Applications of Biased Randomization and Simheuristic Algorithms to Arc Routing and Facility Location Problems

Sergio González-Martín

sgonzalezmarti@uoc.edu

Department of Computer Science, Multimedia and Telecommunication

Universitat Oberta de Catalunya – Internet Interdisciplinary Institute (UOC-IN3)

Advisors: Dr. Angel A. Juan & Dr. Daniel Riera

PhD. Thesis Dissertation

Information and Knowledge Society Doctoral Programme

Network and Information Technologies

Barcelona, February 2015

Abstract

Most metaheuristics contain a randomness component, which is usually based on uniform randomization –i.e., the use of the Uniform probability distribution to make random choices. However, the Multi-start biased Randomization of classical Heuristics with Adaptive local search framework **(MIRHA, Gonzalez-Martin et al., 2014a; Juan et al. ,2014a)** proposes the use of biased (non-uniform) randomization for the design of alternative metaheuristics -i.e., the use of skewed probability distributions such as the Geometric or Triangular ones. In some scenarios, this non-biased randomization has shown to provide faster convergence to near-optimal solutions. The MIRHA framework also includes a local search step for improving the incumbent solutions generated during the multi-start process. It also allows the addition of tailored local search components, like cache (memory) or splitting (divide-and-conquer) techniques, that allow the generation of competitive (near-optimal) solutions. The algorithms designed using the MIRHA framework allows to obtain 'high-quality' solutions to realistic problems in reasonable computing times. Moreover, they tend to use a reduced number of parameters, which makes them simple to implement and configure in most practical applications. This framework has successfully been applied in many routing and scheduling problems.

One of the main goals of this thesis is to develop new algorithms, based in the aforementioned framework, for solving some combinatorial optimization problems that can be of interest in the telecommunication industry (**Figure 1**). One of the current issues in the telecommunication sector is that of planning of telecommunication networks. For instance, finding the best placements of base station on a mobile network, the placement of optical fiber networks or selecting the servers which could provide a given service with the lowest network latency, are examples of these problems. Two known families of combinatorial optimization problems that can be used for modeling the aforementioned problems are the Facility Location Problems (FLP) and the Arc Routing Problems (ARP). In this thesis we will propose new algorithms based on the MIRHA framework for problems belonging to these two families.

The FLP is a location problem where the goal is to find the best placement of some facilities which minimizes the costs of providing a service to a set of customers.

The ARP is similar to the well-known Vehicle Routing Problem (VRP), but its nature makes it also a good candidate for modeling certain real-life telecommunication problems. While the VRP has been extensively studied in the literature -mainly because of its applications to logistics and transportation-, there is much less research in the ARP and its potential applications to other fields.

Regarding the ARP, we will work with several variations in order to model a wide range of real-life problems. Thus, we will also consider the ARP under uncertainty conditions. In this scenario, Simheuristics (the combination of simulation with metaheuristics, **Juan et al., 2011a)** is a useful tool that can be combined with the MIRHA framework in order to obtain robust solutions to the stochastic version of the problem.

Keywords: Biased Randomized Heuristics, Simheuristics, Arc Routing Problem, Facility Location Problem, Optimization, Telecommunication applications, Logistic & Transportation.

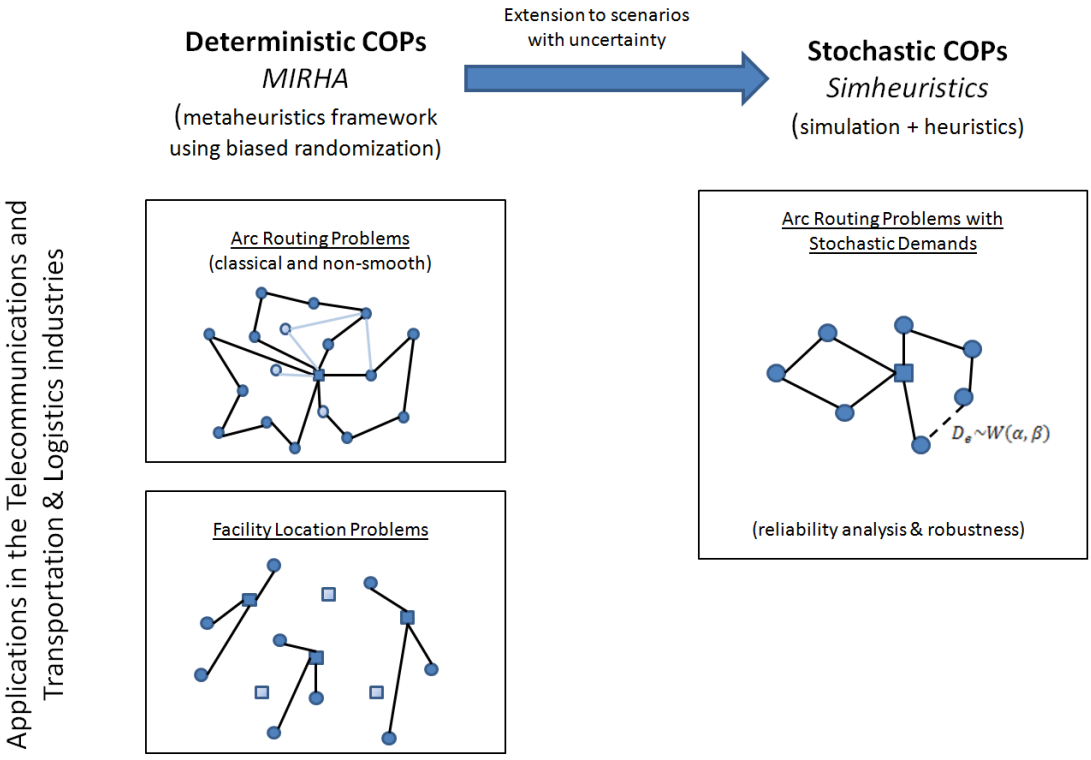

Figure 1. Summary of the key topics discussed on this dissertation.

Resum

La majoria de *metaheuristiques* tenen una component aleatori, que normalment està basada en aleatorització uniforme –i.e., l'ús de la distribució de probabilitat uniforme per fer seleccions aleatòries. Per altra banda, el marc *Multi-start biased Randomization of classical Heuristics with Adaptive local search* **(MIRHA, Gonzalez-Martin et al. 2014a; Juan et al., 2014a)** proposa l'ús de aleatorització esbiaixada (no uniforme) per al disseny de algoritmes *metaheuristics* alternatius –i.e., l'ús de distribucions de probabilitat esbiaixades com la geomètrica o la triangular. En algunes situacions, aquesta aleatorització no uniforme ha obtingut una convergència més ràpida a la solució quasi òptima. El marc MIRHA també inclou un pas de cerca local per a millorar les solucions generades durant el procés iteratiu. A més, permet afegir passos de cerca adaptats al problema, com *cache* (memòria) o *splitting* (dividir i conquerir), que permeten la generació de solucions competitives (quasi òptimes). Els algoritmes dissenyats amb el marc MIRHA permeten obtenir solucions d'alta qualitat a problemes realistes en temps de computació raonables. A més, tendeixen a utilitzar un nombre reduït de paràmetres, el que els fa simples d'implementar i configurar en la majoria d'aplicacions pràctiques. El marc s'ha aplicat exitosament a diversos problemes d'enrutament i planificació.

Un dels principals objectius d'aquesta tesi és desenvolupar nous algoritmes , basats en el marc mencionat, per solucionar problemes d'optimització combinatòria que poden ser d'interès a la industria de les telecomunicacions (**Figure 1**). Un dels problemes actuals en el sector de les telecomunicacions es aquell de la planificació de xarxes de telecomunicació. Per exemple, trobar les millors situacions per les estacions base en una xarxa mòbil, la ubicació de xarxes de fibra òptica, o la selecció dels servidors que poden proveir un determinat servei amb la menor latència de xarxa, són exemples d'aquest problemes. Dues famílies conegudes de problemes d'optimització combinatòria que poden utilitzar se per a modelar aquestes situacions son el *Facility Location Problem* (FLP) i l'*Arc Routing Problem* (ARP). En aquesta tesi proposarem nous algoritmes basats en el marc MIRHA per a problemes pertanyents a aquestes dues famílies.

El FLP es un problema de localització on l'objectiu es trobar les millors ubicació de uns nodes proveïdors de serveis que minimitzin el cost de proveir aquest servei a tots els clients. El ARP és un problema semblant al conegut *Vehicle Routing Problem* (VRP), però la seva natura el fa també útil per modelar alguns problemes reals de telecomunicacions. Mentre que el VRP s'ha estudiat extensivament en la literatura existent –principalment per les seves aplicacions al transport i la logístics-, per a l'ARP hi ha molta menys investigació existent.

En quant a l'ARP, treballarem amb diverses variacions del problema per tal de poder modelar un rang més ampli d'aplicacions a la vida real. Així, també considerarem el ARP sota condicions d'incertesa. En aquest escenari, Simheuristics (la combinació de simulació i metaheuristics, Juan et al. 2011a) en una eina molt útil que es pot combinar amb el marc MIRHA per a obtenir solucions robustes per a la versió estocàstica del problema.

Paraules clau: Randomització esbiaixada d'heurístiques, Simheuristics, Arc Routing Problem, Facility Location Problem, Optimització, Aplications en Telecomunicacions i Transport & Logistica.

Acknowledgements

First of all, I would like to thank my thesis advisors Dr. Angel A. Juan and Dr. Daniel Riera for their always valuable support, motivation, enthusiasm and continuous guidance during the elaboration of this thesis. Without their help, this thesis would not have been possible. Also I would like to thank Dr. Juan J. Ramos for accept being the external advisor of this thesis and his insightful comments.

I would like to thank also to all the people in the IN3-UOC with I have had the opportunity to collaborate in one or other way during these years. Special mentions to my fellows from the PhD studies Jose Caceres and Guillem Cabrera who have made me feel part of the department despite being a part-time student. To David Fernandez, Quim Castella and Enoc Martinez for their help and support in the development of different parts of this work. To Alejandra Perez for being always open to discus and clarify different concepts.

Lots of thanks also to my friends Oriol, Jordi, David, Albert and Vero for always being open to hear my problems, and always giving good advices. Also to a long list of colleagues from Presence Technology first, and Roche Diagnostics now, for encouraging me to reach the end of this work.

Finally, lots of thanks to my family, especially to my wife Laura and my son Aimar, for their patience during the long after-work hours for completing this work.

