all variants of local search present the set-back of focusing on intensification of the solution quality, disregarding the diversification, without a disturbance method, this will induce the algorithm to be stuck in a local optimum.

- **Iterated-local search (ILS)-** Local search methods are viable to get caught in a local minimum, not being able to improve the current solution. ILS deals with this problem by disrupting the local search generating different initial solutions which will then be locally searched for improvement. In other words ILS complements LS by integrating a random or constructive heuristic to generate different initial solutions. In Vansteenwegen and Mateo (2014) was developed a metaheuristic which was able to solve a Single-Vehicle Cyclic Inventory Routing Problem (SVCIRP) outperforming the compared literature and thus proving the quality of the ILS method in solving NP-hard routing problems. Having found 32 new optimal solutions for the used benchmark, and improving running times.

- **Tabu Search (TS)-** Ts consists "in a kind of ILS and is characterized by the use of flexible memory" Pham and Karaboga (2012). It escapes local optima using disruptive methods and selects the best neighbor solution, even if represents a worst solution when compared with the initial one (global optimum), keeping in memory a list of last feasible moves and solutions found, which will be classified as forbidden as long as they are kept in the Tabu-list. This will prevent cycles created by performing repeatedly the same moves for the same solutions (Griffis et al. (2012)). Tabu-search is highly viable to be worked along side with many different methods. Zeng et al. (2016) verifies the profitable symbiosis between TS and Variable Neighborhood Descent (VND); In Lai et al. (2016) it is possible to analyze the fruitful contribution to the study of time-constrained heterogeneous vehicle routing problem on a multigraph where parallel arcs between pairs of vertices represent different travel options based on criteria such as time, cost, and distance" (Lai et al. (2016)). It represents a good solution to approach the trade-off of intensification and diversification as TS search has preference for the "elite" which are solutions in possession of common attributes that result in a local optimum thus applying an intense search for local optima (Talbi (2009)), and it also considers Diversification as it keeps in memory a list of visited regions forbidding the algorithm from embarking in the same neighborhoods.

2.3.2 Population-based Algorithms

Population-based metaheuristics like evolutionary algorithms (EA), scatter search, particle swarm optimization or Ant-colony Optimization (ACO), also rely on a method to improve an initial solution, but unlike Single-solution based metaheuristics, Population-based do not improve just one solution at a time but instead, an entire population of individual solutions, evaluating pools of solutions for the pretended attributes (Griffis et al. (2012)). The method consists in firstly creating an initial population of solutions, then focuses in generating another population of solutions which will integrate the first one and, taking into account the selection methods in place, evolve into a population with the best specimen from both populations. This process will be repeated until a stopping criteria is met. There are many approaches for population-based metaheuristics, which normally differ from one another in the methods used for generating and selection of solutions. Unlike single-solution metaheuristics which mostly focuses in intensification of the search space, population-based metaheuristics focuses in diversification due to the large amount and associated diversity of initial populations (Talbi (2009)) , which means the method uses a population of possible solutions to spread the range of search of the search-space.

A major concern with population-metaheuristics is the diversification of the individuals of the the initial population as the risk of convergence shrinks with the rise of diversification (Talbi (2009)). Population-based metaheuristics not only have the common concepts of all metaheuristics like representation, objective function and constraint handling, but also present common concepts for all population-metaheuristics such as the definition of the initial population and stopping criteria. With this defined it becomes possible to consider two types of population-metaheuristics:

2.3.2.1 Evolution-based

In this class of Population-based algorithms, populations will be improved by applying variation operators such as mutation to the initial solution, in other words at each iteration new generations of solutions are fostered by the ones before, evolving from the most fit individuals of the pool of solutions Talbi (2009) just like in nature where two parents are necessary to give birth to a new generation. In evolutionary metaheuristics the most pressing concepts are Selection and Variation, as these are the concepts which will define the search for different populations. Some examples of this method of population-metaheuristics are:

- **Genetic Algorithms(GA)-** This class of evolutionary algorithms inspires its' approach in evolutionary biology. Each individual of the population is uniquely defined by its chromosomes and thus its genome. As stochastic population-based metaheuristic in each generation an evaluation of each individual's fitness is performed. Using this attribute a group of the most fit is selected for recombination, mutation and other methods of modification (Griffis et al. (2012)). The algorithms runs until stopping conditions are met, which usually are when a good quality solution is reached or the number of maximum generations is met. GA tend to be highly oriented towards diversification algorithms though sacrificing accuracy to

get better computational times (Davies and Lingras (2003)), which is many times considered a key aspect to take into consideration. One of the most pressing problems in GA is the risk of rapid convergence into the same genotype, instead, the desirable is maximum diversification so the search may continue (Grefenstette (2014)).

A good example of GA application is Lu and Huang (2015) where a method called corner space algorithm is applied as placement procedure in a GA, optimizing the Cutting-Stock Problem (CSP) embed in the TFT-LCD industry; the developed algorithm found better results then the ones compared with from the literature, presenting it self more efficient and effective.

- **Scatter Search (SS)-** Scatter Search is a deterministic population algorithm (Glover et al. (2000)), as usual in population-based metaheuristics, it requires an initialization of the population of solutions; a reference set is then constructed using the most fit solutions which will take a part in creating new populations; even if it is not possible to improve the initial solution (local optimum) the reference set keeps in memory second-grade solutions to allow a good dispersion and diversification of the search-space; this solutions are then improved with help of a single-solution metaheuristic or other search-intensification method; the population and the reference set are then updated and the procedure iterates until meeting the stopping criteria. (Talbi (2009)).

  Scatter search framework allows some flexible approaches, as in Russell and Chiang (2006) where it was developed a Scatter search framework for the vehicle routing problem with time windows (VRPTW), obtaining 6 new best solutions in 56 test problems, presenting more efficiency for VRPs with a small number of customers per route.

2.3.2.2 Blackboard-based

In Blackboard-base class of metaheuristics the construction of the shared memory and the creation of a solution based in that shared memory occupy the position of interest (Talbi (2009)), as every population contributes to the shared memory and new pools of solutions are created taking into account the information kept in memory and not generated from the attributes of their "parents".

In this class fit algorithms such as:

- **Ant-colony Optimization (ACO)-** In the insect world, ants assemble in colonies searching for food while taking advantage of a big number of individuals. An ACO just like in nature works with big sized populations where each individual (just like ants) leaves behind a trail of pheromones, making it more probable to each individual to follow a trail as more pheromones were left behind (usually stands for more food) (Griffis et al. (2012)). As time periods pass, individuals (ants) pheromones lose their intensity (evaporate) being discarded after some periods. Selecting shortest paths to food imply higher values of pheromone in the trails. Taking into account that ACO is a stochastic method (Talbi (2009)), there is the probability of some individuals selecting not traveled trails, if shorter, the trails will show a

high concentration of pheromones, as the individual takes less time to travel and thus making such a route more probable to be accepted (Griffis et al. (2012)); this way is possible to guarantee a wide-spread exploration of the search-space. Xiang et al. (2015) proposed an ant colony optimization approach for solving an operating room surgery scheduling problem. Puris et al. (2010) proposes a two phase ACO showing better results than the best know solution at the time for the TSP and and QAP (quadratic assignment problem).

- **Particle Swarm Optimization (PSO)-** PSO is a stochastic population-based metaheuristic Talbi (2009) in which problem solving system mimic collective intelligence in nature (Alam et al. (2015)). Each particle in a population represents a possible solution to the problem and is defined by its position where each position has its fitness evaluated by the optimized fitness function (Sethanan and Neungmatcha (2016)) and velocity which redirects its movement (replicating birds behavior when flying). Each particle adjusts it is position taking into account two previous solutions: the best personal value and the best populational value (Talbi (2009)) which are also kept in memory and thus allowing a quick convergence to global optima (Lynn and Suganthan (2015)). In Belmecheri et al. (2013) is presented a PSO algorithm with a local search for the vehicle routing problem with heterogeneous fleet, mixed backhauls and time windows (HVRPMBTW), which when comparing its results with an exact method and best know solutions of the literature for specific VRPTW problems, returns that HVRPMBTW achieves optimal solutions for small problems and was able to improve the GAP, which measures the distance to the optimal solution, by over 5.5% with 29 of 56 cases displaying its quickness in convergence, few parameter settings and fewer memory needed.

2.3.3 Multi-objective Metaheuristics

In many decision problems faced in life and fields of research such as engineering, logistics, economics etc. problems often come with more than one objective (e.g. minimizing production costs while maximizing the product's efficiency; Maximizing profits while reducing transportation and holding costs). The objective function in multi-objective metaheuristcs (MOMs) is represented by a vector in which many other objectives are contained $F(X) = (f(y), f(w), f(z), \ldots (n))$, which means many of these objectives may be competing between each other. The main goal of multi-objective metaheuristics resides in optimizing all objectives without degrading any other (Talbi (2009)), which can be rephrased into finding the Pareto Optimal set or in other words find the solutions vectors that allow such constraints where the optimum of the integrated system consists in a trade-of of multiples optimums. Evidently each problem is different and the weight of each objective must be studied for each situation permitting a trade-off between objectives suitable to each situation. An optimum is achieved when it no longer possible to improve an objective without deteriorating any other (Talbi (2009)). The number of Pareto's optimal solution increases with the amount of objectives and the size of the problem Talbi (2009), which is commonly large in NP-hard problems, as is the IRP.

The pareto front consists in a set of the most efficient solutions found, which means a set of points able to form a convex on concave front.

MOMs need to tackle the representation, constraint handling and other concepts just as common metaheuristics, the differentiation concepts are fitness assignment, diversity preservation and elitism.

2.3.4 Hybrid Metaheuristics

As stated before, there are many possible methods to try to solve optimization problems. Many of the solutions proposed require simplifications of the problem or relaxation of constraints. Single solution metaheuristics focus in intensifying the search around a neighborhood while population-based try to diversify the neighborhoods, but there exists no standard method which is capable of doing both, thus arising the need for methods capable of dealing with trade-of between intensification and diversification. Hybrid metaheuristics allow an enhancement of strengths and decreasing of weaknesses of the hybridized methods (Baños et al. (2013)) making it possible to present better results than the ones achieved without the hybrid method.Considering the classification criteria proposed by Talbi (2009) and Jourdan et al. (2009), it is possible to define the following approaches o hybridization:

- low-level: Addressing the functional composition where a specific function of a metaheuristic is replaced by a different one (Jourdan et al. (2009) Talbi (2009)), referenced as integrative metaheuristics by Lozano and García-Martínez (2010).

- high-level: Different metaheuristics are self-contained (Talbi (2009) Jourdan et al. (2009)) having no relationship allowing independence between them. Lozano and García-Martínez (2010) refers to this criteria as collaborative metaheuristics.

And on a second level:

- Relay: Where different methods are applied in sequence, where the firsts output is the seconds input (Talbi (2009) Lozano and García-Martínez (2010)).

- Teamwork: Where different methods are applied in parallel, allowing cooperation between the different methods.

This assortment proposed by Talbi (2009) results in the following classification:

- low-level relay hybrid - in which a metaheuristic is embedded into a single solution metaheuristic. In Küçükoğlu and Öztürk (2015) is possible to learn about a hybrid approach which combines SA and TS fo the for the vehicle routing problem with backhauls and time windows (VRPBTW) obtaining very good results finding 34 new best solutions for 45 instances while presenting a reduce computational time.

- low-level teamwork hybrid - in which a metaheuristic is embedded into a population based metaheuristic. In Baños et al. (2013) is presented the development of a multi-start multi-objective evolutionary algorithm with simulated annealing as acceptance criterion embedded in an Evolutionary algorithm for the VRPTW achieving competitive results when compared to those of the literature.

- high-level relay hybrid - in which the self-contained metaheuristics work in series where the first ones output represents the seconds input (Jourdan et al. (2009)). In Bent and Hentenryck (2004) is presented a two-stage hybrid algorithm for the VRPTW, starting by minimizing the number of routes using simulated annealing and then he travel costs by using a large neighborhood search, when tested with Solomon benchmarks the hybridized metaheuristic returned an improvement of 10 out of 56 best published solutions presenting a highly robust algorithm presenting the benefits of a two-stage approach.

- high-level teamwork hybrid - in which the different algorithms work in parallel sharing information to improve each others performance. In Nwana et al. (2005) is studied a hybrid method which implements a B&B algorithm cooperating with SA improving the method suggested by Connolly (1992), outperforming this method for the test problems considered.

2.4 GRASP

This section will present a more focused analysis on the Greedy Randomized Adaptive Search Procedure as it was the elected method to solve the problem at hand. GRASP was first introduced in Feo and Resende (1989) and consists on a multi-start, random and iterative procedure where each cycle consists firstly in a Construction Phase which iteratively constructs an initial feasible solution, sampling the search-space in search for good regions(diversification) and a second where LS is applied to the generated solution in search of local optimum(intensification), in the end the best overall solution is kept as the result (Duarte et al. (2015)).