Contents

List of Figures

List of Tables

Glossary

Abbreviation	Description
ARP	Arc Routing Problem
ARPSD	Arc Routing Problem with Stochastic Demands
BKS	Best Known Solution
CFLP	Connected Facility Location Problem
COP	Combinatorial Optimization Problem
CVOP	ConVex Optimization Problem
DPCS	Distributed Parallel and Collaborative Systems
FLP	Facility Location Problem
FTTH	Fiber To The Home
GRASP	Greedy Randomized Adaptive Search Procedure
HAROSA	Hybrid Algorithms for solving Realistic rOuting, Scheduling and Availability problems.
HBSS	Heuristic Biased Stochastic Sampling
ILP	Integer Linear Programming
IN3	Internet Interdisciplinary Institute
MCS	Monte-Carlo Simulation
MDVRP	Multi Depot Vehicle Routing Problem
MIRHA	Multi-start bIased Randomization of classical Heuristics with Adaptive local search
NCVOP	Non-ConVex Optimization Problem
OBS	Our Best found Solution
PON	Passive Optical Network
PSH	Path Scanning Heuristic
RNG	Random Number Generator
SA	Simulated Annealing
Simheuristic	Simulation-optimization using Heuristics
STP	Steiner Tree Problem
T&L	Transportation and Logistics
TS	Tabu Search
TSP	Traveling Salesman Problem
UOC	Universitat Oberta de Catalunya
VPN	Virtual Private Network
VRP	Vehicle Routing Problem
VRPSD	Vehicle Routing Problem with Stochastic Demands
WCN	Wireless Community Network
WSN	Wireless Sensors Network

Table 1. List of abbreviation used on this dissertation

1 Introduction

Combinatorial problems involve finding a grouping, ordering or assignment of a discrete, finite set of objects that satisfies given conditions. Combinatorial Optimization Problems (COPs) consist on finding the optimal or near-optimal solution for a given cost function. In the case of NP-hard problems, metaheuristics are usually a good alternative to exact methods in order to find near-optimal solutions in reasonable computing times. Metaheuristics are a class of algorithms which provide approximation solutions of certain quality, while having relatively low execution times even for large-scale NP-hard problems. Many metaheuristics contain a randomness component, which in most of the cases is based on Uniform randomization -i.e., the use of the Uniform probability distribution to make random choices. However, as showed in different works related to this thesis, use of biased (non-uniform) randomization -i.e., random variates generated from skewed probability distributions- can become a more efficient strategy when introducing randomness into a metaheuristic. The Multi-start biased Randomization of classical Heuristics with Adaptive local search **(MIRHA, Gonzalez-Martin et al., 2014a; Juan et al. ,2014a)**, is a general framework to develop biased-randomized metaheuristics. MIRHA has demonstrated to be helpful for defining competitive algorithms in a wide range of vehicle routing and scheduling problems.

COPs have application in many real-life scenarios. Examples can be found in key economic sectors, like telecommunications or transportation and logistics (T&L). The telecommunication sector is one of the key role players in current economy. The evolution experienced, with considerable improvements in network infrastructures and capabilities, allow governments and enterprises to offer new electronic services to their customers. This evolution has been in part promoted due to the liberalization of the sector in many countries, where, according to the International Telecommunication Union (ITU) in its annual world information society report **(Choudhary et al. 2008)**, “the number of telecom regulatory authorities grew from 14 in 1990 to 147 in 2007 and partial or full privatization of incumbent telecom operators grew from 10% in 1990 to

50% in 2006". The growth of the sector in the last two decades has been very fast. Apart from the market liberalization, it can be explained by the advances in telecommunication technology, and the market privatization **(Lam et al. 2010)**. This makes the advances in telecommunications technology one of the driving forces of globalization and the rapid growth of the world's economy.

This importance and the sector's fast growth, has brought with it new optimization problems in several related fields. In **Partridge (2011)**, up to forty open research questions in the telecommunications arena are presented. They represent big challenges for the future: to define a network topology which can vary over the time, to define a new network addressing paradigm or to provide networks with resource auctions which could allow service providers to shift resources among customers in real time, among others. Most of these research questions have associated different combinatorial optimization problems and, in particular, routing, assignment, availability, and scheduling ones.

Likewise, transportation and logistics also has a great economic importance in most countries. The costs associated with transportation have growth considerably over the last decade, especially due to the raise of petroleum prices. Also environmental policies, like the Kyoto protocol, are forcing enterprises to consider these environmental conditions when planning routes, in addition to the direct costs associated with the transport **(Vieira et al. 2007)**.

Both T&L and telecommunication routing and assignment problems are suitable to be studied with optimization (exact and approximate methods) and simulation techniques. Also, distributed and parallel-computing systems allow the practical development of new solutions to these problems which are considered to be computationally very difficult, being many of them proven to be NP-hard (e.g. **Al-Karaki et al. 2004)**. The MIRHA approach for solving deterministic NP-hard problems in the telecommunications and L&T areas, as well as its natural extension throughout the concept of Simheuristics **(Juan et al., 2011a)** for solving stochastic versions of these problems, are the main pillars over which this thesis is based.

1.1 Structure of this Thesis

This thesis studies different optimization problems, designing algorithms for them based on the aforementioned MIRHA and Simheuristics frameworks, and evaluating the performance compared with the state of the art. To this end, the thesis is structured in the following chapters:

⇒ *Chapter 2: MIRHA Framework and Simheuristics*. This dissertation will start by introducing the MIRHA framework and the concept of Simheuristics. Both are closely related as they are based on the use of biased randomization for solving complex optimization problems. They have been proven to be useful for solving routing, scheduling and availability problems, especially in the field of Transportation and Logistics. With this thesis we are aiming to adapt these algorithms to be applied in other optimization problems which are more suitable in other field like is the case of Telecommunications.

⇒ *Chapter 3: Capacitated Arc Routing Problem*. The first problem studied is the Capacitated Arc Routing Problem or CARP. This is a deterministic routing problem very similar to the Capacitated Vehicle Routing Problem or CVRP, but which is more suitable for modeling certain telecommunication routing optimization problems. The MIRHA framework has been successfully used for defining algorithms for the CVRP, but has never been used for defining algorithms for the CARP. In this chapter we will propose a new heuristic for the CARP, the SHARP heuristic, and use it as a base heuristic for the MIRHA framework in order to define a new competitive algorithm. We will compare the performance of our new proposed heuristic and the randomized algorithm with a classical heuristic, and also with the best known solutions available in the literature.

⇒ *Chapter 4: Arc Routing Problem with Stochastic Demands*. In this chapter we will introduce a stochastic component in the customer demands which is also more suitable for modeling certain real-life telecommunication routing problems, where usually the customer demands cannot be known beforehand, but can be modeled using data from the past. In this case we will combine the algorithm defined for the deterministic problem in **Chapter 3** with the Simheuristics framework to define a robust algorithm for the ARPSD. We will evaluate the robustness and performance of the proposed methodology using different datasets available on the literature.

⇒ *Chapter 5: Non-smooth Arc Routing Problem*. An additional variation of the CARP is that in which the optimization function is non-smooth. In this case, the capacity constraint is converted to a soft-constraint, so it can be violated but incurring in some penalty. This adds a non-smooth component to the cost function which makes the problem a Non-smooth ARP. This is also suitable for modeling certain real life scenarios and in this chapter we will evaluate how the proposed algorithm for the CARP behaves.

⇒ *Chapter 6: Facility Location Problem*. The final problem of study is the Facility Location Problem or FLP. This problem is an assignment problem which is very useful for modeling telecommunication problems like finding the best placements of servers for providing some services to a set of customers. For this problem we will completely define a new algorithm based on the MIRHA framework and will assess its performance with some data sets available on the literature.

⇒ *Chapter 7: Conclusions and Contributions Derived from this Thesis*. On this chapter we will extract some conclusions from the research work and state some open points to be considered by future researches. Also, this final chapter collects all the publications that have been produced as a result of the work carried on with this thesis.

In all these chapters we can appreciate the adaption of some heuristics to different optimization contexts and constraints. We explore how the integration of heuristics, simulation, biased randomization and parallel computing, can help to design high quality algorithms for problems which have real-life application in the Telecommunication arena (see **section 1.2** for specific examples). The algorithms are tested with different benchmarks available in the literature, comparing the results with the state of the art. Some quantitative methods are used to analyze the generated results where we remark the high quality of the proposed methods, especially in terms of execution times and gaps with respect to the best known solution.

1.2 Motivation

The MIRHA framework and Simheuristics have demonstrated to be very helpful for solving complex computational optimization problems with applications focused in the field of T&L. Being based in quite simple concepts, they make quite straightforward to define new algorithms based on this framework. The produced algorithms have usually a single configuration parameter or even without parameter. This makes that the time to deploy the algorithm in a real environment is faster as it avoids the long and complex fine tuning phase which usually is required by other metaheuristics. The promising results obtained by the algorithms designed with the framework, open the opportunity to afford new problems to which create algorithms based on MIRHA. Also, the flexibility of the framework makes it quite straightforward to combine produced algorithms with additional problem-specific steps. One example is the Iterated Local Search **(ILS, Lourenço et al. 2010)**.

One important Telecommunication problem in real life is that of planning and deployment of telecommunication networks, both physical (by means of physical cable connection) and virtual (by means of service providing). In this thesis we will focus on problems which are suitable for modeling those kinds of problems. For instance, the Arc Routing Problem (which is introduced on **Chapter 3**) can be used for minimizing the total length of cable for deploying an optical fiber network in order to interconnect all the required cabinets in a Fiber-To-The-Home (FTTH) scenario. Another example is the Facility Location Problem (which is introduced on **Chapter 6**), which can be useful for determining the best locations of optical splitter that provide optical connectivity to subscribers in a given service area of a FTTH-Passive Optical Network (FTTH-PON).

Extending the MIRHA framework to new problems which have applications in other fields in addition to T&L, like is the case of Telecommunications, will make it more interesting as will increase the number of possible applications. Real-life applications make the problems more interesting to be studied, as it can get the attention of companies which can combine efforts with researchers and practitioners in order to produce more interesting results. Also, it can help to improve the evaluation of the algorithm under real world constraints, as the companies can provide researchers with such information. Usually results produced under real-life scenarios are not exactly the same as the ones produced using artificially generated data, so this will improve the performance evaluation of the algorithms.

1.3 Research Objectives

In general, a desirable or efficient optimization algorithm for a computational optimization problem should be able to generate results in a short period of time; should produce good or high quality solutions; should be simple to configure; flexible to be adapted to new constraints or new computing architectures; and easy to understand and implement **(Cordeau et al., 2002)**. One of the main goals of this thesis is the proposal of new algorithms based on the MIRHA framework and Simheuristics, for the Arc Routing Problem (ARP) and the Facility Location Problem (FLP). Neither MIRHA framework nor Simheuristics have been used previously for defining algorithms for both problems. The new problems to be studied have the common characteristic that they are suitable for modeling real-life Telecommunications problems, such as the deployment of an optical fiber network or the design of a Wireless Sensors Network (WSN).

To reach the general goal, several specific objectives should be achieved. First of all, we will design and implement new algorithms for every one of the studied

problems. These hybrid algorithms will be based on the MIRHA framework. To this end, for every problem we will present the application of the methodology to the problem at a higher level, and then we will introduce the pseudo-code of the proposed algorithm. The pseudo-code will be implemented as an application and executed with benchmarks extracted from the literatures. Secondly, the results produced by the algorithm will be collected and analyzed using statistical methods. With this, we will be able to compare the algorithm performance with the state of the art, being able to analyze the quality of the designed algorithms.

Finally, we also are willing to evaluate the performance of the algorithm under uncertainty scenarios, when combined with Simheuristics. For this, we will propose an algorithm which combines the algorithm designed for the deterministic problems with ideas from the Simheuristics framework. We will then evaluate the performance with a set of benchmarks available on the literature adapted for the uncertainty scenario, using statistical techniques for evaluate the performance and reliability of the designed solution.