Many authors have already applied GRASP to the most varied combinatorial optimization problems, such as Resende (1998) who presents a detailed explanation of the basic GRASP and Feo and Resende (1995) in where is presented the many enhancements that GRASP can suffer such as parallelization in order to reduce computational times, hybridization with other complementary optimization method and the Reactive GRASP. In fact parallelization is a very common and appropriate methodology to implement in GRASP as each iteration is independent from one another, making it possible to be run in different threads ((Cung et al., 2002)). Many are the industries and activities in which GRASP as been applied such as: Scheduling (Feo et al. (1996)), Routing (Kontoravdis and Bard (1995)), Location (Holmqvist et al. (1997)), Manufacturing (Bard and Feo (1989)), among others.

2.5 Inventory Routing Problem Review

The IRP was firstly tackled in 1983 by Bell et al. (1983) who took into account transportation costs as well as customers demand. Since then the problem has been defined as an integration of the VRP and Inventory Management, in other words deciding whom to serve and when, how much to deliver while taking into account the route to be traversed. Many were the authors who proposed different methods of approaching this problem, Archetti et al. (2007) proposed the first exact branch-and-cut approach to solve the deterministic order-up-to level IRP and was then followed by Coelho and Laporte (2013a) who proposed an exact B&C algorithm for a multi-product multi-vehicle variation of IRP. Being an NP-hard problem and many times presenting big-size instances or not so easily structured instances, most authors implement their solutions resorting to Heuristic methods, mainly metaheuristics, Park et al. (2016) proposes a GA to solve the IRP with lost sales, while Qin et al. (2014) proposes a LS with TS to optimize this problem.

There are many industries which benefit from these studies, as these should present as real-life applications as it is possible, but many of these proposed solutions are often relaxed preventing its applications. When developing a solution for the IRP, one must take into account some problem characteristics such as Time Horizon which can be finite or infinite; Products quantity as many suppliers will only produce one product and others may produce a large quantity; Structure of the problem, as it can illustrate an assembly factory which receives multiple products to be assembled in one location, or as in the Vending-machines situation where a central depot as to supply many different machines, or as in maritime cases where many vessels transpose materials from many depots into others; Routing which can be direct or multiple, where in the first case it means a vehicle can only supply one customer per route and in the latter it can supply more than one; Inventory policies which can be characterized as Maximum-level (ML) or Order-up-to level (OU) which means, in the first case that the inventory delivered is such as the needed to achieve maximum holding capacity of each customer and in the second case is calculated as the difference between a balanced level and the demand in each period; Inventory decisions where is possible to distinguish between ignoring a deliver which was not possible to perform on time (Lost Sales) or the possibility of delivering once the conditions are viable again (Backlogs); Fleet composition which differentiates homogeneous or heterogeneous vehicle capacities; The quantity of vehicles;

The following Tables 2.2 and 2.1 compare approximate and exact method, respectively, used for the IRP, following the enunciated criteria.

Table 2.1: Exact methods

Reference	Time Horizon		Products		Structure			Routing		Inventory Policy		Inventory Decision		Fleet Composition		Fleet Size		Method
	Finite	Infinite	Single	Many	Many-to-one	One-to-many	Many-to-many	Direct	Multiple	ML	OU	Lost Sales	Backlogging	Homogeneous	Heterogeneous	Single	Multiple	
Coelho and Laporte (2013a)	x			x		x			x	x		x			x		x	B&C
Coelho and Laporte (2013b)	x		x			x			x	x		x		x			x	B&C
Archetti et al. (2007)	x		x			x			x		x			x		x		B&C
Desaulniers et al. (2015)	x		x			x			x					x			x	BP&C & TS
Mirzapour Al-e hashem and Rekik (2014)	x		x			x			x	x	x	x		x	x		x	B&C
Bredström et al. (2015)	x			x	x				x						x		x	Coin-OR (B&C)
Solyali and Süral (2011)	x		x			x			x		x	x		x		x		B&C

Table 2.2: Approximate methods

Reference	Time Horizon		Products		Structure			Routing		Inventory Policy		Inventory Decision		Fleet Composition		Fleet Size		Method
	Finite	Infinite	Single	Many	Many-to-one	One-to-many	Many-to-many	Direct	Multiple	ML	OU	Lost Sales	Backlogging	Homogeneous	Heterogeneous	Single	Multiple	
Bard and Nananukul (2009)	x		x			x			x		x		x	x			x	2 step Heuristic
Archetti et al. (2012)	x		x			x			x	x	x			x		x		Hybrid TS
Bertazzi et al. (2002)	x		x			x			x		x			x		x		2 step Heuristic
Coelho et al. (2012b)	x		x			x			x	x	x	x		x		x		ALNS
Coelho et al. (2012a)	x		x			x			x	x	x	x		x			x	ALNS
Park et al. (2016)	x		x									x		x			x	GA
Qin et al. (2014)	x		x			x			x						x		x	TS
Song and Furman (2013)	x		x				x		x						x		x	LNS & B&C
Bertazzi et al. (2013)	x		x			x			x		x	x		x		x		Hybrid
Moin et al. (2011)	x			x	x			x				x		x			x	Hybrid GA
Askin and Xia		x	x			x			x	x		x		x			x	MS,TS-SA,ILS,HGA
Mjirda et al. (2014)	x			x	x				x	x		x		x			x	VNS

Chapter 3

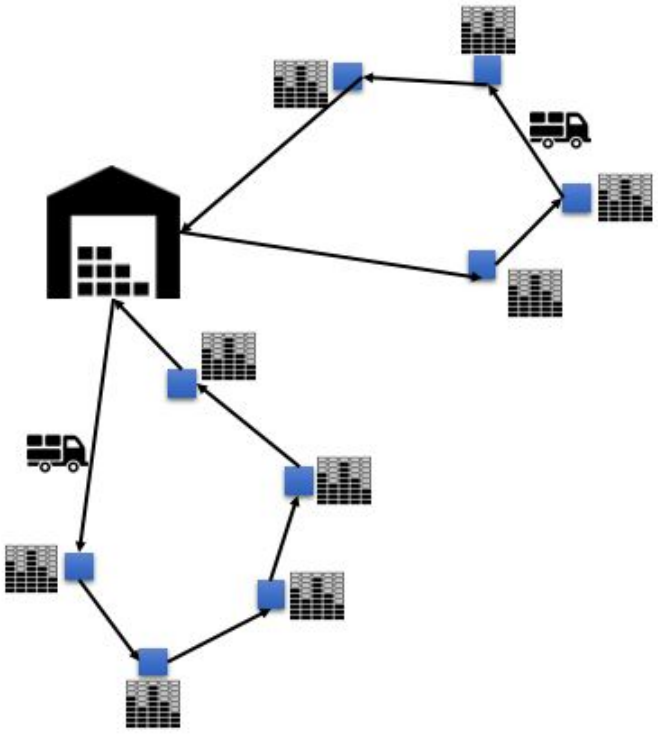
Problem Description

This chapter will address the work developed in modeling the problem, presenting the variables and parameters used, as well as the mathematical expressions that define the problem at hand.

As stated before, the IRP is characterized by tackling the VRP and Inventory Management producing an integrated solution presenting as main objective the reduction of distribution and holding costs without any stockouts throughout the planning horizon.

For the routing problem (Figure 3.1) the proposed approach will search for the best route that each vehicle can perform covering all customer set to be visited in each period.

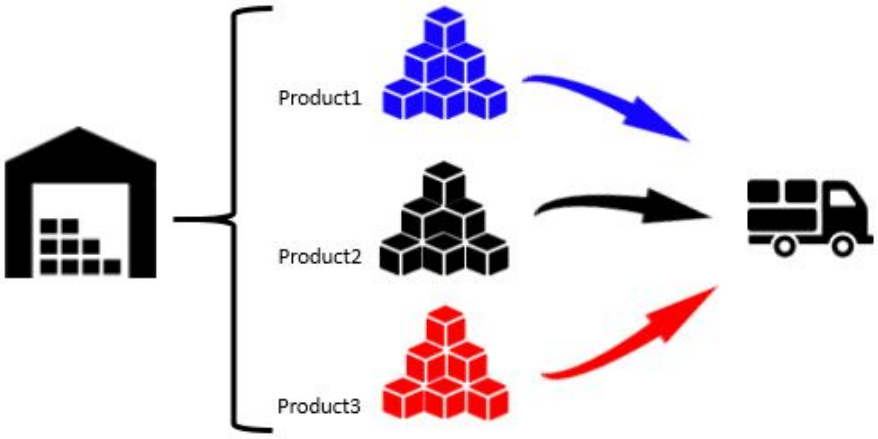
Figure 3.1: Routing



This will necessarily be integrated with Inventory management (Figure 3.2) as each vehicle

will supply the needed quantities of products in order to balance the inventory levels (following ML or OU policies). The solution to this sub-problem will consist in the amount of inventory of each product that goes to each vehicle taking into account there are no split deliveries and also considering the storage capacity at the supplier, customer and vehicles.

Figure 3.2: Inventory assignment



It is also interesting to consider a third sub-problem of when to perform each delivery. Bearing in mind the previous two sub-poblems, it becomes perceptive how the deliveries timing will affect the the overall solutions. Following the Just-in-Time methodology every distribution should be realized in the instant the customer needs the stock as to minimize the inventory costs. Nevertheless, when looking from a bigger perspective it is sometimes preferable to guarantee a specific amount of stock a few periods earlier possibly allowing for a cost reduction in the routing sub-problem. As such, the inventory policy applied int this case is such that ensures no lost sales and when visited, a customer will be supplied with the exact inventory needed to fulfill the demand up until the next visit, once again taking into consideration the capacity constraints.

The IRP is an NP-hard problem which has induced many practical simplifications. In the past, approaches often over-simplify the problem by considering only one vehicle or not considering time-windows which are nowadays present in most supply chains. Such simplified approaches are sometimes needed to be able to deliver an algorithm capable of solving problems in reasonable time. Nevertheless, a more realistic model is proposed as to set a base for future work.

For this model it is considered a supply chain comprised by a single-supplier and multiple customer. Each vehicle may perform an one-way route to satisfy the customers needs for multiple products. It is also assumed that each customer has a time-window in which it is able to welcome new stock taking into account service times for those deliveries.

3.1 Problem Notation

The IRP is usually defined as a graph $G = (V,A)$, where $V = \{0,\ldots,n\}$ is the vertex set, $A = \{(i,j) : I,j \in V, i \neq j\}$ the arc set and the node $\{0\}$ represents the supplier.

Table 3.1 will present the Indexes ad Sets, Table 3.2 Parameters and Table 3.3 Decision Variables, chosen to formulate the problem at hand:

Table 3.1: Indexes and Sets

Parameter	Description
n	Number of vertexes in V
$i,j = \{1,\ldots,n\}$	Customers
(i,j)	Arc $\in A$
t	Subdivided time periods of T
k	Vehicle $\in K$
p	Product $\in P_i$
$V\{n\}$	Vertex set
$A\{(i,j)\}$	Arc set
$T = \{1,...,t\}$	Set of t time periods of the planing horizon T
$K = \{1,\ldots,k\}$	Set of k vehicles of K
$P_i = \{1,\ldots,p\}$	Set of p products contained in P

Table 3.2: Parameters

Parameter	Description
h_i	inventory holding cost incurred by keeping one unit of any product at customer i
c_{ij}	cost incurred when travelling from node i to node j
Q_k	capacity of vehicle k
r_p^t	quantity of product $p \in Pi$ made available at the supplier in period t
C_i	holding capacity at customer i
I_{p0}^t	product $p \in Pi$ inventory level at supplier's at the end of period t
D_{pi}^t	demand for product p at customer i in period t
$[s_i;f_i]$	time-window for deliveries at customer i
w_i	service time at customer i
rt_{ij}	time that a vehicle takes to get from vertex i to vertex j

Table 3.3: Decision Variables

Variable	Description
q_{pi}^{kt}	quantity of product $p \in Pi$ delivered to customer i by vehicle k in period t
I_{pi}^t	customer's product $p \in Pi$ inventory levels at the end of period t
x_{ij}^{kt}	equal to the number of times edge (i,j) is used on the route of vehicle k in period t
y_i^{kt}	equals to one if and only if vertex i (supplier or customer) is visited by vehicle k in period t

Having defined the variables and parameters, it's now possible to formulate the problem.