1.4 Original Contributions

In the process of achieving the objectives described in previous section, we generate a series of original contributions. Among them, we summarize next the most relevant ones, which will be detailed in deep in the next chapters:

1. **The SHARP heuristic for the Capacitated Arc Routing Problem (CARP)**: We define a savings heuristics for the ARP, based on the Clarke and Wright Savings (CWS) heuristic for the CVRP. This heuristic has the characteristic of being fast to execute while obtain relatively good results, which makes it suitable to be used within the MIRHA framework.
2. **The RandPSH and RandSHARP algorithm for the CARP**: We propose two new randomized algorithms based on the MIRHA framework. Mainly they are defined using the same approach, but they only differ on the base heuristic being used. For the first, the classical Path Scanning Heuristic (PSH) is used; for the later, our original SHARP heuristic is used as the base heuristic. Our SHARP heuristics demonstrates to have a better performance when combined with MIRHA framework than the PSH, thus SHARP is more suitable for our MIRHA framework.
3. **The combination of RandSHARP with Simheuristics for the Arc Routing Problem with Stochastic Demands (ARPSD)**: For solving the stochastic version of the Arc Routing Problem, we propose the combination of the

randomized algorithm proposed for the deterministic problem with the Simheuristics concept. With that, we are able to obtain robust solutions for the problem. We also introduce a concept of upper and lower bound for evaluating the quality of the results obtained with our algorithm.

4. **The RandSHARP algorithm for the Non-Smooth Arc Routing Problem**: We have adapted the RandSHARP algorithm for the non-smooth variation of the ARP and evaluated its performance. With that we demonstrate that ideas from MIRHA framework also fit perfectly when the cost function being optimized is non-smooth.
5. **The RandCFH-ILS algorithm for the Facility Location Problem (FLP)**: We propose a new randomized algorithm based on the MIRHA framework for the FLP. The algorithm is able to compete with state of the art algorithms for the problem, with relatively low execution times.

1.5 Chapter Conclusions

In this first chapter, we have defined the motivation, context and objectives of this thesis. We have presented the relation of routing optimization contexts to human economy activities and its impacts on different sectors like telecommunications or transportation and logistics. Also, we enumerate the main contributions of this thesis which will be introduced in the rest of the dissertation. The next chapters will present the MIRHA framework and Simheuristics in deep, will introduce the ARP and several variations of it, and also the FLP which could be considered as an evolution of the former. For every problem, we will present its definition, mathematical notation or modeling and the proposal of competitive algorithms for solving them based on a common framework.

2 MIRHA Framework and Simheuristics

Parts of this chapter have been taken from the co-autored publications **Juan et al. (2014a)** and **Gonzalez-Martin et al. (2014a)**.

In this chapter, we will present the basic ideas that define the Multi-start biased Randomization of classical Heuristics with Adaptive local search (MIRHA) framework. Also, we will introduce the concept of Simheuristics, which is the combination of simulation with heuristics to solve problems which have some uncertainty component. The MIRHA framework offers a clear guidance for designing biased randomized heuristics based on classical problem-specific heuristics. The designed algorithms can be used in many deterministic optimization problems. Also, the algorithms can be combined with Simheuristics to cope with optimization problems which have a stochastic component.

Combinatorial Optimization Problems (COPs) have posed numerous challenges to the human mind over the past decades. From a theoretical perspective, they have a well-structured definition consisting of an objective function that needs to be minimized or maximized, and a series of constraints that must be satisfied. From a theoretical point of view, these problems have an interest on their own due to the mathematics involved in their modeling, analysis and solution. However, the main reason for which they have been so actively investigated is the tremendous amount of real-life applications that can be successfully modeled as a COP. Thus, for example, decision-making processes in fields such as logistics, transportation, and manufacturing contain plentiful hard challenges that can be expressed as COPs **(Faulin et al. (2012); Montoya et al. (2011))**. Accordingly, researchers from different areas –e.g. Applied Mathematics, Operations Research, Computer Science, and Artificial Intelligence– have directed their efforts to conceive techniques to model, analyze, and solve COPs.

A considerable number of methods and algorithms for searching optimal or near-optimal solutions inside the solution space have been developed. In some small-sized problems, the solution space can be exhaustively explored. For those instances, efficient exact methods can usually provide the optimal solution in a reasonable time.

Unfortunately, the solution space in most COPs is exponentially astronomical. Thus, in medium- or large-size problems, the solution space is too large and finding the optimum in a reasonable amount of time is not a feasible task. An exhaustive method that checks every single candidate in the solution space would be of very little help in these cases, since it would take exponential time. Therefore, a large amount of heuristics and metaheuristics have been developed in order to obtain near-optimal solutions, in reasonable computing times, for medium- and large-size problems, some of them even considering realistic constraints.

The main goal of this chapter is to present a hybrid scheme which combines classical heuristics with biased-randomization processes. As it will be discussed later, this hybrid scheme represents an efficient, relatively simple, and flexible way to deal with several COPs in different fields, even when considering realistic constraints.

2.1 The Methodology

2.1.1 Background

In the context of this chapter, we will refer to any algorithm which makes use of pseudo-random numbers to perform *random* choices during the exploration of the solution space by the term randomized search method, or simply randomized algorithm. This includes most current metaheuristics and stochastic local-search processes. Thus, since it does not follow a determinist path, even for the same input, a randomized search method can produce different outputs in different runs. Within these types of algorithms we can include, among others:

⇒ Genetic and Evolutionary algorithms **(Reeves, 2010)**.

⇒ Simulated Annealing **(Nikolaev and Jacobson, 2010)**.

⇒ Greedy Randomized Adaptive Search Procedure or GRASP **(Festa and Resende, 2009a, 2009b)**.

⇒ Variable Neighborhood Search **(Hansen et al., 2010)**.

⇒ Iterated Local Search **(Lourenço et al., 2010)**.

⇒ Ant Colony Optimization **(Dorigo and Stützle, 2010)**.

⇒ Probabilistic Tabu Search **(Lokketangen and Glover, 1998)**.

⇒ Particle Swarm Optimization **(Kennedy and Eberhart, 1995)**.

One of the most popular randomized search methods is GRASP. Roughly speaking, GRASP is a multi-start or iterative process which uses uniform random numbers and a restricted candidate list to explore the solution space **(Figure 2)**. At each iteration, two phases are executed: (i) the construction phase, which generates a

new solution by randomizing a classical heuristic; and (ii) a local search phase, which aims at improving the previously constructed solution. At the end of the multi-start process, the best solution found is kept as the current solution.

```
procedure GRASP(inputs)
01    while stopping criterion is not satisfied do
02       solution ← ConstructGreedyRandomizedSolution(inputs)
03       solution ← ApplyLocalSearch(solution)
04       bestSolution ← UpdateBestSolution(solution)
05    endwhile
06    return bestSolution
endprocedure
```

Figure 2. General pseudo-code for GRASP

It is interesting to notice that most of the work on randomized algorithms is based on the use of uniform randomness, i.e., randomness is generated throughout a symmetric (non-biased) uniform distribution. Thus, when we talk about biased randomization, we refer to the use of probability distributions, other than uniform, which do not distribute probabilities in a symmetric shape but in a non-symmetric or skewed one. Of course, these skewed distributions can also be used to induce biased randomness into an algorithm. As a matter of fact, as far as we know, the first approach based on the use of biased randomization of a classical heuristic is due to **Bresina (1996)**. The author proposed a methodology called Heuristic-Biased Stochastic Sampling (HBSS), which performs a biased iterative sampling of the search tree according to some heuristic criteria. Bresina applies the HBSS to a scheduling problem, and concludes that this approach outperforms greedy search within a small number of samples.

2.1.2 MIRHA

More recently, **Juan et al (2011a)** proposed the use of non-symmetric probability distributions to induce randomness in classical heuristics. Their general framework was called Multi-start biased Randomization of classical Heuristics with Adaptive local search (MIRHA). On this approach, the authors propose to combine classical greedy heuristics with pseudo-random variates from different, non-symmetric, probability distributions. In particular, the algorithm induced biased-randomness to slightly perturbate the greedy behavior of a classical heuristic, which transforms a deterministic heuristic into a probabilistic algorithm. According to the obtained results, the use of proper biased distributions –as an alternative to the use of the uniform distribution– contributes to explore the solution space in a more efficient way. **Figure 3** shows the pseudo-code of the MIRHA general framework. Similar to GRASP, a multi-start

procedure encapsulates the randomization of a heuristic, but this time a non-symmetric distribution will be employed instead. At each iteration, two processes are carried out. First, a new solution is constructed using a biased randomization version of the selected classical heuristic –which will depend on the particular problem being considered. Secondly, an adaptive local search is employed in order to improve the randomized solution. Notice that both the randomization effect and the multi-start process work together to reduce the probabilities that the procedure gets trapped into a local minimum, while ensuring that different feasible regions in the solution space are sampled and explored.

The common aspects of MIRHA with GRASP are the construction of an initial solution using randomization and, afterwards, the application of a local search. But there are relevant differences: (a) MIRHA does not use a restrictive candidate list, one main characteristic of the GRASP algorithm; and (b) MIRHA uses a non-symmetric distribution to select the next element to be included in the solution, while most GRASP implementations only consider uniform distributions. The HBSS proposed by Bresina is similar to MIRHA since it uses a biased distribution function combined with a sampling methodology. In fact, MIRHA can be seen as a natural extension/enhancement of the HBSS methodology, one which incorporates a local search step after each solution obtained by the biased sampling.

```
procedure MIRHA(inputs)
01   heuristic ← DefineHeuristic(inputs) %different for each COP
02   initialSolution ← GenerateSolution(heuristic, inputs)
03       bestSolution    ←    ApplyAdaptiveLocalSearch(initialSolution)
%different for each COP
04     probaDist  ←  DefineProbabilityDistribution(COP)  %different  for
each COP
05   while stopping criterion is not satisfied do
06     solution ← GenerateRandomSolution(heuristic, probaDist, inputs)
07      solution ← ApplyAdaptiveLocalSearch(solution)
08      bestSolution ← UpdateBestSolution(solution)
09   end while
10   return bestSolution
end procedure
```

Figure 3. General pseudo-code for MIRHA

2.2 Logic Behind our Approach

Most classical heuristics for solving combinatorial optimization problems employ an iterative process in order to construct a feasible –and hopefully good– solution. Examples of these heuristics are the Clarke and Wright procedure for the Vehicle Routing Problem **(Clarke and Wright, 1964)**, the NEH procedure for the Flow-Shop

Problem **(Nawaz et al., 1983)**, or the Path Scanning procedure for the Arc Routing Problem **(Golden et al., 1983)**. Typically, a 'priority' list of potential movements is traversed during the iterative process, i.e., at each iteration, the next constructive movement is selected from this list, which is sorted according to some criteria. The criteria employed to sort the list depends upon the specific heuristic being considered. Thus, any constructive heuristic could be seen as an iterative greedy procedure, which constructs a feasible 'good' solution to the problem at hand by selecting, at each iteration, the 'best' option from a list, sorted according to some logical criterion. Notice that this is a deterministic process, since once the criterion has been defined, it provides a unique order for the list of potential movements. Of course, if we randomize the order in which the elements of the list are selected, then a different output is likely to occur each time the entire procedure is executed. However, a uniform randomization of that list will basically destroy the logic behind the greedy behavior of the heuristic and, therefore, the output of the randomized algorithm is unlikely to provide a good solution –in fact, we could run the randomized algorithm thousands of times and it is likely that all the solutions generated would be significantly worse than the one provided by the original heuristic. To avoid losing the 'common sense' behind the heuristic, GRASP proposes to consider a restricted list of candidates –i.e. a sub-list including just some of the most promising movements, that is, the ones at the top of the list–, and then apply a uniform randomization in the order the elements of that restricted list are selected (**Figure 4**).