3.2 Mathematical Formulation

The IRP can be formulated as a Mixed Integer Programming Model as follows:

$$\min \sum_{i\in V}\sum_{p\in Pi}\sum_{t\in T} h_i \cdot I^t_{pi} + \sum_{i\in V}\sum_{j\in V, i<j}\sum_{t\in T}\sum_{k\in K} c_{ij}\cdot x^{kt}_{ij} \tag{3.1}$$

Subject to:

$$I^t_{p0} = I^{t-1}_{p0} + r^t_p - \sum_{k\in K}\sum_{i\in V\setminus\{0\}} q^{kt}_{pi} \quad , \forall t \in T, \forall p \in Pi \tag{3.2}$$

$$I^t_{pi} = I^{t-1}_{pi} + \sum_{k\in K} q^{kt}_{pi} - D^t_{pi} \quad , \forall i \in V\setminus\{0\},\ \forall t \in T, \forall p \in Pi \tag{3.3}$$

$$\sum_{p\in Pi} I^t_{pi} \le C_i, \quad \forall i \in V\setminus\{0\}, \forall t \in T \tag{3.4}$$

$$\sum_{p\in Pi}\sum_{k\in K} q^{kt}_{pi} \le C_i - \sum_{p\in Pi} I^{t-1}_{pi} \quad , \forall i \in V\setminus\{0\}, \forall t \in T \tag{3.5}$$

$$q^{kt}_{pi} \le C_i \cdot y^{kt}_i \quad , \forall i \in V\setminus\{0\}, \forall t \in T, \forall p \in Pi \tag{3.6}$$

$$y^{kt}_i = \sum_{j\ne i} x^{kt}_{ji} \quad , \forall i \in V, \forall t \in T, \forall k \in K \tag{3.7}$$

$$y^{kt}_i = \sum_{i\ne j} x^{kt}_{ij} \quad , \forall i \in V, \forall t \in T, \forall k \in K \tag{3.8}$$

$$s^{tk}_j + w^{kt}_j + rt^k_{ji} - (1-x^{kt}_{ji})M \le s^{kt}_i \quad , \forall i \in V\setminus\{0\}, \forall j \in V\setminus\{0\}, \forall k \in K, \forall t \in T \tag{3.9}$$

$$s_i \cdot y^{kt}_i \le w^{kt}_i \le f_i \cdot y^{kt}_i \quad , \forall i \in V\setminus\{0\}, \forall j \in V\setminus\{0\}, \forall k \in K, \forall t \in T \tag{3.10}$$

$$I^t_{pi} \ge 0 \quad , \forall i \in V, \forall t \in T, \forall p \in PI \tag{3.11}$$

$$q^{kt}_{pi} \ge 0 \quad , \forall i \in V\setminus\{0\}, \forall p \in Pi, \forall t \in T, \forall k \in K \tag{3.12}$$

$$I^t_{p0} \ge 0 \quad , \forall t \in T, \forall p \in Pi \tag{3.13}$$

$$x^{kt}_{ji} \in \{0,1\} \quad , \forall i \ne j, (i,j) \in V, \forall k \in K, \forall t \in T \tag{3.14}$$

$$x_{j0}^{kt} \in \{0,1\} \quad , \forall (i,j) \in V, \forall k \in K, \forall t \in T \tag{3.15}$$

$$y_{i}^{kt} \in \{1,0\} \quad , \forall i \in V, \forall k \in K, \forall t \in T \tag{3.16}$$

The objective function (3.1) minimizes the total sum of holding costs and traversing costs. Constraints (3.2) and (3.3) define the inventory levels at the supplier and each customer at the end of each period. Constraint (3.4) imposes a maximum level of inventory storage at each customer. Constraint (3.5) defines that each delivery cannot exceed customer available capacity while (3.6) guarantees that a vehicle can only deliver products to customer it visits. Constraints (3.7) and (3.8) guarantee that if a node is visited then it must have an input and output arc. Constraints (3.9) and (3.10) guarantee the feasibility of the schedule where constraint (3.9) deal with subtour elimination while constraint (3.10) defines the the earliest start start for a node in the case the selected vehicle traverses through it, imposing service time constraints. Constraints (3.11)-(3.16) ensure non-negativity and integrality.

Chapter 4

Proposed Approach

4.1 Algorithm Design

In order to create a flexible and easy to manipulate solution, the developed algorithm represents the solution's route by the means of an integer vector which always begins and ends at the supplier ($i = 0$), visiting customers along the prescribed route. With this representation one can easily develop simple routines for recurrent tasks as calculating a route total distance or verifying if a node is already inserted in a route. In order to simplify calculations the total cost of each iterated solution was separated in three different groups:

- Initial customer cost (ICC) $\rightarrow$ which is calculated taking into account the initial inventory at each customer and the amount of time it takes to consume such quantity.

$$ICC = max(0; \sum_{t}\sum_{p}\sum_{i} I^{0}_{pi} - \sum_{x=1}^{t} d^{x}_{i,p} * h_{i,p}) \quad ,\forall\, i \in V, \forall k \in K, \forall t \in T \tag{4.1}$$

- Supplier inventory cost (SIC) $\rightarrow$ which is calculated taking into account the initial inventory at the supplier as well as the production which results in an increase of stock and subtracts the total inventory delivered to every customer per period for each product.

$$SIC = max(0; \sum_{t}\sum_{p} I^{0}_{p0} - \sum_{x=1}^{t} d^{x}_{i,p} + production_{t} * h_{i,p}) \quad ,\forall\, i \in V, \forall k \in K, \forall t \in T \tag{4.2}$$

- Customer Cost (CC) $\rightarrow$ which is composed by the final solution routing cost and by the inventory cost inherent to each customer visited, depending on the quantity delivered, the period when the visit takes place and the period when the demand is actually required.

$$CC = RouteCost + delivery_{t,tt} \quad ,\forall\, i \in V, \forall k \in K, \forall t \in T, \forall tt \in T \tag{4.3}$$

$$delivery_{t,tt} = \sum_{i}\sum_{p}\sum_{x=t}^{tt} I^{x}_{pi} * h_{ip} * (tt - x) \quad ,\forall\, i \in V, \forall k \in K, \forall t \in T, \forall tt \in T \tag{4.4}$$

The vehicle capacity constraint is addressed by allowing some unfeasible solutions which are penalized in the final cost but may help broaden the search-space during a LS Phase which may then lead to good feasible solutions.

Taking into account the deterministic nature of the problem it is possible to assume that an instance of predetermined vehicles will not need more that the established number.Therefore, the algorithm will at most initialize a route per vehicle and will recurrently calculate the trade-off between adding a new route or fill a vehicle's capacity.

4.2 GRASP

In order to accomplish the search for good quality solutions to the problem at hand, a GRASP framework is proposed. This section will describe the design and implementation of the selected method.

The election of the GRASP methodology is justified by it's easiness of implementation as it requires few parameters. The basic GRASP framework algorithm is depicted in Algorithm 1. As stated before a GRASP cycle consists in constructing an initial Greedy Solution and then applying to it a LS routine.

Due to its randomized greedy methodology the algorithm will start building the solution from scratch iteratively adding a new component to the partial solution in development, until a feasible solution has been constructed.

Algorithm 1 GRASP

```
ReadInputInstance()
for cicle = 0 to MAXITERATIONS do
    ConstructGreedySolution()
    LocalSearch()
    UpdateSolution()
end for
return BestSolution
```

For each cycle the algorithm will compare the final solution cost with the current best, if the first one is better than the latter the algorithm will save the current best and proceed to the next cycle until the stopping conditions are met.

4.2.1 Constructive Phase

For the construction phase two different approaches were considered. For both possibilities the algorithm starts by computing the updated demand of each customer taking into account the initial inventory found at customers in the beginning of the horizon and the demand until the end of the planning horizon as seen in Algorithm 2.

Algorithm 2 Update Demand

```
if InventoryLevel[t-1] < r[i][l][t] then
    r[i][l][t] ← r[i][l][t] - InventoryLevel[t]
    InventoryLevel[t] ← 0;
else
    InventoryLevel[t] ← InventoryLevel[t] - r[i][l][t];
    r[i][l][t] ← 0
end if
```

4.2.1.1 Construction 1: customer per period

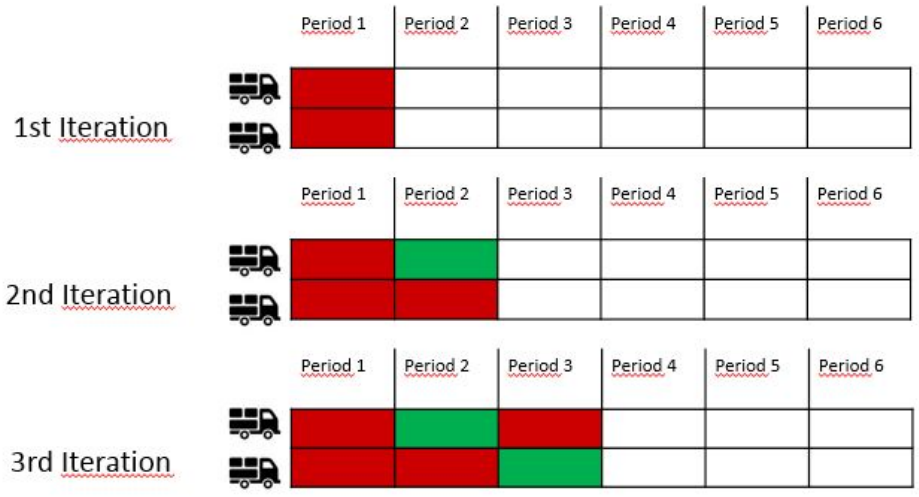
The first approach presented in Algorithm 3 uses the updated demand to compile a list of customers which need delivery for each period.

Algorithm 3 Createcandidates1

```
for every t do
    for every customers do
        if demand > previousperiod finalinventory then
            Candidates[t] ← customer
        end if
    end for
    PeriodFinalInventory = previousperiod finalinventory − demand
end for
```

Figure 4.1 portrays an example of construction following the described strategy. The algorithm selects a periods and proceeds to allocate every customer that needs to be visited.

Figure 4.1: Customers per period



4.2.1.2 Construction 2: periods per customer

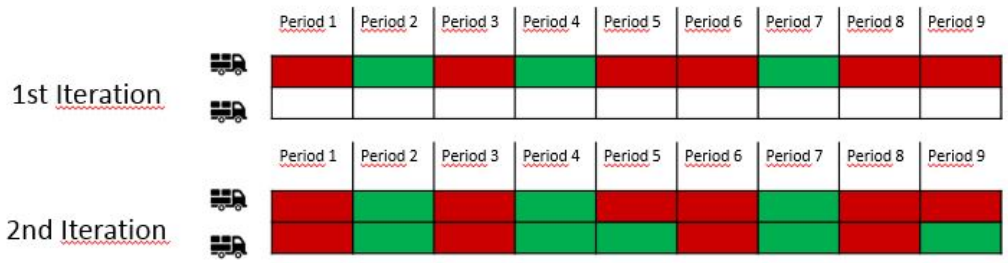
The second approach presented in Algorithm 4 uses the updated demand to compile the list of periods in which the customers need to have their demand fulfilled.

Algorithm 4 Createcandidates2

```
for every t do
    for every customers do
        if demand > previousperiodfinalinventory then
            Candidates[i] ← t
        end if
    end for
    PeriodFinalInventory = previousperiodfinalinventory − demand
end for
```

Figure 4.2 represents an example of construction following the second strategy. The algorithm selects a customer and proceeds to allocate said customer to every period in which it needs visiting.

Figure 4.2: Periods per customer



4.2.1.3 Candidate List

After the preliminary list is compiled the algorithm will seek to satisfy every customers demands and will proceed to create a CandidateList (CL) as presented in Algorithms 5 and 6.

Each candidate to be compiled in CL is defined by the following tupple:

- customer id
- (t) base period: period in which the visit will occur
- (tt) target period: period until which demand is being satisfied
- the calculated saving cost of insertion for such customer.
- flag which signalizes the need for adding a new node to route[tt]
- position in which the node will be placed

The savings parameter considered, takes into account the cost of inserting each each customer in $route_{tt}$ taking into consideration the extra holding costs by the early deliver subtracted by the

number of not necessary routes, originated by visiting a customer earlier than the demand period. In other words if a customer is visited in period 2 (the first period with updated demand) and supplies the demand up until period 7, there will be a saving of 5 visits minus the holding costs appended with each batch of inventory meant for periods from 2 to 7. In the case a customer is visited in period 1 covering the demand up until period 7 and it's updated demand is only bigger than zero at period 3, the visit will only save 4 routes and the holding costs will be considered as in the first case.