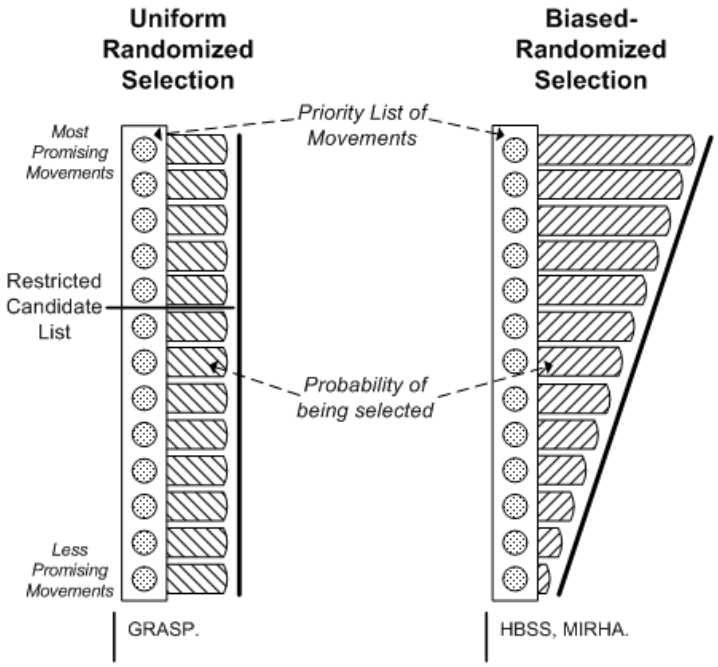

Figure 4. Uniform randomization vs. biased randomization

This way, a deterministic procedure is transformed into a randomized algorithm –which can be encapsulated into a multi-start process–, while most of the logic or common sense behind the original heuristic is still respected. The MIRHA approach goes one step further, and instead of restricting the list of candidates, it assigns different probabilities of being selected to each potential movement in the sorted list. In this way, the elements at the top of the list receive more probabilities of being selected than those at the bottom of the list, but potentially all elements could be selected. Notice that by doing so, we are not only avoiding the issue of selecting the proper size of the restricted list, but we also guarantee that the probabilities of being selected are always proportional to the position of each element in the list. Thus, it is possible to identify the following steps when transforming a classical heuristic into a probabilistic algorithm by means of biased randomization:

1. Given a COP, select a deterministic and constructive heuristic with the following characteristics –which most classical heuristics already satisfy: (a) it should be able to run quite fast –typically in less than a second– even for large-size problems –this is a critical requirement since the probabilistic algorithm relies in executing over and over a randomized version of the heuristic; (b) it should provide, by design, some stage able to be randomized –e.g. a priority list as the one described before; and (c) it should provide 'good' solutions which are not too far from the ones generated with more complex and time-consuming metaheuristics –e.g. average gap about 5-10%.
2. Once the base heuristic is selected, the new probabilistic algorithm should follow some kind of multi-start process –e.g. a pure multi-start or an iterated local search. At each round of this multi-start process, a new complete solution would be generated. For the construction of this solution, the base heuristic is randomized –e.g. its priority list is randomized– using a non-symmetric probability distribution. The specific distribution to employ will depend upon the specific COP being considered. Some candidate distributions to be considered are the geometric and a discrete version of the descendent triangular.
3. Optionally, a local search process can be added to the algorithm in order to improve the solution provided at each round of the multi-start process. This local search is COP-tailored, meaning that it will be different for each COP.

The approach described above is able to quickly generate several feasible solutions with different properties. Therefore, a list containing the top 'best-found' solutions – each of them having different characteristics– can be saved and considered by the decision maker.

2.2.1 Benefits of Biased-Randomization Approaches

The main motivation behind designing biased-randomized heuristics is to meet many of the desirable features of a metaheuristic as described by **Cordeau et al. (2002)**, i.e.: accuracy, speed, simplicity, and flexibility. Most of the metaheuristics in literature are measured against accuracy –the degree of departure of the obtained solution value from the optimal value–, and against speed –the computation time. However, there are two other important attributes to be considered in any optimization method: simplicity and flexibility. Simplicity is related to the number of parameters to be set and the facility of implementation. This is an important feature since the method can be applied to different instances than the ones tested without losing quality or performance and without the need of performing a long run test. Flexibility is related to the possibility of accommodating new side constraints and also with the adaptation to other similar problems.

Having in mind these measured attributes; we list the main benefits of biased-randomized heuristics over other related approaches:

⇒ They allow a simplification of the fine-tuning process, since typically the employed probability distributions require just one (e.g. the geometric) or zero parameters (e.g. the descendent triangular). This is not common in most current metaheuristic approaches, which usually employ many parameters and, therefore, require from difficult and time-expensive fine-tuning processes to adjust their associated values.

⇒ Being based on classical well-tested heuristics, they are relatively simple and easy to implement methods, which can be adapted to account for new constraints (flexibility). Thus, when facing a combinatorial-optimization problem with already existing heuristics, some of the most efficient of these heuristics can be selected and enhanced throughout biased randomization.

⇒ Using non-uniform (skewed) distributions rather than uniform ones, they offer a more natural and efficient way to select the next movement from the priority list, since biased randomization allows keeping the logic behind the heuristic by assigning more probabilities of being selected to those movements which better fulfill the heuristic. Notice that a uniform randomization does not respect the 'common sense' of the heuristic, since it assigns equal probabilities of being selected to all potential movements.

⇒ By combining randomization with a multi-start-like process, they promote diversification during the exploration of the solution space, i.e., the search is not restricted to just a reduced number of regions **(Talbi, 2009)**. Notice that these

two features –randomization and multi-start approach– help the algorithm to escape from local minimums and also increase diversification during the exploration of the solution space.

⇒ Likewise, the combination of randomization with a multi-start-like process promotes parallelization in an easy and natural way **(Juan et al., 2011c)**. Notice that the odds of finding 'good' solutions increase as more parallel runs of the algorithm are executed.

⇒ Finally, the biased-randomized heuristics can also be combined with other techniques, such as Monte Carlo simulation, in order to consider stochastic variants of COPs, as we discuss in later in this chapter.

2.2.2 Examples of Use

In general, probabilistic algorithms have been widely used to solve many combinatorial optimization problems such as, for example: Sequencing and Scheduling Problems **(Pinedo, 2008)**, Vehicle Routing Problems **(Laporte, 2009)**, or Quadratic and Assignment Problems **(Loiola et al., 2007)**. They have also been used to solve real combinatorial optimization problems that arise in different industrial sectors, e.g.: Transportation, Logistics, Manufacturing, Aeronautics, Telecommunication, Health, Electrical Power Systems, Biotechnology, etc.

As described in **Festa and Resende (2009b)**, GRASP algorithms have been applied to solve a wide set of problems like scheduling, routing, logic, partitioning, location, graph theory, assignment, manufacturing, transportation, telecommunications, biology and related fields, automatic drawing, power systems, and VLSI design. Regarding the use of biased/skewed randomization as proposed by the HBSS and MIRHA general schemes, **Juan et al. (2010)** proposed a specific implementation, called SR-GCWS, for solving the Capacitated Vehicle Routing Problem. The SR-GCWS algorithm combines a biased randomization process with the Clarke & Wright savings heuristic **(Clarke and Wright, 1964)**. A geometric distribution is used to randomize the constructive process while keeping the logic behind the heuristic. Other authors have also proposed the randomization of a classical heuristic for solving the Arc Routing Problem. For example, **Gomes and Selman (1997)** propose a randomized version of the Backtrack Search algorithm where randomization is added to break ties, and also a randomization of the Branch-and-Bound algorithm where randomization is added in the variable selection strategy by introducing noise in the ranking of the variables. However, in both cases the authors add uniformly distributed randomization to the base heuristic. Finally, **Juan et al. (2012)** propose the ILS-ESP algorithm for solving the Permutation Flow-Shop Problem. The ILS-ESP uses an Iterated Local

Search framework and combines the NEH heuristic **(Nawaz et al, 1983)** with a biased randomization process guided by a descending triangular distribution.

All in all, the proposed methodology can be used to improve the efficiency of most existing heuristics for solving combinatorial-optimization problems. This is done by transforming the greedy deterministic behavior of the heuristic into a probabilistic behavior which still follows the logic behind the heuristic but randomizes the construction process using a biased, non-uniform, distribution.

2.3 Simheuristics

There is an emerging interest on introducing randomization into combinatorial optimization problems as a way of describing new real problems in which most of the information and data cannot be known beforehand. This tendency can be observed in **Van Hentenryck and Bent (2010)**, which provides an interesting review of many traditional combinatorial problems with stochastic parameters. Thus, those authors studied Stochastic Scheduling, Stochastic Reservations and Stochastic Routing in order to make decision on-line, i.e., to re-optimize solutions when their initial conditions have changed and, therefore, are no longer optimal.

The Simheuristic approach **(Figure 5)** is a particular case of simulation-optimization which combines a heuristic/metaheuristic algorithm with simulation methodologies – e.g. Monte-Carlo, discrete-event, agent-based, etc… - in order to efficiently deal with the two components in a Stochastic Combinatorial Optimization Problem (SCOP) instance: the optimization nature of the problem and its stochastic nature. Examples of Simheuristics applications to different fields can be found in the optimization-simulation literature. For instance, **Juan et al. (2011b)** combined Monte Carlo Simulation (MCS) with routing metaheuristics in order to solve the Vehicle Routing Problem with Stochastic Demands (VRPSD); **Peruyero et al. (2011)** combined MCS with a scheduling metaheuristic for solving the permutation flow-shop problem with stochastic processing times; and **Caceres et al. (2012)** combined also MCS with a routing metaheuristic for solving the inventory routing problem with stock-outs and stochastic demands.

Typically, given a SCOP instance, a heuristic/metaheuristic algorithm is run in order to perform an oriented search inside the solution space. This iterative process aims at finding feasible solutions with the best possible value, which is expected to be near-optimal as well. During the iterative search process, the algorithm must deal with the stochastic nature of the SCOP instance. One natural way to do this is by taking

advantage of the capabilities simulation methods offer to manage randomness. Of course, other approaches can also be used instead of simulation –e.g. dynamic programming, fuzzy logic, etc. However, under the presence of historical data on stochastic behavior, simulation allows the development of both accurate and flexible models. Specifically, randomness can be modeled throughout a best-fit probability distribution –including parameterization– without any additional assumptions or constraints. Thus, simulation is usually integrated with the heuristic/meta-heuristic approach and it frequently provides dynamic feedback to the searching process in order to improve the final outcome. In some sense, simulation allows to extend already existing and highly efficient meta-heuristics –initially designed to cope with deterministic problems– so that they can also be employed to solve SCOPs.

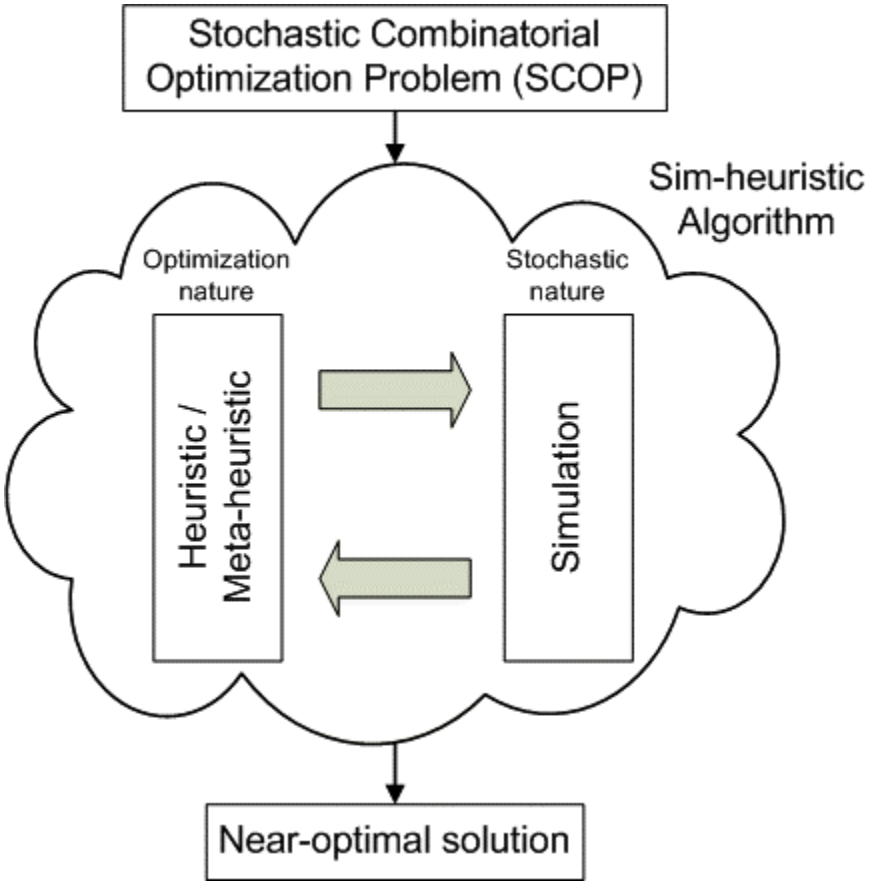

Figure 5. Overview scheme of the Simheuristic approach

Obviously, one major drawback of this approach is that the results are not expected to be optimal anymore, since Simheuristics are combining two approximate methodologies. Nevertheless, real-life problems are complex enough and usually NP-hard even in their deterministic versions. Therefore, Simheuristics constitute a quite interesting alternative for most practical purposes, since they represent relatively

simple and flexible methods which are able to provide near-optimal solutions to complex real-life problems in reasonable computing times.

2.3.1 An Integrated Simulation-based Framework

The solution proposed for the VRPSD in **Juan et al. (2011b)** was based on the same framework, which will be described in this section. The methodology is based in two facts: (a) the stochastic problem can be seen as a generalization of the deterministic problem where the random value has zero variance; and (b) efficient meta-heuristics already exist for the deterministic problems while the stochastic problems are an emerging field. Accordingly, the key idea behind this framework is to transform the stochastic instance into a new problem which consists of solving several conservative instances of the deterministic problem, each of the characterized by a specific risk of showing routing failures. The term conservative refers to the fact that only a certain percentage of the vehicle capacity is considered during the route design phase, so the rest of capacity is considered as safety stock. So, this remaining capacity of the vehicle will be available in case the actual demand of the route is greater than expected.

The methodology consists on the following steps (see **Figure 6**):

1. Consider a problem instance with a set of customers. Each customer has associated a stochastic demand characterized by its mean and probability distribution.
2. Compute the value of the safety stock given by a percentage of the vehicle capacity. This percentage will be a parameter defined for the algorithm.
3. Consider the deterministic instance of the problem, consisting on the same problem instance than the stochastic version, but where the demands are deterministic and equal to the mean, and the vehicle capacity is restricted to the value computed on step 2.
4. Solve previous instance using a biased randomized algorithm (e.g. SG-GCWS for CVRP or RandSHARP for CARP). This solution will be an aprioristic solution for the original problem instance. Furthermore, it will be a feasible solution as long as there are not any route failures.
5. Using the previous solution, estimate the expected cost due to possible failures on any route. This is done by using Monte Carlo simulation. To this end, random demands are generated and, whenever a route occurs a repair action is applied, registering the associated cost of this action in the total cost of the solution. The repair action consists of a round-trip from the depot to the failing customer so the vehicle capacity is reloaded. After iterating this process some thousands of times, a random sample of costs is obtained, and an estimate for

its expected value can be calculated. Then, the expected total costs due to possible route failures can be computed by addition of these variable costs and the costs of the deterministic solution obtained during the design phase.

6. Using the deterministic solution, obtain an estimate for the reliability of each route of the solution. In such context, the reliability index is defined as the

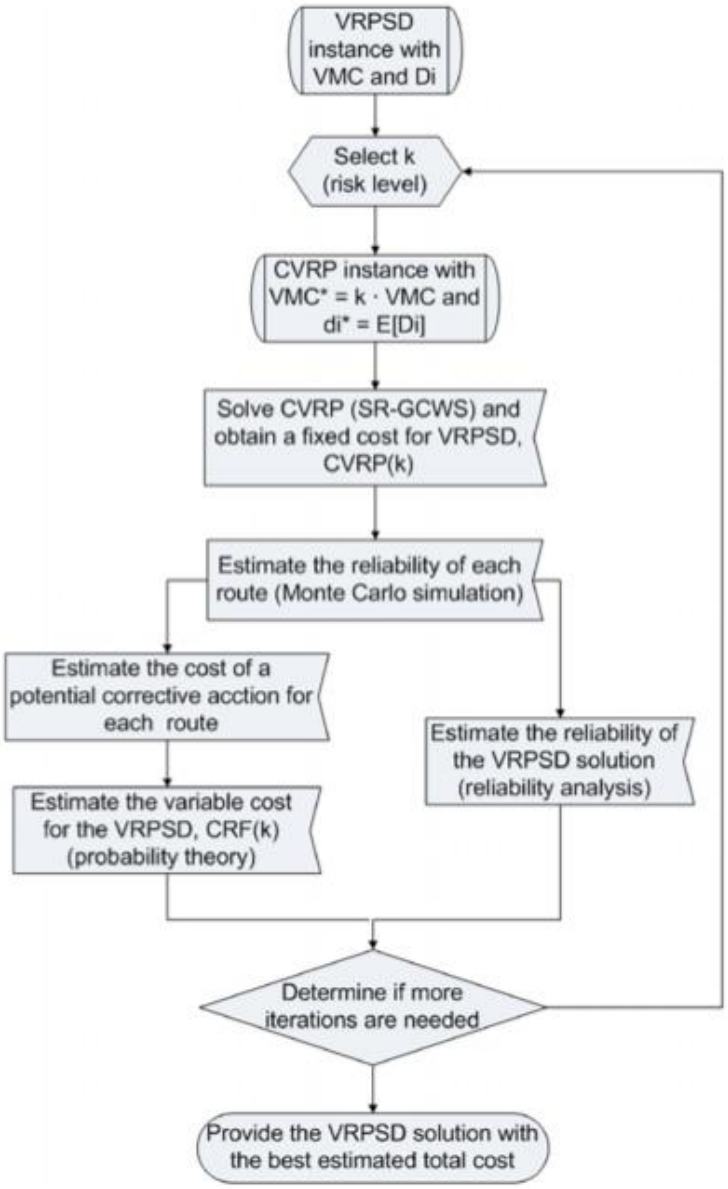

Figure 6. Flow diagram for the described methodology

probability that a route will not fail. This route reliability index is computed by direct Monte Carlo simulation using the probability distributions that model customer demands in each route. Remark that in each route, over-estimated demands could sometimes be compensated by under-estimated demands.

7. The reliability index for the total solution is computed as the multiplication of the value for each route in the solution. This value can be considered as a measure for the feasibility for the solution in the context of the stochastic problem.
8. Depending on the total costs and the reliability indices associated with the solutions already obtained, repeat the process from Step 1 with a different value of the value used for the safety stock.
9. Finally, the best solution found so far is returned as solution to the problem, as well as its corresponding properties such cost or reliability index.

2.3.2 Examples of Use

The Vehicle Routing Problem with Stochastic Demands (VRPSD) is a NP-hard problem in which a set of customers with random demands must be served by a fleet of homogeneous vehicles departing from a depot, which initially holds all available resources **(Novoa and Storer, 2009)**. Obviously, there are some tangible costs associated with the distribution of these resources from the depot to the customers. In particular, it is usual for the model to explicitly consider costs due to moving a vehicle from one node – customer or depot – to another. These costs are often related to the total distance traveled, but they can also include other factors such as number of vehicles employed, service times for each customer, etc. The classical goal here consists of determining the optimal solution (set of routes) that minimizes those tangible costs subject to the following constraints: (i) all routes begin and end at the depot; (ii) each vehicle has a maximum load capacity, which is considered to be the same for all vehicles; (iii) all (stochastic) customer's demands must be satisfied; (iv) each customer is supplied by a single vehicle; and (v) a vehicle cannot stop twice at the same customer without incurring in a penalty cost.

The random behavior of customers' demands could cause an expected feasible solution to become an unfeasible one if the final demand of any route exceeds the actual vehicle capacity. This situation is referred to as route failure, and when it happens some corrective actions must be introduced to obtain a new feasible solution. Thus, for example, after a route failure the associated vehicle might be forced to return to the depot in order to reload and resume the distribution at the last visited customer. As discussed in **Juan et al. (2011b)**, one possible methodology to deal with this problem is to design reliable solutions, i.e., solutions with a low probability of suffering route failures. This is basically attained by constructing routes in which the associated expected demand will be somewhat lower than the vehicle capacity. Particularly, the idea is to keep a certain amount of vehicle capacity surplus (safety stock) while designing the routes, so that if final routes' demands exceed their expected values up

to a certain limit, they can be satisfied without incurring in a route failure. Using safety stocks not only contributes to reduce variable costs due to route failures but, related to that, it also increases the reliability or robustness of the planned routes, i.e.: as safety stock levels increase, the probability of suffering a route failure diminishes. Notice, however, that employing safety stocks also increases fixed costs associated with aprioristic routing design, since more vehicles and more routes are needed when larger buffers are considered. Therefore, when minimizing the total expected cost a tradeoff exists between its two components, fixed costs and expected variable costs. Thus, the challenge relies on the selection of the appropriate buffer size.

Given a VRPSD instance, **Juan et al. (2011b)** consider different levels of this buffer size and then solve the resulting scenarios. This is performed by employing Monte Carlo simulation, which allows estimating the variable costs associated with each candidate solution. Thus, among the multiple solutions generated for each scenario, the ones with lowest total expected costs are stored as the best-found result associated with the corresponding safety-stocks level. Once the execution of the different scenarios ends, the corresponding solutions are compared to each other and the one with the lowest total expected costs is selected as the best-found routing plan.

2.4 Chapter Conclusions

In this chapter we have described the MIRHA framework and the Simheuristics methodology. We have analyzed some key aspects, benefits, and examples related to the combination of randomization strategies with classical heuristics as a natural way to develop probabilistic algorithms to solve combinatorial optimization problems. Both uniform as well as non-symmetric randomization strategies have been reviewed. In particular, we have discussed how the non-symmetric or biased approach constitutes an efficient way to randomize the priority list of a constructive heuristic without losing the logic behind it. Some examples of applications to several combinatorial optimization problems have also been exposed, including: vehicle routing problems and flow-shop problems. One of the main advantages of this type of probabilistic algorithms is their relative simplicity, since they are based in well-known heuristics and they do not incorporate many additional parameters. Moreover, these algorithms are flexible, quite efficient, and can be implemented and parallelized in a natural way, which make them an interesting alternative to more sophisticated metaheuristics in most practical applications.