Let $q'_{p,i,k,t,tt}$ be the quantity of product $p \in Pi$ delivered to customer i by vehicle k in period t satisfying the demand up until tt calculated as seen in 4.5.

$$q'_{p,i,t,tt} = \sum_{x=t}^{tt} q_{pi}^{kx} \quad ,\forall i \in V, \forall k \in K, \forall t \in T, \forall tt \in T \tag{4.5}$$

The β parameter in the savings equation (4.6) varies depending on the conditions each candidate is created:

$$saving_{i,t,tt} = -\beta_{i,t,t''} - ((t'' - t) * c_{i,j} \quad ,\forall i \in V, \forall k \in K, \forall t \in T, \forall tt \in T \tag{4.6}$$

- If the created candidate customer is meant to satisfy the demand of the current period, it will only incur in routing cost by adding a new node to the route; the inventory cost are not considered as the inventory is consumed in the same period as it is delivered

$$\beta_{i,t,tt} = c_{i,j} \quad ,\forall i \in V, \forall j \in V \tag{4.7}$$

- If the created candidate customer is meant to be visited in period t satisfying the demand until period tt and the customer is already visited in period t, then the candidate will only incur in holding costs until period t, as the routing costs have already been accounted for.

$$\beta_{i,t,tt} = \sum_{t''=t}^{tt} (q'_{p,i,t,tt} - \sum_{x=t}^{t''} d_{i,p}^{x}) * h_{i,p}) \quad ,\forall i \in V, \forall j \in V, \forall t \in T, \forall tt \in T \tag{4.8}$$

- If the created candidate customer is meant to be visited in period t satisfying the demand until period tt and is not yet visited in route[tt], then the candidate must take into account both routing cost and inventory holding costs:

$$\beta_{i,t,tt} = \sum_{t''=t}^{tt} (q'_{p,i,t,t''} - \sum_{x=t}^{t''} d_{i,p}^{x}) * h_{i,p}) + c_{i,j} \quad ,\forall i \in V, \forall j \in V, \forall t \in T, \forall tt \in T \tag{4.9}$$

- If the created candidate is meant to satisfy demand from period t until tt and there is no initiated route in period t, the candidate will incur in both routing and inventory holding

costs:

$$\beta_{i,t,tt} = \sum_{t''=t}^{tt} (q'_{p,i,t,t''} - \sum_{x=t}^{t''} d^x_{i,p}) * h_{i,p}) + c_{0,i} + c_{i,0} \quad ,\forall i \in V, \forall j \in V, \forall t \in T, \forall tt \in T \quad (4.10)$$

Algorithms 5 and 6 describe the creation of the CL following strategies 1 and 2 respectively:

Algorithm 5 CreateCL1

for $all\, n \in Candidates_t$ **do**
  for tt:=0 to tt < t **do**
    if candidate already in route[tt] **then**
      $subcost \leftarrow (q'_{p,i,t,tt} - \sum_x^t d^t_{i,p}) * h_{i,p})$
      $CL[iterator] \leftarrow [n,t,tt,subcost,flag,position]$
    else
      if route not empty **then**
        for every position position in route[tt]
        $subcost \leftarrow ((q'_{p,i,t,tt} - \sum_x^t d^t_{i,p}) * h_{i,p}) - c_{i,j} + c_{i,n} + c_{n,j}$
        $CL[iterator] \leftarrow [n,t,tt,subcost,flag,position]$
      else
        $subcost \leftarrow (c_{i,n})$
        $CL[iterator] \leftarrow [n,t,tt,subcost,flag,position]$
      end if
    end if
  end for
  for $tt == t$ **do**
    $subcost \leftarrow (c_{i,n})$
    $CL[iterator] \leftarrow [n,t,tt,subcost,flag,position]$
  end for
end for

Algorithm 6 CreateCL2

for $all\, t \in Candidates_n$ **do**
  for tt:=0 to tt < t **do**
    if n already in route[tt] **then**
      $subcost \leftarrow ((q'_{p,i,t,tt} - \sum_x^t d_{i,p}^t) * h_{i,p})$
      $CL[iterator] \leftarrow [n,t,tt,subcost,flag,position]$
    else
      if route not empty **then**
        for every position position in route[tt]
        $subcost \leftarrow (q'_{p,i,t,tt} - \sum_x^t d_{i,p}^t) * h_{i,p}) - c_{i,j} + c_{i,n} + c_{n,j}$
        $CL[iterator] \leftarrow [n,t,tt,subcost,flag,position]$
      else
        $subcost \leftarrow (c_{i,n})$
        $CL[iterator] \leftarrow [n,t,tt,subcost,flag,position]$
      end if
    end if
  end for
  for $tt == t$ **do**
    $subcost \leftarrow (c_{i,n})$
    $CL[iterator] \leftarrow [n,t,tt,subcost,flag,position]$
  end for
end for

The creation of CL will naturally be affected by the approach used to create the preliminary list. Using the first approach the sequence of selected t must be taken into account whether in the second approach it is the sequence of customers by which the candidates are created that need a special consideration as the sequence of this parameters will influence the variability of the CL which is crucial to guarantee good results.

Once CL is compiled the algorithm scans the CL in search for the best candidates, which will be kept in the Restricted Candidate List (RCL) in accordance with a previously defined α which controls the trade-off between greedy and random; if α value is 0 the the construction will be completely greedy as the RCL will only accept the candidate with the cheapest subcost presenting a variance of zero and a mean value equal to the only accepted value; if α value is 1 then the construction is completely random as the RCL will accept all values between the worst and best subcosts allowing for more solutions than with a higher value for α, presenting the biggest variance possible and a mean higher than the completely greedy solution. A Limit Cost is calculated as in Equation 4.11 for each iteration taking into account the values for maximum and minimum candidate costs.

$$LimCost = min + \alpha \cdot (max - min) \tag{4.11}$$

The need for randomness of the algorithm comes from the fact that always selecting the cheapest delivery does not represent the cheapest overall solution. Applying randomness at this stage will allow for a more diverse scan of the search-space possibly fleeing from local optima. Algorithm 7

describes the construction of the RCL.

Algorithm 7 Update RCL

for all candidates in CL **do**
  Identify max subcost
  Identify min subcost
end for
$limcost = min + \alpha \cdot (max - min)$
for all candidates with subcost < limcost **do**
  $RCL \leftarrow CL$
end for

Finally the algorithm will randomly select one element at a time from the RCL and will insert it in the solution reevaluating the costs of every element, This cycle will be repeated until all customers in the preliminary candidates list are satisfied and the solution complete.

The construction phase may then be modeled using Algorithm 8.

Algorithm 8 ConstructGreedySolution

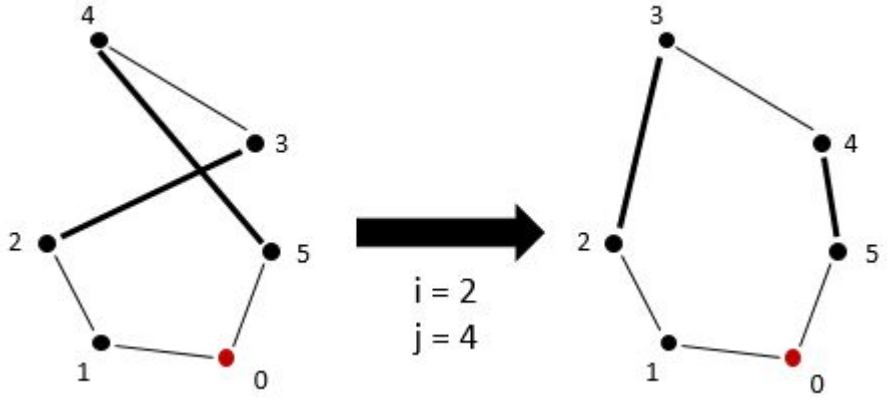
$PreliminaryCandidates \leftarrow Createcandidates()$;
while PreliminaryList not null **do**
  $Selectrandomxfrom PreliminaryList$
  $y = CreateCL(x)$
  $UpdateRCL()$
  select random element from RCL
  insert in route[tt]
  Update inventory levels of supplier and customers
  erase candidate from $PreliminaryCandidates$
end while

4.2.2 Local Search

To ensure the LS Phase and in order to complete the GRASP cycle the proposed approach uses a variation of the k-opt methodology as depicted in Algorithm 9. Its has the purpose of improving the initial constructed solution by rearranging k arcs from the route aiming to reduce the total routing cost of every period. This subsection will present the 2-opt and 3-opt algorithms so as to introduce the implemented k-opt variant.

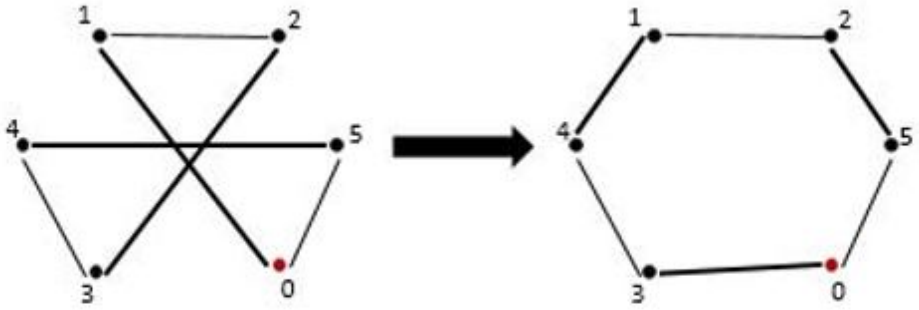
The 2-opt procedure iteratively selects 2 arcs in the initial route and swaps the nodes in search of a better solution. Let the arcs be defined as $(i, i+1)$ and $(j, j+1)$, the 2-opt procedure would create a neighborhood by performing swaps which would result in the arcs (i, j) and $(i+1, j+1)$ or $(i, j+1)$ and $(j, i+1)$. If the new route results in a cheapest route the algorithm keeps the solution and may decide to perform the improvement immediately (greedy) taking less computational time, or it may wait for the completion of the cycle and performs the best improvement found.

Figure 4.3: 2-opt result



The same happens for the 3-opt where instead the procedure considers 3 arcs, allowing the detection of crossovers in the route that cannot be identified by the 2-opt.

Figure 4.4: 3-opt result



Algorithm 9 Local Search Phase

```
BestDistance ← CalculateDistance(Solution);
for arc1 do
    for arc2 do
        newSolution = Kopt(solution, route, arc1, arc2);
    end for
end for
NewDistance ← CalculateDistance(newSolution);
if NewDistance < BestDistance then
    Solution = NewSolution;
    BestDistance = NewDistance;
end if
```

The introduced k-opt procedure receives two parameters $arc1$ and $arc2$ which set boundaries for the methodology. The algorithm will copy the original route from the beginning until the $arc1$th position and will place in reverse order the nodes between $arc1$ and $arc2$; from the $arc2$th position until the end of the route, the algorithm will copy the route in the original order, as seen in Algorithm 10. The expected results are routes without any crossovers and with smaller total distances which can be translated as an intensification of the search.

Algorithm 10 K-opt Algorithm

```
for route position beginning until route position arc1 do
    insert in order: new route ← route
end for
for route position arc1+1 until route position arc2 do
    insert in reverse order route in new route
end for
for route position arc2+1 until route end do
    insert in order route in new route
end for
return new route
```

4.3 Reactive GRASP

As stated before, a completely greedy GRASP ($\alpha = 0$) is not the best solution neither is the random variant ($\alpha = 1$), in fact there is no fixed α which can hold the best overall solution for every instance. Trying to escape this, the implemented GRASP was made Reactive so as to iteratively update the best α for each instance as proposed in Prais and Ribeiro (2000). This strategy usually presents improvements in comparison with the basic GRASP framework at the cost of bigger computational times Resende and Ribeiro (2008). The next section will present the comparative study of both approaches.

Following the routine presented in (Boudia et al., 2007) let ζ be the number of possible values for α, $\omega = \alpha_1, \alpha_2, ..., \alpha_\zeta$ and α_s the sth value of ω; all values of ω are initialized with the same probability $P_\omega = (1/\zeta)$ (uniform distribution) and in each iteration the algorithm will randomly select a value taking into account the probability given to each value of α, which are updated at the end of each iteration. For each iteration the algorithm will keep the number of times an α is chosen in $count_s$ and will also keep the overall cost that resulted from the LS in $score_s$. In order to update the probabilities of each α for every sequence of $ncicles$ the algorithm updates the value for Z_min which keeps the the best solution values up-to-date and will compute the following equations 4.12, 4.13 and 5.1:

$$avg_s = score_s / count_s \tag{4.12}$$

$$quant_s = (Z_{min} / avg_s)^\lambda \tag{4.13}$$

$$\phi = \sum quant_s \tag{4.14}$$

it is then possible to calculate the new probabilities as $P_s = quant_s/\phi$. Algorithm 11 presents the implemented pseudo-code for the RGRASP

Algorithm 11 Reactive GRASP

```
Z_min = MAXVALUE;
for s = 1 → ζ do
    P_s = 1/ζ
    Count_s = 0
    Avg_s = 0
    Score_s = 0
end for
ReadInputInstance()
for cicle = 0 → MAXITERATIONS do
    Randomly select s ∈ [1,...,ζ] taking into acocunt P_s
    Count_s = Count_s + 1;
    Greedy ← ConstructGreedySolution(α_s)
    Local ← LocalSearch(Greedy)
    BestSolution ← UpdateSolution(Local)
    if Z_local < Z_min then
        Z_min ← Z_local;
    end if
    if cicle mod ncicles == 0 then
        avg_s = score_s/count_s;
        quant_s = (Z_min/avg_s)^λ
        φ = Σ quant_s
        for s = 1 → ζ do
            P_s ← (quant_s/φ)
        end for
    end if
    Score_s = Score_s + Z_local;
end for
return BestSolution
```

After a few sets of *ncicles* the α which presented the smaller overall costs will present a bigger probability of being chosen for the next cycle.

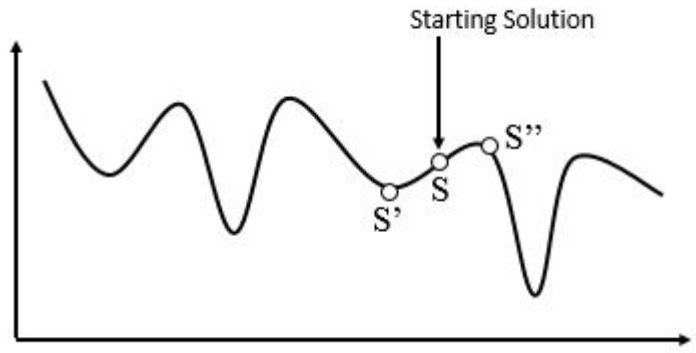
4.4 Simulated Annealing

The simulated annealing technique proposed by Kirkpatrick (1984) simulates the annealing procedure which consists in a heat based method in the metallurgy industry that is used to alter physical properties by heating a material up to high temperature followed by a controlled cooldown which enables some properties as ductility and reduces hardness.

The SA tries to mimic this natural event, considering that the LS phase will have an associated temperature which will set the standard for accepting the solutions reached with the LS. The temperature is initialize with a high number which reflects in allowing worst solutions than the ones set to be improved. As the temperature cools down by a factor of γ the acceptance standard will become more and more demanding permitting only improved solutions in the final iterations.

Allowing for worst solutions, as previously stated, helps escaping local optima diversifying the search while the k-opt procedure deals with the intensification. In Figure 4.5 let S be the initially constructed solution; without de SA the algorithm would search for improving neighbors which would eventually lead to the local optimum S'; with SA the algorithm may accept a solution such as S" which may lead to a better final solution.

Figure 4.5: Neighbors



In the next section will be presented a comparative study between algorithms with and without SA. Modifying the Search Phase Algorithm to implement the SA results in Algorithm 12.

Algorithm 12 Simulated Annealing Local Search

```
Initialize: Temperature; Tmin; γ;
BestDistance ← CalculateDistance(Solution);
while Temperatue < Tmin do
    for arc1 = 0; arc1 < RouteSize - 1; arc1++ do
        for arc2 = 0; arc2 < RouteSize - arc1 - 1; arc2++ do
            newSolution = opt(solution, route, arc1, arc2);
        end for
    end for
    NewDistance ← CalculateDistance(newSolution);
    ap = acceptanceprobability(BestDistance, NesDistance, Temperature)
    if ap < random() then
        Solution = NewSolution;
        BestDistance = NewDistance;
    end if
    Temperature = Temperature * γ;
end while
```

Chapter 5

Computational Results

In this chapter the computational results obtained with the designed algorithm will be scrutinized as to obtain as much information as possible. A comparative study was performed between the GRASP based approaches and the B&C algorithm developed by Coelho and Laporte (2013a).

The algorithm was coded in C++ and all experiments were performed in a Toshiba Satellite L50-A running Linux Mint 14.04 with an Intel Core i5-4200M running at 2.4Ghz with 64bits and 6Gb of RAM. The benchmark used was developed by Coelho and Laporte (2013a) allowing for a comparison with the Upper-bound of the MMIRP solved by a specialized B&C algorithm.

The B&C algorithm was able to solve 675 instances, 5 instances for each combination of customers, periods, products and vehicles, considering:

- 10, 20, 30, 40 and 50 customers
- 3, 5 and 7 periods
- 1, 3 and 5 products
- 1,3 and 5 vehicles

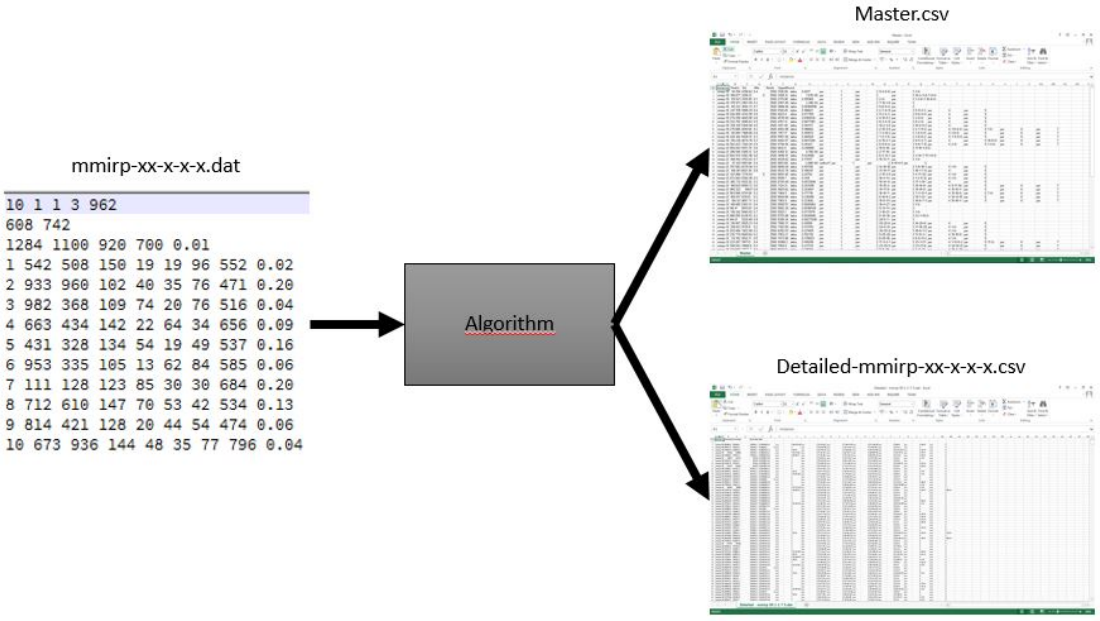
The presented work was developed throughout a six month period and due to the limited time available the developed algorithm only solves the MMIRP for one product and one vehicle considering 3, 5 and 7 periods. As to present a reliable comparison, each tested instance as a measured GAP. The GAP represents the distance to the best known solution; let S be the solution found with the implemented algorithm and BKS be the best known solution up to comparison, the GAP is then calculated as in Equation 5.1

$$GAP = (S - BKS)/BKS \tag{5.1}$$

The created algorithms receives the benchmark instances as an input, and retrieves a Master excel file which presents the best solution found for each instance and the GAP for the compared results as well as the computational time needed to run the pre-determined number of cycle. The output also contains the routes for each period of the best solution found (as portrayed in Figure

5.1). A detailed excel file for each instance, called upon the input file, which presents the computational time needed to ran each cycle, the solution found before and after LS, the GAP for the compared results as well as the routes for each period per cycle.

Figure 5.1: Input / Output



5.1 Parameter Settings

During preliminary testing the developed GRASP algrithm was calibrated with the following parameters:

- Number of GRASP cycles: n * 2;
- Number of Local Search cycle (without SA): 100;
- Initial temperature for SA: 1;
- Minimum temperature for SA: 0.0001;
- SA cooldown factor: 0.9;

5.2 Computational Results

This section presents the comparative study for the computational times and the GAP between the developed algorithms. Four algorithms are up to the test:

1. Standard basic GRASP + K-opt

2. Reactive GRASP + K-opt
3. GRASP with SA + K-opt
4. RGRASP with SA + K-opt

In each subsection a Table is displayed presenting the mentioned indicators values in comparison with the Basic GRASP. Each Table exhibits the average values for each group of 5 instances that share the number of periods and customers; also displaying the final Δ for GAP and Computational. Let the GRASP algorithm be (1) and the algorithm up for comparison (2). Δ_{GAP} and Δ_{CPU} are calculated as in Algorithm 5.2 and Algorithm 5.3 respectively.

$$\Delta_{GAP} = GAP_{(2)} - GAP_{(1)} \tag{5.2}$$

$$\Delta_{CPU} = (CPU_{(2)} - CPU_{(1)})/CPU_{(1)} \tag{5.3}$$

5.2.1 Comparison between GRASP and RGRASP

When comparing the basic GRASP with the Reactive variant (Table 5.1) is possible to verify that the latter tends to present on average better overall solutions, displaying an average improvement of 4% of GAP.

Table 5.1: Comparison: GRASP - RGRASP

		GRASP		**RGRASP**		Δ_{GAP}	Δ_{CPU}
		GAP(%)	**CPU(s)**	**GAP(%)**	**CPU(s)**		
	n=10	19,53%	176	11,23%	245	-8,29%	38,65%
	n=20	20,49%	182	18,60%	258	-1,89%	41,72%
t=3	**n=30**	10,23%	470	4,25%	685	-5,98%	45,77%
	n=40	5,10%	184	3,68%	276	-1,42%	50,15%
	n=50	7,46%	374	1,46%	592	-6,00%	58,27%
	n=10	36,37%	367	14,05%	608	-22,33%	65,83%
	n=20	18,00%	296	16,58%	411	-1,42%	38,91%
t=5	**n=30**	27,61%	457	24,75%	173	-2,86%	-62,07%
	n=40	26,92%	268	20,77%	406	-6,16%	51,61%
	n=50	20,87%	646	20,04%	924	-0,83%	43,16%
	n=10	20,60%	539	16,39%	708	-4,21%	31,27%
	n=20	25,22%	319	21,75%	433	-3,47%	36,00%
t=7	**n=30**	23,21%	132	25,60%	178	2,39%	35,24%
	n=40	24,23%	279	25,59%	393	1,36%	40,77%
	n=50	24,63%	684	24,58%	933	-0,05%	36,32%
Averages		21%	358,16	17%	481,58	-4%	36,77%

In Figure 5.2 is possible to observe the GAP evolution throughout the increased number of customers, for each considered period.

Figure 5.2: Basic GRASP vs RGRASP: GAP(%)

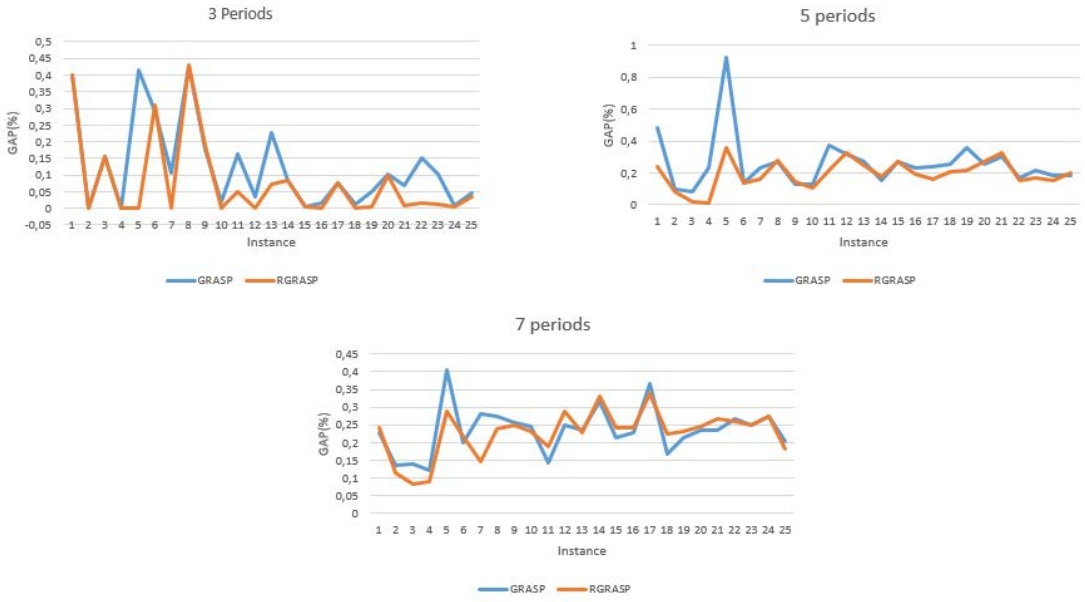


Figure 5.3 allows a graphical comparison of the difference in computational times, which is on average 37% higher for the RGRASP (Table 5.1)

Figure 5.3: Basic GRASP vs RGRASP: CPU(s)

5.2.2 Comparison between GRASP and GRASPSA

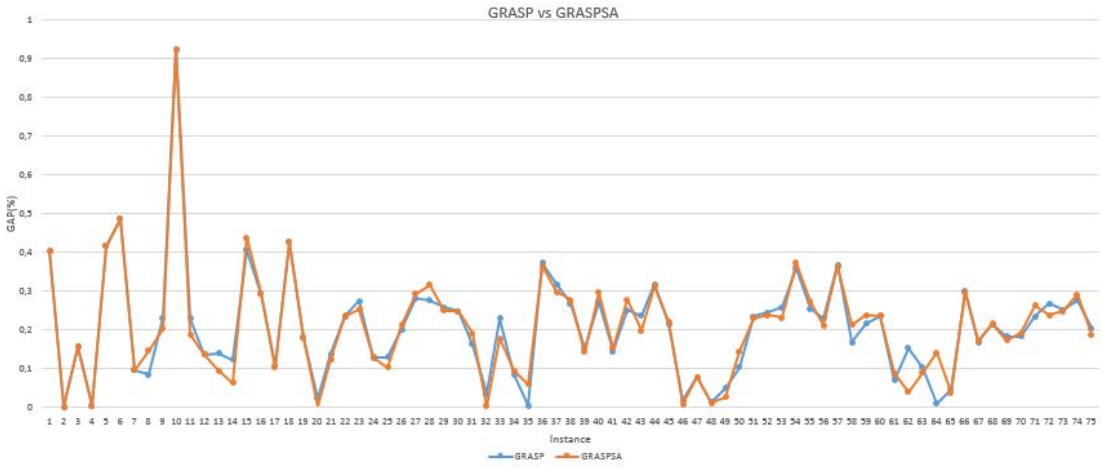
When comparing the final GAP values for both the basic GRASP algorithm and the GRASP with the Simulated Annealing in Figure 5.4 it is possible to verify that both approaches present similar results. In fact when analyzing Table 5.2 is possible to establish that the GRASPSA does not

improve the average results. The basic GRASP shows better solutions for 33 instances while the second approach shows a slight improvement of 32 instances which translates as a consequence of allowing worst solution in LS.