3 Capacitated Arc Routing Problem

Parts of this chapter have been taken from the co-autored publications: **Gonzalez-Martin et al. (2011)** and **Gonzalez-Martin et al. (2012a)**.

The Capacitated Vehicle Routing Problem, or simply CVRP, is a well-known combinatorial optimization problem in which a fleet of homogeneous vehicles, initially located at a single depot with unlimited capacity, has to be routed so that all customers' demands are served at a minimum cost and without violating the loading capacity of each individual vehicle **(Toth and Vigo, 2002)**. In the CVRP, it is assumed that the road network is a complete graph containing nodes (vertices) representing the customers and the depot. Also, it is assumed that the customers' demands are located at these nodes. Being a *NP*-hard problem studied for several decades already, the CVRP is still attracting researchers' attention due to its potential applications both to real-life scenarios and to the development of new algorithms, optimization methods and metaheuristics. A related but different problem is the so-called CARP, in which the demands are located over the network edges or arcs (instead of at nodes), and in which the completeness of the road network is no longer guaranteed, i.e. not all pair of nodes are connected by an edge. The capacity restriction refers to the maximum load that each vehicle can carry. Sometimes other constraints also apply, e.g., maximum distance that can be covered by any single vehicle, the maximum time a vehicle can be operating, etc.

This chapter will introduce the first problem of study in this thesis, the CARP. The CARP, due to its characteristics, is also suitable for modeling certain Telecommunication problem, when compared to the CVRP. Having the demands located in the arcs of the graph instead of the nodes, for instance, we can think on them as telecommunications cables which should provide certain services to the users. Usually telecommunications users do not have a demand located on a single point, but a bandwidth required in the connection (arc) to the centralizer node.

3.1 Literature Review

Vehicle Routing Problems (VRPs) constitute a relevant topic for current researchers and practitioners. In fact, according to **Eksioglu et al. (2009)**, the number of related articles published in refereed journals has experienced an exponential growth in the last fifty years. However, although the existing literature on the CVRP field is quite extensive and includes a wide range of efficient approaches, this is not the case with the CARP. One of the basic references on the CARP is the book "*Arc Routing: Theory, Solutions and Applications*" **(Dror, 2000)**. This book, which contains twelve chapters devoted to different CARP-related topics, describes theoretical aspects associated with arc traversal, solution methodologies, and also a number of representative applications.

3.1.1 Approaches for Solving the CARP

Since the CARP was first described in by **Golden and Wong (1981)**, a wide range of both exact and approximate methods have been suggested for solving it. For extensive surveys on the arc routing field, including CARP and some of its variations, the reader is referred to **Assad and Golden (1995)** or **Wøhlk (2008)**. Among the exact approaches, Branch and Bound and Cutting Plane are the most common methods employed in the CARP literature. Branch-and-bound methods for the CARP were proposed by **Hirabayashi et al. (1992)** and **Kiuchi et al. (1995)**. In Cutting Plane, a Linear Programming relaxation of the problem is solved. **Belenguer and Benavent (2003)** presented a Cutting Plane algorithm for the CARP, which is partly based on several classes of valid inequalities presented earlier by the same authors in **Belenguer and Benavent (1998)**. Using their algorithm, these authors are able to reach the best existing lower bounds for all test instances, even improving the existing lower bounds for several instances. Branch-and-Cut-and-Price based algorithms were suggested by **Letchford and Oukil (2009)**. However, algorithms that guarantee finding the optimal solution can be used only for relative small and medium-small instances.

A different strategy for solving the CARP is based on transforming the problem into a Capacitated Vehicle Routing Problem (CVRP), which is in turn solved by state-of-the-art CVRP algorithms, as described in **Eglese and Letchford (2000)**, **Longo et al. (2006)**, and **Baldacci and Maniezzo (2006)**. Results reported in those works are highly competitive compared to the previously existing ones. Likewise, methods providing lower bounds for the CARP were put forward by **Golden et al. (1983)**, **Pearn et al. (1987)**, **Pearn (1988)**, **Win (1988)** or **Amberg and Voß (2002)**.

Due to the complexity of the problem, several heuristics have been also developed for the CARP. Among them it is possible to point out those of **Golden et al.**

(1983), **Chapleau et al. (1984)**, **Ulusoy (1985)**, or **Pearn 1989 and 1991**. Many other heuristics are described by **Assad and Golden (1985)**, **Eiselt et al. (1995)**, and **Wøhlk (2005)**. It should be mentioned that most of these approaches are problem-specific heuristics and their performance is generally 10 to 40 percent above the optimal solution according to some studies **(Wøhlk, 2008)**. More recently, different metaheuristics have been proposed according. Among these, a considerable number of Tabu Search (TS) algorithms have been designed to solve the CARP. One of the first ones, called CARPET, was proposed by **Hertz et al. (2000)**. In this work, unfeasible solutions are allowed but are also penalized. CARPET is one of the most efficient metaheuristics published so far for the CARP. Other TS-based approaches were developed by **Amberg et al. (2000)** for the Multi Depot version of the CARP, and by **Greistorger (2003)**, who combined TS with Scatter Search to construct a Tabu Scatter Search for the CARP. A deterministic TS algorithm has been suggested by **Brandâo and Eglese (2008)**. Their TS penalizes infeasible solutions in the objective function and alternates between different neighborhood structures. A hybrid Genetic Algorithm and a Memetic Algorithm were presented by **Lacomme et al. (2001)** and **Lacomme et al. (2004a)**, respectively. These algorithms are currently among the best performing ones. Other approach methods for the CARP are related with Simulated Annealing (SA) algorithms. For instance, **Li (1992)**, in his Ph.D. thesis, applied this technique and TS to a road gritting problem. Also, **Eglese (1994)** designed a SA approach for a multi depot gritting problem with side constraints. Finally, **Wøhlk (2005)** suggested a SA algorithm for the classical CARP, where the order of the edges on a giant tour is changed during the algorithm, and at each step the optimal partitioning of the tour is computed. Ant Colony Systems were proposed by **Lacomme et al. (2004b)** and **Doerner et al. (2003)**. In the former, the authors reported results competitive to the best algorithms with respect to solution quality, but employing longer computation times. In the latter, the authors reported limited success. A Guided Local Search algorithm has been presented for the CARP by **Beullens et al. (2003)**. In this work, the distance of each arc is penalized according to some function, which is adjusted throughout the algorithm. Computational experiments show that this approach is a promising one. Additionally, it is worthy to mention the Variable Neighborhood Descent algorithm proposed by **Hertz and Mittazl (2001)**, since it also reports competitive results.

3.1.2 Problem variations

There are multiple variations of the CARP existing in the literature, each of them reflecting different real-life scenarios. Among the most popular variations, it is possible to cite the following ones:

- ⇒ The CARP defined on a directed graph or Directed Capacitated Arc Routing Problem **(Maniezzo and Roffilli, 2008)**.
- ⇒ The CARP defined on mixed graphs or Mixed Capacitated Arc Routing Problem **(Belenguer et al. 2006)**.
- ⇒ The CARP with objective functions other than one based on total traveling costs, i.e., minimizing the total number of vehicles, minimizing the length of the longest tour, etc. For instance, **Ulusoy (1985)** considers minimizing the traveling costs plus some fixed costs associated with the use of different types of vehicles.
- ⇒ The CARP with multi-objective functions. For instance, **Shang et al. (2014)** consider a multi objective CARP that can be viewed as a mix between the CARP and the Min-Max K-Chinese Postman Problem. The Min-Max K-Chinese Postman Problem **(Frederickson, 1978)** is a CARP-like problem where the vehicle loading capacity is infinite.
- ⇒ The CARP with Time Windows, considered by **Reghioui et al. (2007)** and by **Wøhlk (2007)**.
- ⇒ The CARP with time-dependent service costs, as considered by **Tagmouti et al. (2011)**.
- ⇒ The CARP with multiple depots, proposed by **Cattrysse et al. (2002)**. There is also a variant with mobile depots, proposed by **Del Pia et al. (2006)**.
- ⇒ The Periodic CARP, studied by **Chu et al. (2003)** and by **Lacomme et al. (2005)**.
- ⇒ The CARP with profits, in which the objective is to maximize the profit collected from a set of potential customers as described in **Zachariadis and Kiranoudis (2011)**.
- ⇒ The CARP with Stochastic Demands (ARPSD), where the customer's demands are not known beforehand **(Gonzalez-Martin et al., 2014b)**.

3.1.3 Applications

Routing problems have been extensively studied due to their wide number of real-life applications, which usually involve very large costs associated with the delivery of goods demanded by customers. The CARP is a delivery problem with several practical applications. Some classical examples of arc routing problems where the service

activity requires finding *minimum-cost* routes in a transportation network can be found in: refuse collection, snow removal, inspection of distributing systems, routing of street sweepers, routing of electric meter readers, school bus routing, etc.

For example, **Beltrami and Bodin (1974)** describe a computer system for garbage collection planning that was applied in New York city; **Eglese and Li (1992)** study the problem of spraying roads with salt to prevent ice in the Lancashire County Council; **Bodin and Kursh (1979)** develop a computer assisted system for the routing and scheduling of street sweepers; **Stern and Dror (1979)** describe an arc routing problem associated with electric meter reading in the city of Beersheba (Israel). In most applications, several vehicles are involved and there may be a number of restrictions on capacity, travel distance, or travel times, among others.

More recently, we can find applications of the CARP to the Telecommunications field. One example, it is useful for the planning of the placement of optical cable network. In this problem, the objective is to decide for which streets the cable should be installed in order to provide access to all the customers which are distributed along the streets (and thus we consider that demand is located in the arcs).

3.2 Problem Description

3.2.1 Basic description

The CARP is a combinatorial optimization problem that can be informally described as follows **(Golden and Wong (1981))**: Let $G = (V, E, C, Q)$ be an undirected graph, where:

i. *V* is a set of nodes, including the one representing the depot or distribution center
ii. *E* is a set of edges or arcs connecting some nodes in *V*.
iii. *C* is a costs matrix representing the positive costs of moving from on node to another – these costs are usually based on distances or travel times between a pair of nodes.
iv. *Q* is a demands vector representing the non-negative demands associated with each arc.

Consider also a set of identical vehicles (homogeneous fleet), each of them with a maximum loading capacity $W \gg \max\{q_{ij} \in Q\}$. Under these circumstances, the usual goal is to find a set of vehicle routes that minimizes the total delivering costs while satisfying the following constraints:

i. Every route starts and ends at the depot (roundtrips).
ii. All arcs demands are satisfied.
iii. Each arc with a positive demand is served by exactly one vehicle. Notice, however, that an arc can be traversed more than once by the same or different vehicles.
iv. The total demand to be served in any route does not exceed the vehicle loading capacity *W*.

3.2.2 Mathematical model

Different mathematical formulations have been proposed for the CARP. The first ILP formulation was the one proposed by **Golden and Wong (1981)**, and it is based on directed variables even though the formulated problem is undirected. A formulation using undirected variables is presented by **Belenguer and Benavent (1992)**. In his Ph.D. dissertation, Letchford gives several ILP formulations of the CARP, and derives additional valid inequalities and separation algorithms for the problem. Other mathematical formulations are due to **Welz (1994)** and **Eglese and Letchford (2000)**, all of them for the undirected case. For describing the model, we use an integer programming formulation adapted from **Golden and Wong (1981)** to formally described the CARP. Although we are not explicitly using it in our solving approach based on MIRHA, analyzing this model contributes to a better understanding of the complexity, as well as some details of the problem.