Table 5.2: Comparison: GRASP - GRASPSA

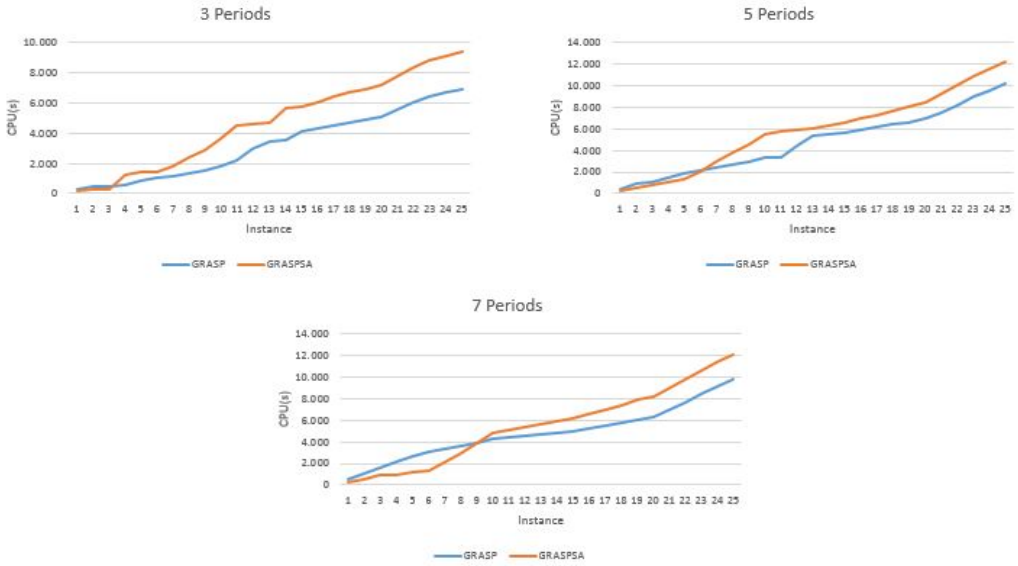
		GRASP		**GRASPSA**		Δ_{GAP}	Δ_{CPU}
		GAP(%)	**CPU(s)**	**GAP(%)**	**CPU(s)**		
t=3	**n=10**	19,53%	176	19,53%	280,38	0,00%	58,99%
	n=20	20,49%	182	20,02%	440,65	-0,48%	141,71%
	n=30	10,23%	470	10,43%	437,54	0,20%	-6,91%
	n=40	5,10%	184	5,20%	277,60	0,10%	51,10%
	n=50	7,46%	374	7,79%	444,78	0,33%	18,94%
t=5	**n=10**	36,37%	367	37,05%	255,48	0,68%	-30,37%
	n=20	18,00%	296	16,81%	850,16	-1,19%	187,13%
	n=30	27,61%	457	27,50%	203,56	-0,11%	-55,46%
	n=40	26,92%	268	26,83%	393,10	-0,09%	46,87%
	n=50	20,87%	646	20,90%	754,57	0,03%	16,85%
t=7	**n=10**	20,60%	539	18,31%	239,46	-2,29%	-55,61%
	n=20	25,22%	319	26,31%	733,62	1,09%	130,28%
	n=30	23,21%	132	23,21%	262,19	0,00%	99,26%
	n=40	24,23%	279	25,17%	416,86	0,94%	49,49%
	n=50	24,63%	684	24,46%	773,47	-0,17%	13,06%
Averages		21%	358,16	21%	450,89	0%	44%

Figure 5.4: Basic GRASP vs GRASPSA: GAP(%)



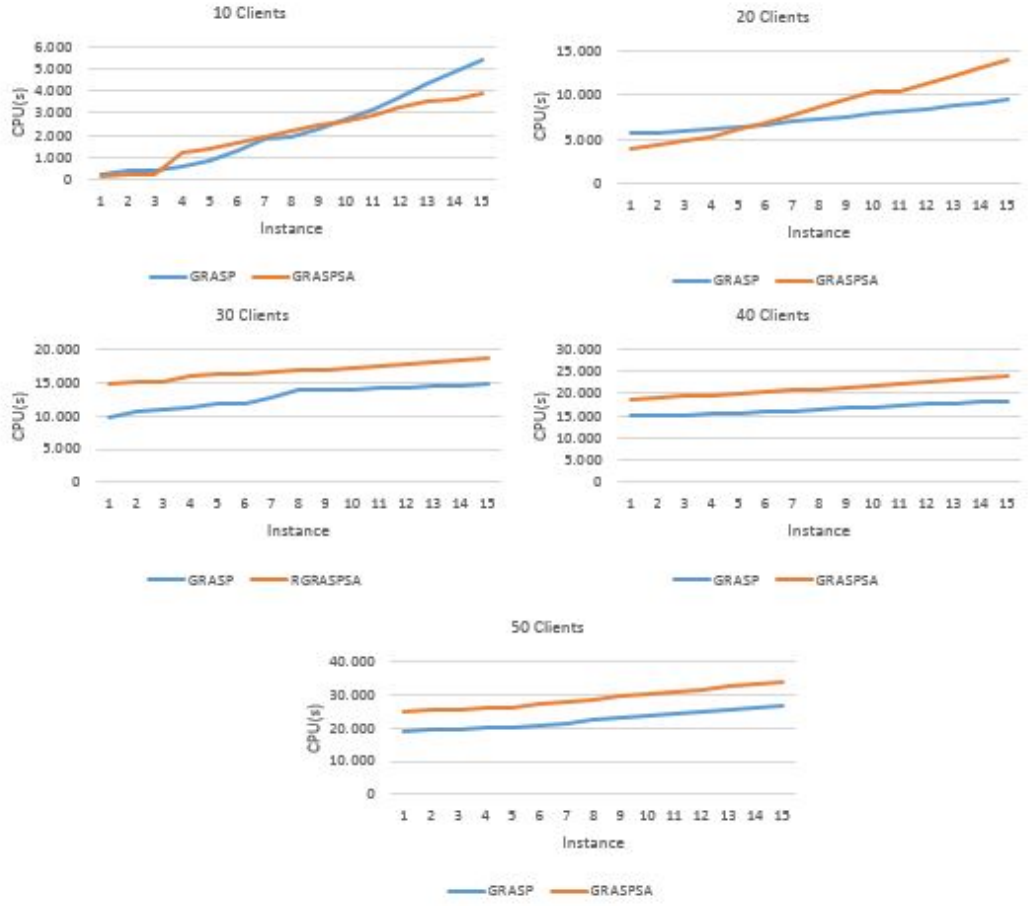
When comparing the computational time it is possible to verify in Figure 5.5 that for a fixed period the SA approach tends to take longer computational times as the number of customers increases, though presenting better computational times for instances with less customers.

Figure 5.5: Basic GRASP vs GRASPSA: Computational Time (fixed number of period) (s)



When fixing the number of customers and increasing the number of periods it is evident as seen in Figure 5.6 that the SA approach tends to present a slightly higher computational times for a higher number of customers. Fitting with the assumption that computational time increases with the increase in the number of customers.

Figure 5.6: Basic GRASP vs GRASPSA: Computational Time (fixed number of customers) (s)



5.2.3 RGRASPSA

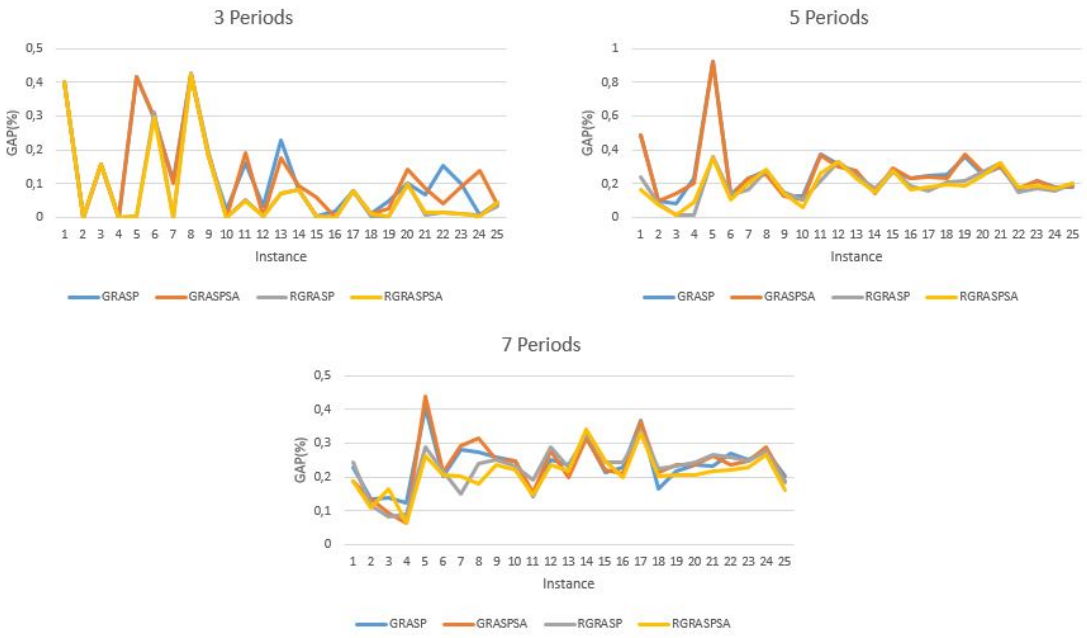
The fourth algorithm, RGRASPSA, which combined the Reactive variation of the GRASP with the SA in the LS phase is compared with GRASP in Table 5.3. It is possible to verify that the RGRASPSA presents, on average a smaller GAP at the cost of an increase of 41% in computational time.

Table 5.3: Comparison: GRASP - RGRASPSA

		GRASP		**RGRASPSA**		Δ_{GAP}	Δ_{CPU}
		GAP(%)	**CPU(s)**	**GAP(%)**	**CPU(s)**		
	n=10	19,53%	176	11,19%	306,91	-8,34%	74,04%
	n=20	20,49%	182	17,95%	425,77	-2,54%	133,55%
t=3	**n=30**	10,23%	470	4,20%	304,38	-6,03%	-35,24%
	n=40	5,10%	184	3,80%	285,70	-1,31%	55,50%
	n=50	7,46%	374	1,76%	492,89	-5,70%	31,80%
	n=10	36,37%	367	14,02%	267,63	-22,36%	-27,06%
	n=20	18,00%	296	15,73%	736,66	-2,27%	148,79%
t=5	**n=30**	27,61%	457	25,07%	214,86	-2,55%	-52,99%
	n=40	26,92%	268	19,35%	407,71	-7,57%	52,33%
	n=50	20,87%	646	21,37%	765,68	0,50%	18,57%
	n=10	20,60%	539	15,72%	304,83	-4,88%	-43,49%
	n=20	25,22%	319	20,93%	597,71	-4,30%	87,62%
t=7	**n=30**	23,21%	132	23,77%	275,90	0,56%	109,67%
	n=40	24,23%	279	22,87%	416,70	-1,36%	49,43%
	n=50	24,63%	684	21,88%	783,72	-2,75%	14,55%
Averages		21%	358,16	16%	439,14	-5%	41%

When comparing the four algorithms in Figure 5.7 is verifiable that the RGRASPSA presents the best behavior of the four algorithm as it is usually the one with the lower GAP.

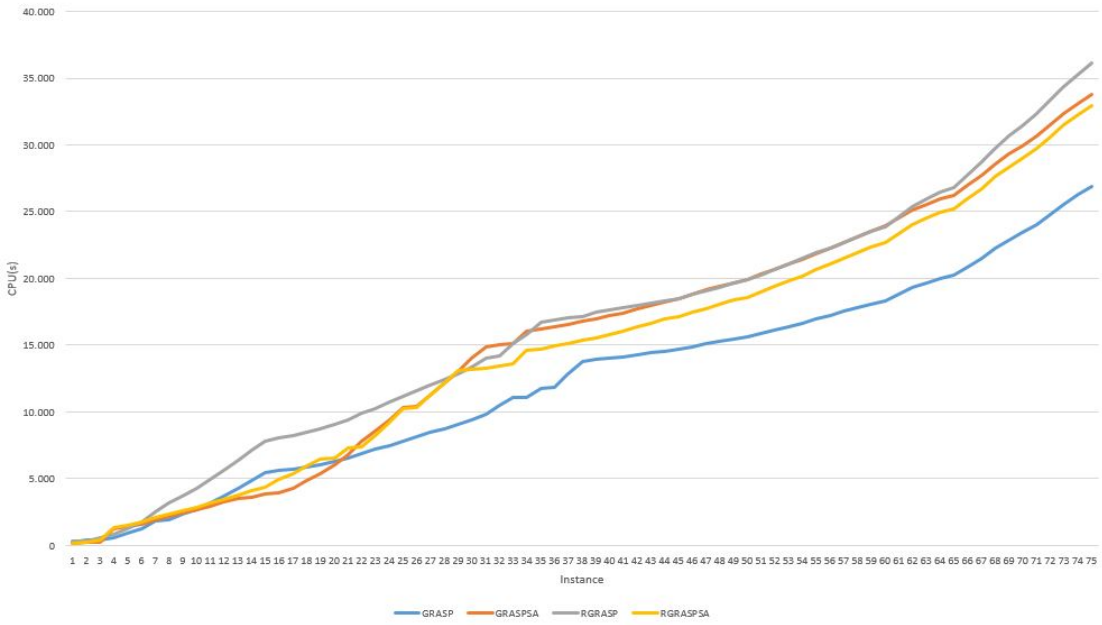
Figure 5.7: 4-Algorithm GAP (%)



In Figure 5.8 is possible to observe the difference between computational time of the four

methods. It is possible to verify that for smaller instances of 10 customers the GRASP SA methodology presents the better computational time, worsening as the instance's size grows. The basic GRASP wields the overall best computational time, which is explained by an increase of the LS cycles for the SA and Reactive variants. The RGRASPSA presents better overall results when compared to the RGRASP and GRASPSA which may be explained by better average result for the initial solution which translates in less cycles for the LS as the SA breaks the routine when no more improvements are made.

Figure 5.8: 4-Algorithm Computational Time(s)



5.2.4 Comparison with B&C algorithm

When comparing the GAP for best know solutions, out of the four implemented algorithms, with the B&C as seen in Figure 5.9 it becomes clear that the B&C wields a better solution for every tested instance.

Figure 5.9: B&C vs Best Known Value

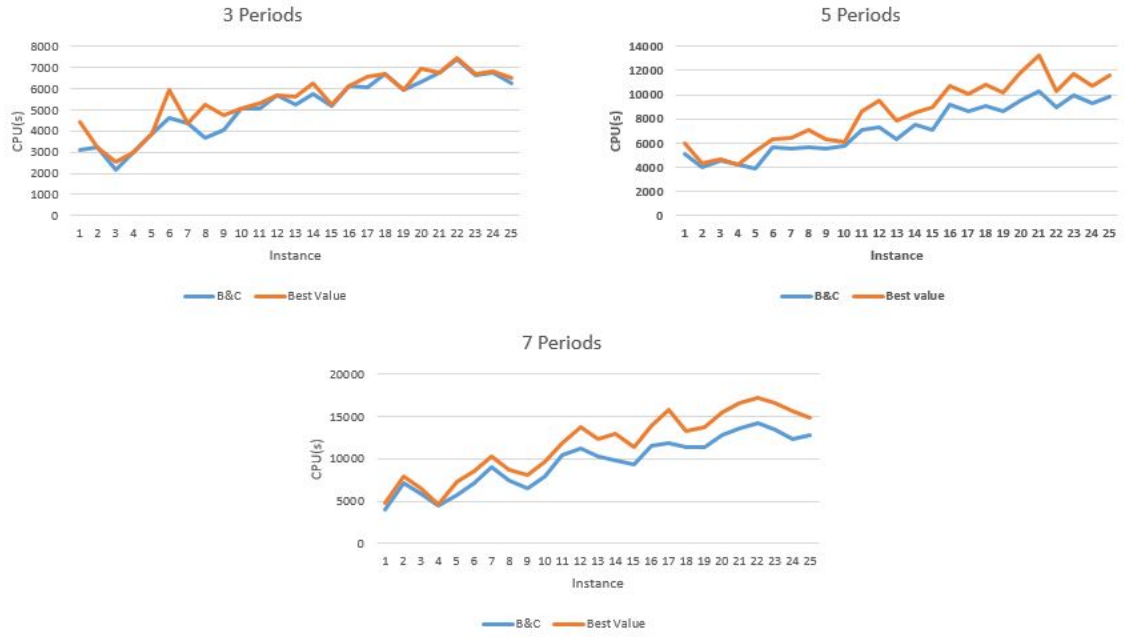


Table 5.4 presents the average solution for each group of five instances with the same number of customers, products and periods. Let the instances be defined by the following tupple "mmirp-number of customers - number of products - number of vehicles - number of periods".

Table 5.4: Algorithms Comparison

		GRASP		RGRASP		GRASPSA		RGRASPSA		B&C
		Sol.	**GAP(%)**	**Sol.**	**GAP(%)**	**Sol.**	**GAP(%)**	**Sol.**	**GAP(%)**	**Sol.**
t=3	**n=10**	3727,34	19,53%	3405,09	11,23%	3727,336	19,53%	306,909	**11,19%**	3083,726
	n=20	5196,47	20,49%	5110,13	18,60%	5173,204	20,02%	425,771	**17,95%**	4353,142
	n=30	5933,69	10,23%	5620,95	4,25%	5940,766	10,43%	304,3804	**4,20%**	5391,252
	n=40	6539,27	5,10%	6452,75	**3,68%**	6540,35	5,20%	285,6976	3,80%	6224,512
	n=50	7273,16	7,46%	6854,92	**1,46%**	7283,664	7,79%	492,8878	1,76%	6757,696
t=5	**n=10**	5937,44	36,37%	4981,04	14,05%	5970,772	37,05%	267,6274	**14,02%**	4368,824
	n=20	6671,46	18,00%	6592,08	16,58%	6603,332	16,81%	736,6648	**15,73%**	5655,94
	n=30	9005,36	27,61%	8808,33	**24,75%**	8994,28	27,50%	214,8606	25,07%	7062,532
	n=40	11427,18	26,92%	10888,88	20,77%	11419,5	26,83%	407,7088	**19,35%**	9010,172
	n=50	11674,90	20,87%	11599,64	**20,04%**	11677,56	20,90%	765,6818	21,37%	9643,828
t=7	**n=10**	6603,97	20,60%	6356,25	16,39%	6501,222	18,31%	304,83	**15,72%**	5481,396
	n=20	9544,19	25,22%	9243,56	21,75%	9629,834	26,31%	597,711	**20,93%**	7613,238
	n=30	12579,02	**23,21%**	12825,46	25,60%	12582,54	**23,21%**	275,8988	23,77%	10211,74
	n=40	14687,76	24,23%	14843,74	25,59%	14795	25,17%	416,6992	**22,87%**	11816,84
	n=50	16532,28	24,63%	16528,46	24,58%	16504,86	24,46%	783,718	**21,88%**	13263,4
Averages			21%		16,6%		20.6%		15.9%	

The RGRASPSA approach presents the better average GAP, achieving better average solutions for most instances when compared against the B&C.

The B&C algorithm used as comparison presents better computational times for small instances as the developed algorithms present much less computational time with the GRASP achieving 63 times smaller times on average for the last 5 instances of 50 customers and 7 periods, the

Table 5.5: Computational Times Comparison

		GRASP	RGRASP	GRASPSA	RGRASPSA	B&C
t=3	n=10	176	245	280	3138,1	0
	n=20	182	258	441	4612,9	1,6
	n=30	470	685	438	5050,6	13,2
	n=40	184	276	278	6140,1	101,8
	n=50	374	592	445	6735,2	299,6
t=5	n=10	367	608	255	5130,5	1
	n=20	296	411	850	5699,3	20,4
	n=30	457	173	204	7080,4	182,2
	n=40	268	406	393	9217,6	9464,6
	n=50	646	924	755	10257	20712,6
t=7	n=10	539	708	239	4054	27,8
	n=20	319	433	734	7124,3	3335,6
	n=30	132	178	262	10366	28529
	n=40	279	393	417	1155,	43200
	n=50	684	933	773	13644	43187,8

GRASPSA and RGRASPSA present a 55 times decrease in computational time and the worst case, the RGRASP presents 43 times improved computational time.

Chapter 6

Conclusions

In the nowadays economic conjecture where companies struggle to keep or enhance their competitiveness, cost reduction policies stand at the core of an enterprise operational management. It becomes more and more imperative to hold market shares, improve profits and reduce operational costs. This growing need of improvement leads to higher necessity of optimization methods. Metaheuristics represent a growing branch in optimization trying to fill the void for improving and fast-response solutions. As such this dissertation tries to shorten that gap by trying to solve an IRP using a metaheuristic optimization method.

In this thesis, the Inventory Routing Problem is addressed by performing an extensive study of the literature concerning the IRP, introducing the current state of the art for optimization algorithms, from exact to approximated algorithms, with a special focus in metaheuristics which was the elected optimization method to study in the dissertation. A mathematical model for the MMIRPTW is proposed which has a large realistic applicability taking into account that many logistics operators depend on a large number of vehicles, many times with heterogeneous capacities and many supply chains appear restricted in terms of delivery windows.

Four GRASP variations were developed and studied in order to assess the GRASP strength in solving big-sized NP-hard problems.

When looking into the results of each algorithm it is clear that the RGRASP variation usually holds better results which comes from the fact it tries to find the better α for each instance permitting the algorithm to adapt to each different situation. The implemented k-opt algorithm was considered successful as it was able to completely remove any crossover in all the tested routes making redundant the aggregation of the SA methodology as it was not able to significantly improve the overall results. The SA improvements are not significant which may come from the fact that the initially constructed solutions are already placed near a local optimum denying a satisfactory diversification.

The developed algorithms have shown potential in solving big sized instances as they present much shorter computational times. However the developed algorithms did not present a small enough GAP to be considered successful when compared to a similar approach developed by

Guemri et al. (2016). This predicament is justified by the lack of optimization of the inventory sub-problem as no LS was implemented for the scheduling of the visits. In other words, the LS phase only improves the routing part which considers a fixed visit schedule. This prevents the designed algorithms to thoroughly explore the search space, making the algorithm highly dependent of the initial solution constructed with GRASP.

Taking into account the planning horizon for this dissertation the developed algorithm is only capable of solving instances for one vehicle and one product, making it impossible to verify if the implemented strategy would hold better GAP results for different number products or vehicles.

6.1 Future Work

As future work it is recommended an improvement in the scheduling sub problem by the means of guaranteeing an qausi-optimal solution at the constructive phase of the sub problem, or by the means of improving the LS phase. For the second approach a path-relinking or an inter-period swap-based heuristic are suggested as they were the most likely next steps in this study if it had a longer time horizon. It would also be interesting to adapt the algorithm so as to accept different values of products and vehicles assessing the algorithms capacity to deliver good solutions for even bigger instances and possibly adding the time-windows and turning the addressed problem into the MMIRPTW modeled in Section 3.

References

Shafiq Alam, Gillian Dobbie, and Saeed Ur Rehman. Analysis of particle swarm optimization based hierarchical data clustering approaches. *Swarm and Evolutionary Computation*, 25:36 – 51, 2015. ISSN 2210-6502. doi: http://dx.doi.org/10.1016/j.swevo.2015.10.003. URL http://www.sciencedirect.com/science/article/pii/S2210650215000784. SI: {RAMONA}.

UF Aminu and Richard W Eglese. A constraint programming approach to the chinese postman problem with time windows. *Computers & Operations Research*, 33(12):3423–3431, 2006.

Claudia Archetti, Luca Bertazzi, Gilbert Laporte, and Maria Grazia Speranza. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Science*, 41(3): 382–391, 2007.

Claudia Archetti, Luca Bertazzi, Alain Hertz, and M Grazia Speranza. A hybrid heuristic for an inventory routing problem. *INFORMS Journal on Computing*, 24(1):101–116, 2012.

Ronald G Askin and Mingjun Xia. Hybrid heuristics for infinite period inventory routing problem.

Raúl Baños, Julio Ortega, Consolación Gil, Antonio L Márquez, and Francisco De Toro. A hybrid meta-heuristic for multi-objective vehicle routing problems with time windows. *Computers & Industrial Engineering*, 65(2):286–296, 2013.

Jonathan F Bard and Thomas A Feo. The cutting path and tool selection problem in computer aided process planning. *Journal of Manufacturing Systems*, 8(1):17–26, 1989.