The CARP is defined over a non-complete graph. Let $G = (V, E)$ be an undirected graph, where $V = \{0,1,2,\dots,n\}$ represents the set of *n+1* nodes (node 0 being the depot), and where $E \subseteq \{e_{ij} | i, j \in V, i \neq j\}$ represents the set of *m* arcs connecting some (not necessarily all) of the nodes. Associated with each edge $e_{ij} \in E$ there is a symmetric cost $c_{ij} = c_{ji} > 0$, as well as a non-negative demand $q_{ij} > 0$. Edges with strictly positive demand, $R = \{e_{ij} \in E | q_{ij} > 0\}$, are called required arcs as they must be traversed at least once in order to be served. Notice also that non-required arcs (those with zero demand) do not need to be traversed, although they may be traversed one or more times.

A fleet of *K* identical vehicles (or postmen in other contexts), each of them with a limited capacity *W*, are available and based on the depot node. We assume that $W \gg \max\{q_{ij} \in Q, \forall i, j \in V\}$. Let *N(i)* denote the subset of nodes which are adjacent (i.e. directly connected) to node *i*. Additionally, two set of variables are introduced:

i. x_{ij}^k is 1 if vehicle *k* traverses edge e_{ij} from node *i* to node *j*, and 0 otherwise.

ii. l_{ij}^k is 1 if vehicle *k* delivers edge e_{ij} while traversing it from node *i* to node *j*, and 0 otherwise.

The formulation is as follows:

$$min \sum_{k=1}^{K} \sum_{e_{ij} \in E} c_{ij} x_{ij}^k \tag{1}$$

Subject to:

$$\sum_{j \in N(i)} (x_{ij}^k - x_{ji}^k) = 0, \forall i \in V \backslash \{0\}, \forall k \in \{1,2,\dots,K\} \tag{2}$$

$$x_{ij}^k \geq l_{ij}^k, \forall e_{ij} \in R, \forall k \in \{1,2,\dots,K\} \tag{3}$$

$$\sum_{k=1}^{K} (l_{ij}^k + l_{ji}^k) = 1, \forall e_{ij} \in R \tag{4}$$

$$\sum_{i=0}^{n} \sum_{j=0}^{n} l_{ij}^k \cdot q_{ij} \leq W, \forall k \in \{1,2,\dots,K\} \tag{5}$$

$$\left(\sum_{\forall i \neq j, (i,j) \in SxS} x_{ij}^k \right) - |V|^2 \cdot y_S^k < |S|, \forall S \subset V \backslash \{0\} \ni |S| \geq 1, \forall k \in \{1,2,\dots,K\} \tag{6}$$

$$\sum_{\forall i \neq j, (i,j) \in SxS} x_{ij}^k + u_S^k \geq 1, \forall S \subset V \backslash \{0\} \ni |S| \geq 1, \forall k \in \{1,2,\dots,K\} \tag{7}$$

$$u_S^k + y_S^k \leq 1; u_S^k, y_S^k \in \{0,1\}, \forall k \in \{1,2,\dots,K\} \tag{8}$$

$$x_{ij}^k, l_{ij}^k \in \{0,1\}, \forall i \neq j, \forall k \in \{1,2,\dots,K\} \tag{9}$$

Constraint (2) ensures that, for each vehicle route *k* and node *I*, the vehicle route *k* returns to the same node *i*. Constraint (3) states that every edge e_{ij}, the value of x_{ij}^k is always greater or equal than l_{ij}^k, which ensures that every serviced edge is a traversed edge, so it is not possible to serve an edge without traversing it. Constraint (4) forces all required arcs to be serviced once and only once, as it ensures that the sum of the servicing variable x_{ij}^k for all routes is always equal to one. The explicit vehicle capacity constraint (5) applies to each vehicle route *k*, and states that the total demand served by a vehicle servicing route *k* cannot surpass the vehicle capacity *W*. This constraint is the one which makes the CARP a capacitated problem. Constraint (6) states that for any route *k* and subset of nodes *S*, the sum of all edges connecting any pair of node in *S* which are traversed by route *k*, minus the square of the total number of nodes

multiplied by y_S^k is always less than the number of nodes in *S*. Constraint (7) forces that, for any vehicle route *k* and any subset of nodes *S*, the sum of all edges connecting a node in *S* and a node outside *S* which are traversed by a given route *k*, plus u_S^k is always greater or equal than 0. This means that there always is, at least, a traversed arc to go outside *S*. The binary variables y_S^k and u_S^k are used here as auxiliary variables (the reader is referred to **Golden and Wong, 1981** for more details). All in all, the group of constraints (6)-(8) removes the possibility of disconnected sub-tours but allows tours that include two or more closed cycles. It must be noted that with this group of constraints the number of restrictions to consider grows exponentially as they are considering any subset of nodes *S*.

3.3 Classical heuristics for the CARP

As described in **Chapter 2**, the MIRHA framework requires a problem specific with the characteristic of being simple and fast to execute in order to be able to define randomized algorithms. Among all the classical heuristics existing in the literature, we have selected two of them: the Path Scanning Heuristic (PSH) for the CARP and the Clarke and Wright Savings (CWS) heuristic for the CVRP. It is worth to mention that the second one is an heuristic for the CVRP problem. So we will define an adaption of the heuristic for the ARC routing problem, the Savings Heuristic for the Arc Routing Problem (SHARP), which is also one of the original contributions of this thesis.

3.3.1 The Path Scanning heuristic for the CARP

The PSH is a classical heuristic proposed by **Golden et al. (1983)** for solving the CARP. As a constructive heuristic, it can be used as a base heuristic for defining a MIRHA based algorithm. The PSH is a simple and fast method for solving the CARP by building one route at a time. Its main idea is to construct five complete solutions, every one of them following a different optimization criterion. The final solution of the algorithm is the best – in terms of total costs – of the five solutions obtained.

The way every route is constructed is not clearly defined in the original paper, so it allows different interpretations when trying to implement it. Following **Belenguer et al. (2006)** the approach we use in this chapter is to extend the current route by selecting only adjacent arcs with not served demand, choosing only the arc which best accomplishes the given criteria. Assuming that the vehicle is currently located at node *I*, then an adjacent arc e_{ij} connecting nodes *i* and *j* must be selected such that *eij* meets one of the following criteria:

1. Minimizes the cost per unit demand, i.e. $\min\{c_{ij}/q_{ij}\}$.

2. Maximizes the cost per unit demand, i.e. $\min\{q_{ij}/c_{ij}\}$.
3. Minimizes the distance from node *j* back to the depot.
4. Maximizes the distance from node *j* back to the depot.
5. If the vehicle is less than half-full, it minimizes the distance from node *j* back to the depot, otherwise it maximizes the distance.

In case all adjacent arcs have been already served, then the closest not served arc – in terms of the shortest path distance – is selected. If there is more than one arc at the same minimum distance, then the best arc accomplishing the current optimization criteria is selected. Finally, once the vehicle capacity is exhausted, the current route is closed by returning the vehicle to the depot through the shortest path. The original algorithm does not state how the shortest path is computed. In our approach an implementation of the Dijkstra's algorithm **(Cormen et al., 2009)** is used.

Several modifications of the heuristic can be found on the literature. **Santos et al. (2009)** propose a new heuristic which is similar to the one by **Belenguer et al. (2006)**. The latter one randomly selects tied edges and solves each problem *k* times. Pearn et al. (1989) basically uses the same algorithm as the original PSH, but instead of applying each of the five next-step criterion in a separate way, a time-based loop is run. In every loop iteration, all five criteria are randomly combined together, i.e.: at each step during the constructive phase, a criterion is randomly selected following either a uniform or a weighted distribution.

3.3.2 The SHARP heuristic for the CARP

The CWS heuristic is probably the most cited heuristic in the CVRP arena. The CWS is an iterative method that starts out by considering an initial dummy solution in which each non-depot node (customer) is served by a dedicated vehicle. Next, the method initiates an iterative process for merging some of the routes in the initial solution. Merging routes can improve the expensive initial solution so that a unique vehicle serves the nodes of the merged route. The merging criterion is based upon the concept of savings. Roughly speaking, given a pair of nodes to be served, a savings value can be assigned to the edge connecting these two nodes. This savings value is given by the reduction in the total cost function due to serving both nodes with the same vehicle instead of using a dedicated vehicle to serve each node as proposed in the initial dummy solution. This way, the algorithm constructs a list of savings, one for each possible edge connecting to demanding nodes. At each iteration of the merging process, the edge with the largest possible savings is selected from the list as far as the following conditions are satisfied: (a) the nodes defining the edge are adjacent

to the depot; and (b) the two corresponding routes can be feasibly merged, that is, the vehicle capacity is not exceeded after the merging.

Since the CWS is considered by many authors as the best single heuristic for solving the CVRP, we decided to develop a new constructive heuristic for the CARP, based on the savings concept of the CWS. The savings-based approach, while extensively used in the CVRP, has received little attention in the CARP. Only recently, an adaptation version of the CWS has been used by **Tagmouti et al. (2007)**. However, these savings-based versions are only used to provide an initial starting point of a complex metaheuristic, and no details are given on how the CWS has been adapted to be used in the CARP. Since this adaptation process is not trivial and can be done in very different ways, this is a clear gap in the current ARP literature.

```
procedure SHARP(nodes, edges, vCap)
01    for {each pair of nodes iN,jN in nodes} do
02      sp ← calcShortestPath(iN,jN,edges)
03      dist ← calcDistance(iN,jN,sp)
04      spMatrix ← addPath(iN,jN,sp)
05      dMatrix ← addDistance(iN,jN,dist)
06    end for
07    rE ← selectRequiredEdges(edges)
08    rN ← selectRequiredNodes(rE)
09    currentSol ← buildDummySol(rE)
10    savings ← calcSavings(rN,dMatrix)
11    savings ← sortList(savings)
12    while {savings is not empty} do
13      edge ← selectEdgeAtTop(savings)
14      iN ← selectInitialNode(edge)
15      jN ← selectEndNode(edge)
16      for {each route iR crossing iN} do
16        for {each route jR crossing jN} do
17          if {isMegePossible(iR,jR,vCap)} then
18            newRoute ← mergeRoutes(iR,jR)
19            currentSol ← deleteRoutes(iR,jR)
20            currentSol ← addRoute(newRoute)
21            exit the for loops
22          end if
23        end for
24      end for
25    end while
26    for {each route iR in currentSol} do
27      iR ← completeRoute(iR,spMatrix)
28    end for
29    return avgVarCosts and solReliability
end procedure
```

Figure 7. Pseudo-code of the SHARP heuristic

The pseudo-code of SHARP (see **Figure 7**) describes how we developed a new heuristic for the CARP. First, we use the Floyd–Warshall algorithm **(Cormen et al.,**

2009) to compute the shortest paths $\delta(i, j)$ for all pairs of nodes (i, j) in the graph or road network (lines 1–6). This allows us to treat the graph as if it was a complete one. Having a virtually complete graph, we can now calculate the savings associated to each arc (line 10) – either if it is real or virtual – in a similar way as savings are computed in the CWS heuristic for the CVRP, i.e. $s(i, j) = \delta(depot, i)+\delta(j, depot)-\delta(i, j)$, for each pair of nodes (i, j) and the depot node. However, in the CARP case, we will only calculate savings between nodes that lie on required arcs (line 8), i.e. on edges with demand greater than zero. The edges are then sorted in a list according to their associated savings value (line 8). We now create a dummy solution by assigning a vehicle (route) to serve each required arc (line 9). At this stage, we keep track only of the required edges (and their orientation) in our routes, as the complete final route is reconstructed at the end. Additionally, at each node we keep track of routes for which that particular node is at the very start/end of the route (i.e., is an exterior node). We then iterate over the list of savings and look at each node in the selected arc to see which routes (if any) have that node as an exterior node, attempting to merge these routes if possible, i.e. as far as the capacity constraint is not violated (lines 12–26). This is done by iterating first over every route in node *i*, then over every route in node *j*, and finally checking if both routes can be merged (lines 16–18). If they can be merged, they are merged and removed from *i*'s and *j*'s routes lists (lines 19–22), and we proceed attempting other routes pairs until no more merges can be made. Finally, once we have run out of edges in the savings list, we will have a set of routes composed only of a set of sorted and oriented edges with demand greater than zero. We now reconstruct the final solution by computing the shortest path between the edges in the route using the all-pairs shortest path matrix we generated at the beginning of the procedure (lines 27–29).