Jonathan F. Bard and Narameth Nananukul. Heuristics for a multiperiod inventory routing problem with production decisions. *Computers & Industrial Engineering*, 57(3):713 – 723, 2009. ISSN 0360-8352. doi: http://dx.doi.org/10.1016/j.cie.2009.01.020. URL http://www.sciencedirect.com/science/article/pii/S0360835209000370.

Walter J Bell, Louis M Dalberto, Marshall L Fisher, Arnold J Greenfield, Ramchandran Jaikumar, Pradeep Kedia, Robert G Mack, and Paul J Prutzman. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces*, 13(6):4–23, 1983.

Farah Belmecheri, Christian Prins, Farouk Yalaoui, and Lionel Amodeo. Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows. *Journal of intelligent manufacturing*, 24(4):775–789, 2013.

Russell Bent and Pascal Van Hentenryck. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 38(4):515–530, 2004. doi: 10.1287/trsc.1030.0049. URL http://pubsonline.informs.org/doi/abs/10.1287/trsc.1030.0049.

Luca Bertazzi, Giuseppe Paletta, and M Grazia Speranza. Deterministic order-up-to level policies in an inventory routing problem. *Transportation Science*, 36(1):119–132, 2002.

Luca Bertazzi, Adamo Bosco, Francesca Guerriero, and Demetrio Lagana. A stochastic inventory routing problem with stock-out. *Transportation Research Part C: Emerging Technologies*, 27: 89–107, 2013.

M. Boudia, M.A.O. Louly, and C. Prins. A reactive {GRASP} and path relinking for a combined production–distribution problem. *Computers & Operations Research*, 34(11):3402 – 3419, 2007. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2006.02.005. URL http://www.sciencedirect.com/science/article/pii/S0305054806000426.

David Bredström, Kjetil Haugen, Asmund Olstad, and Jan Novotnỳ. A mixed integer linear programming model applied in barge planning for omya. *Operations Research Perspectives*, 2: 150–155, 2015.

Jens Clausen. Branch and bound algorithms-principles and examples. *Department of Computer Science, University of Copenhagen*, pages 1–30, 1999.

Leandro C Coelho and Gilbert Laporte. A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. *International Journal of Production Research*, 51(23-24): 7156–7169, 2013a.

Leandro C Coelho and Gilbert Laporte. The exact solution of several classes of inventory-routing problems. *Computers & Operations Research*, 40(2):558–565, 2013b.

Leandro C Coelho, Jean-François Cordeau, and Gilbert Laporte. Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, 24:270–287, 2012a.

Leandro C Coelho, Jean-François Cordeau, and Gilbert Laporte. The inventory-routing problem with transshipment. *Computers & Operations Research*, 39(11):2537–2548, 2012b.

Leandro C Coelho, Jean-François Cordeau, and Gilbert Laporte. Thirty years of inventory routing. *Transportation Science*, 48(1):1–19, 2013.

David Connolly. General purpose simulated annealing. *The Journal of the Operational Research Society*, 43(5):495–505, 1992. ISSN 01605682, 14769360. URL http://www.jstor.org/stable/2583568.

Van-Dat Cung, Simone L Martins, Celso C Ribeiro, and Catherine Roucairol. Strategies for the parallel implementation of metaheuristics. In *Essays and surveys in metaheuristics*, pages 263–308. Springer, 2002.

Cedric Davies and Pawan Lingras. Genetic algorithms for rerouting shortest paths in dynamic and stochastic networks. *European Journal of Operational Research*, 144(1):27 – 38, 2003. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/S0377-2217(01)00354-X. URL http://www.sciencedirect.com/science/article/pii/S037722170100354X.

Guy Desaulniers, Jørgen G Rakke, and Leandro C Coelho. A branch-price-and-cut algorithm for the inventory-routing problem. *Transportation Science*, 2015.

Abraham Duarte, Jesús Sánchez-Oro, Mauricio G.C. Resende, Fred Glover, and Rafael Martí. Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization. *Information Sciences*, 296:46 – 60, 2015. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2014.10.010. URL http://www.sciencedirect.com/science/article/pii/S0020025514009906.

Thomas A Feo and Mauricio G.C Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67 – 71, 1989. ISSN 0167-6377. doi: http://dx.doi.org/10.1016/0167-6377(89)90002-3. URL http://www.sciencedirect.com/science/article/pii/0167637789900023.

Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

Thomas A Feo, Kishore Sarathy, and John McGahan. A grasp for single machine scheduling with sequence dependent setup costs and linear delay penalties. *Computers & Operations Research*, 23(9):881–895, 1996.

F. Glover, M. Laguna, and R. Marti. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, Vol. 29, no 3:653–684, 2000.

J.J. Grefenstette. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. Taylor & Francis, 2014. ISBN 9781317760252. URL https://books.google.pt/books?id=lI17AgAAQBAJ.

Stanley E Griffis, John E Bell, and David J Closs. Metaheuristics in logistics and supply chain management. *Journal of Business Logistics*, 33(2):90–106, 2012.

Oualid Guemri, Abdelghani Bekrar, Bouziane Beldjilali, and Damien Trentesaux. Grasp-based heuristic algorithm for the multi-product multi-vehicle inventory routing problem. *4OR*, pages 1–28, 2016.

Kristina Holmqvist, Athanasios Migdalas, and Panos M Pardalos. Greedy randomized adaptive search for a location problem with economies of scale. In *Developments in global optimization*, pages 301–313. Springer, 1997.

Irina Ioachim, Sylvie Gelinas, Francois Soumis, and Jacques Desrosiers. A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31 (3):193–204, 1998.

L. Jourdan, M. Basseur, and E.-G. Talbi. Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, 199(3):620 – 629, 2009. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2007.07.035. URL http://www.sciencedirect.com/science/article/pii/S0377221708003597.

Scott Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34(5-6):975–986, 1984.

Philip N Klein and Neal E Young. Approximation algorithms for np-hard optimization problems. In *Algorithms and theory of computation handbook*, pages 34–34. Chapman & Hall/CRC, 2010.

George Kontoravdis and Jonathan F Bard. A grasp for the vehicle routing problem with time windows. *ORSA journal on Computing*, 7(1):10–23, 1995.

İlker Küçükoğlu and Nursel Öztürk. An advanced hybrid meta-heuristic algorithm for the vehicle routing problem with backhauls and time windows. *Computers & Industrial Engineering*, 86: 60–68, 2015.

David S.W. Lai, Ozgun Caliskan Demirag, and Janny M.Y. Leung. A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph. *Transportation Research Part E: Logistics and Transportation Review*, 86:32 – 52, 2016. ISSN 1366-5545. doi: http://dx.doi.org/10.1016/j.tre.2015.12.001. URL http://www.sciencedirect.com/science/article/pii/S1366554515002264.

M. Lozano and C. García-Martínez. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Computers & Operations Research*, 37(3):481 – 497, 2010. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2009.02.010. URL http://www.sciencedirect.com/science/article/pii/S0305054809000471. Hybrid Metaheuristics.

Hao-Chun Lu and Yao-Huei Huang. An efficient genetic algorithm with a corner space algorithm for a cutting stock problem in the tft-lcd industry. *European Journal of Operational Research*, 246(1):51 – 65, 2015. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2015.04.044. URL http://www.sciencedirect.com/science/article/pii/S0377221715003379.

Nandar Lynn and Ponnuthurai Nagaratnam Suganthan. Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. *Swarm and Evolutionary Computation*, 24:11 – 24, 2015. ISSN 2210-6502. doi: http://dx.doi.org/10.1016/j.swevo.2015.05.002. URL http://www.sciencedirect.com/science/article/pii/S2210650215000401.

SMJ Mirzapour Al-e hashem and Yacine Rekik. Multi-product multi-period inventory routing problem with a transshipment option: A green approach. *International Journal of Production Economics*, 157:80–88, 2014.

Anis Mjirda, Bassem Jarboui, Rita Macedo, Saïd Hanafi, and Nenad Mladenović. A two phase variable neighborhood search for the multi-product inventory routing problem. *Computers & Operations Research*, 52, Part B:291 – 299, 2014. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2013.06.006. URL http://www.sciencedirect.com/science/article/pii/S0305054813001639. Recent advances in Variable neighborhood search.

N.H. Moin, S. Salhi, and N.A.B. Aziz. An efficient hybrid genetic algorithm for the multi-product multi-period inventory routing problem. *International Journal of Production Economics*, 133(1):334 – 343, 2011. ISSN 0925-5273. doi: http://dx.doi.org/10.1016/j.ijpe.2010.06.012. URL http://www.sciencedirect.com/science/article/pii/S0925527310002203. Leading Edge of Inventory Research.

V. Nwana, K. Darby-Dowman, and G. Mitra. A co-operative parallel heuristic for mixed zero–one linear programming: Combining simulated annealing with branch and bound. *European Journal of Operational Research*, 164(1):12 – 23, 2005. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2002.12.002. URL http://www.sciencedirect.com/science/article/pii/S0377221703008324.

Yang-Byung Park, Jun-Su Yoo, and Hae-Soo Park. A genetic algorithm for the vendor-managed inventory routing problem with lost sales. *Expert Systems with Applications*, 2016.

Gilles Pesant, Michel Gendreau, Jean-Yves Potvin, and Jean-Marc Rousseau. An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Transportation Science*, 32(1):12–29, 1998.

Duc Pham and Dervis Karaboga. *Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks*. Springer Science & Business Media, 2012.

Marcelo Prais and Celso C Ribeiro. Reactive grasp: An application to a matrix decomposition problem in tdma traffic assignment. *INFORMS Journal on Computing*, 12(3):164–176, 2000.

Amilkar Puris, Rafael Bello, and Francisco Herrera. Analysis of the efficacy of a two-stage methodology for ant colony optimization: Case of study with {TSP} and {QAP}. *Expert Systems with Applications*, 37(7):5443 – 5453, 2010. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2010.02.069. URL http://www.sciencedirect.com/science/article/pii/S0957417410001089.

Lei Qin, Lixin Miao, Qingfang Ruan, and Ying Zhang. A local search method for periodic inventory routing problem. *Expert Systems with Applications*, 41(2):765–778, 2014.

MAURICIO GC Resende. Greedy randomized adaptive search procedures (grasp). *AT&T Labs Research Technical Report*, 98(1):1–11, 1998.

Mauricio GC Resende and Celso C Ribeiro. Greedy randomized adaptive search procedures: Advances and applications. *Handbook of Metaheuristics, 2nd edn. Springer, New York*, 2008.

Ahmad Rusdiansyah and De bi Tsao. An integrated model of the periodic delivery problems for vending-machine supply chains. *Journal of Food Engineering*, 70(3):421 – 434, 2005. ISSN 0260-8774. doi: http://dx.doi.org/10.1016/j.jfoodeng.2004.05.073. URL http://www.sciencedirect.com/science/article/pii/S026087740400490X. Operational Research and Food Logistics.

Robert A. Russell and Wen-Chyuan Chiang. Scatter search for the vehicle routing problem with time windows. *European Journal of Operational Research*, 169(2):606 – 622, 2006. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2004.08.018. URL http://www.sciencedirect.com/science/article/pii/S0377221704005570. Feature Cluster on Scatter Search Methods for Optimization.

Kanchana Sethanan and Woraya Neungmatcha. Multi-objective particle swarm optimization for mechanical harvester route planning of sugarcane field operations. *European Journal of Operational Research*, pages –, 2016. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2016.01.043. URL http://www.sciencedirect.com/science/article/pii/S0377221716000886.

Oguz Solyali and Haldun Süral. A branch-and-cut algorithm using a strong formulation and an a priori tour-based heuristic for an inventory-routing problem. *Transportation Science*, 45(3):335–345, 2011.

Jin-Hwa Song and Kevin C Furman. A maritime inventory routing problem: Practical approach. *Computers & Operations Research*, 40(3):657–665, 2013.

El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

Pieter Vansteenwegen and Manuel Mateo. An iterated local search algorithm for the single-vehicle cyclic inventory routing problem. *European Journal of Operational Research*, 237(3):802 – 813, 2014. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2014.02.020. URL http://www.sciencedirect.com/science/article/pii/S0377221714001350.

Zhuo Wang, Zhipeng Lü, and Tao Ye. Multi-neighborhood local search optimization for machine reassignment problem. *Computers & Operations Research*, 68:16 – 29, 2016. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2015.10.015. URL http://www.sciencedirect.com/science/article/pii/S0305054815002464.

Wei Xiang, Jiao Yin, and Gino Lim. An ant colony optimization approach for solving an operating room surgery scheduling problem. *Computers & Industrial Engineering*, 85:335 – 345, 2015. ISSN 0360-8352. doi: http://dx.doi.org/10.1016/j.cie.2015.04.010. URL http://www.sciencedirect.com/science/article/pii/S036083521500159X.

Zhizhong Zeng, Xinguo Yu, Kun He, Wenqi Huang, and Zhanghua Fu. Iterated tabu search and variable neighborhood descent for packing unequal circles into a circular container. *European Journal of Operational Research*, 250(2):615 – 627, 2016. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2015.09.001. URL http://www.sciencedirect.com/science/article/pii/S037722171500822X.