3.4 Randomized algorithms for the CARP

3.4.1 The RandPSH algorithm

Reghioui et al. (2007) proposed a randomization of the PSH. Their randomization process is quite similar to the one we propose in this chapter. They designed a GRASP approach based on the PSH. Thus, their algorithm includes two levels of biased randomization: the first random process is associated with choosing the criterion among the five Path Scanning criteria described before; the second random process relates to choosing the next edge to be used. Moreover, they also consider twelve additional criteria, which make the resulting algorithm more efficient.

Our RandPSH algorithm also introduces two different randomization processes into the original heuristic:

1. When constructing a new solution, the optimization criterion used to select the next edge is not deterministic but probabilistic: a criterion is randomly selected, using a uniform probability distribution in a similar way as proposed in **Pearn et al. (1989)**.
2. When selecting the next edge during the solution construction process, a geometric distribution is used to randomly select the edge from the sorted edges list.

3.4.2 The RandSHARP algorithm

Our savings-based heuristic for the CARP, the SHARP, has been integrated into a schema based on the MIRHA framework for defining a randomized algorithm, the RandSHARP. This randomized algorithm will use the solution produced by the savings-based heuristic as an initial solution. Then, it will iteratively generate new randomized solutions by introducing a probabilistic criterion when selecting edges from the savings list. In this case, we have selected the geometric distribution to implement this probabilistic criterion. Work done by **Juan et al. (2010)** suggests that a geometric distribution with any parameter randomly selected between 0.10 and 0.25 provides acceptable results for the CVRP – notice that no time-costly fine-tuning process is needed here. Therefore, we have used these values as a reference while solving the CARP.

3.5 Results

To evaluate the performance of the proposed heuristics and randomized algorithms, they have all been implemented as computer program. Java SE6 was used here instead of C or C++ for several reasons:

i. Being an object-oriented programming language with advanced memory management features (such as the garbage collector) and with readily-available data structures, it allows a somewhat faster development of algorithmic software.
ii. It offers immediate portability to different platforms, i.e., the same Java application can be run over most operating systems.
iii. It offers better replicability and duplicability than other languages.

However, the downside of using Java instead of C or C++ is probably a reduction on code execution performance, mainly due to the fact that Java is not a compiled

language and to the lack of pointer-based optimization. To perform the tests, a standard personal computer was used with a an Intel® Core2® Quad CPU Q9300 @2.50GHz and 8 GB RAM running the Windows® 7 Pro operating system. The experiments were run with four different benchmarks obtained from **Belenguer (2014)** (see **Table 2**):

⇒ The *egl* dataset **(Eglese and Li, 1992)** consisting of 24 instances which are derived from real world data. This data refers to winter gritting in the county of Lancashire, United Kingdom. These instances contain both required and not required arcs, so there are also arcs without associated demand. The instances are grouped in two different network configurations with very low arc density. The density of arcs is computed as the percentage of arcs of the complete graph which conform the network of the given instance.

⇒ The *gdb* dataset **(Golden et al., 1983)** consisting of 23 instances of small and medium-size – between 10 and 50 arcs – with a mixture of dense and sparse graph networks. The instances are characterized by their number of nodes, number of arcs, number of required arcs, and the density of arcs on the network. These instances are artificially generated and contain only arcs with associated demand.

⇒ The *kshs* dataset **(Kiuchi et al., 1995)** consisting of 6 instances of small size and a medium- to high-density of arcs. They are artificially generated and all the instances have the same number of arcs. All of the arcs contained on them are required as they have an associated demand.

⇒ The *val* dataset **(Belenguer and Benavent, 2003)** consisting of 34 instances, modelled on 10 sparse graph networks, with varying vehicle capacities for each network. Thus, the same instance valX is to be solved with different capacity constraints, obtaining the variants valXY which conform the final instance. These instances include only required arcs and are networks with low arc density (around 10%).

Set	Instances	Nodes	Arcs	Density
egl	24	109	144	2.65%
gdb	23	12	29	53.77%
kshs	6	8	15	55.95%
val	34	36	63	10.59%

Table 2. Average characteristics of problem datasets

3.5.1 Comparison using the BEST10 and AVG10 metrics

For each one of the aforementioned set of instances, and for each approach – either heuristic or randomized algorithm – we designed and performed extensive tests using the same machine, same language program, same execution time, and same programmer. In particular, for each of the randomized algorithms and for each tested instance, 10 independent iterations (replicas) were run. Each replica was run for a maximum time of 180 s. Then, for each set of replicas, the best experimental solution found (BEST10) as well as the average value of the different replicas (AVG10) were registered. Also, the best-known solution (BKS) associated with each instance was obtained from **Santos et al., 2009**. **Table 3 to Table 6** show detailed results for each instance (87 instances in total), including the gap of each approach with respect to the BKS.

		Average gap w.r.t. BKS				Average gap w.r.t. BKS in MaxTime = 180s							
		Heuristic				BEST10				AVG10			
		PSH		SHARP		RandPSH		RandSHARP		RandPSH		RandSHARP	
Set	**BKS**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**
gdb1	316	316	0.00%	323	2.22%	316	0.00%	316	0.00%	316	0.00%	316	0.00%
gdb2	339	367	8.26%	360	6.19%	339	0.00%	339	0.00%	339	0.00%	339	0.00%
gdb3	275	289	5.09%	296	7.64%	275	0.00%	275	0.00%	275	0.00%	275	0.00%
gdb4	287	320	11.50%	320	11.50%	287	0.00%	287	0.00%	287	0.00%	287	0.00%
gdb5	377	439	16.45%	409	8.49%	383	1.59%	377	0.00%	383	1.59%	377	0.00%
gdb6	298	330	10.74%	338	13.42%	298	0.00%	298	0.00%	298	0.00%	298	0.00%
gdb7	325	330	1.54%	359	10.46%	325	0.00%	325	0.00%	325	0.00%	325	0.00%
gdb8	348	408	17.24%	392	12.64%	350	0.57%	350	0.57%	354	1.84%	354	1.58%
gdb9	303	364	20.13%	333	9.90%	313	3.30%	313	3.30%	317	4.62%	315	3.96%
gdb10	275	284	3.27%	303	10.18%	275	0.00%	275	0.00%	275	0.00%	275	0.00%
gdb11	395	424	7.34%	435	10.13%	395	0.00%	395	0.00%	395	0.00%	396	0.30%
gdb12	458	560	22.27%	539	17.69%	490	6.99%	468	2.18%	490	6.99%	470	2.58%
gdb13	536	548	2.24%	556	3.73%	536	0.00%	536	0.00%	536	0.00%	537	0.15%
gdb14	100	104	4.00%	104	4.00%	100	0.00%	100	0.00%	100	0.00%	101	0.81%
gdb15	58	60	3.45%	58	0.00%	58	0.00%	58	0.00%	58	0.00%	58	0.00%
gdb16	127	131	3.15%	133	4.72%	127	0.00%	127	0.00%	127	0.00%	127	0.00%
gdb17	91	91	0.00%	93	2.20%	91	0.00%	91	0.00%	91	0.00%	91	0.00%
gdb18	164	168	2.44%	185	12.80%	164	0.00%	164	0.00%	164	0.00%	166	1.11%
gdb19	55	57	3.64%	63	14.55%	55	0.00%	55	0.00%	55	0.00%	55	0.00%
gdb20	121	127	4.96%	125	3.31%	121	0.00%	121	0.00%	121	0.00%	121	0.00%
gdb21	156	168	7.69%	162	3.85%	156	0.00%	156	0.00%	156	0.00%	157	0.77%
gdb22	200	204	2.00%	205	2.50%	200	0.00%	200	0.00%	200	0.00%	200	0.00%
gdb23	233	246	5.58%	237	1.72%	233	0.00%	233	0.00%	234	0.30%	235	0.60%
Avg.			*7.09%*		*7.56%*		*0.54%*		*0.26%*		*0.67%*		*0.53%*

Table 3. Experimental results for *gdb* instances.

		Average gap w.r.t. BKS				Average gap w.r.t. BKS in MaxTime = 180s							
		Heuristic				BEST10				AVG10			
		PSH		SHARP		RandPSH		RandSHARP		RandPSH		RandSHARP	
Set	**BKS**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**
kshs1	14661	15164	3.43%	16825	14.76%	14661	0.00%	14661	0.00%	14661	0.00%	14661	0.00%
kshs2	9863	9953	0.91%	9953	0.91%	9863	0.00%	9863	0.00%	9863	0.00%	9863	0.00%
kshs3	9320	9757	4.69%	9784	4.98%	9320	0.00%	9666	3.71%	9320	0.00%	9666	3.71%
kshs4	11498	14408	25.31%	13636	18.59%	12076	5.03%	11498	0.00%	12076	5.03%	11498	0.00%
kshs5	10957	12721	16.10%	12095	10.39%	10957	0.00%	10957	0.00%	10957	0.00%	10957	0.00%
kshs6	10197	11091	8.77%	11177	9.61%	10197	0.00%	10197	0.00%	10197	0.00%	10197	0.00%
Avg.			*9.87%*		*9.87%*		*0.84%*		*0.62%*		*0.84%*		*0.62%*

Table 4. Experimental results for *kshs* instances.

		Average gap w.r.t. BKS				Average gap w.r.t. BKS in MaxTime = 180s							
		Heuristic				BEST10				AVG10			
		PSH		SHARP		RandPSH		RandSHARP		RandPSH		RandSHARP	
Set	**BKS**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**
e1-A	3548	4263	20.15%	3826	7.84%	3921	10.51%	3548	0.00%	3923	10.56%	3548	0.00%
e1-B	4498	5499	22.25%	4769	6.02%	4956	10.18%	4498	0.00%	4956	10.18%	4512	0.32%
e1-C	5595	7044	25.90%	5939	6.15%	6596	17.89%	5632	0.66%	6596	17.89%	5632	0.67%
e2-A	5018	6513	29.79%	5288	5.38%	5507	9.74%	5022	0.08%	5546	10.51%	5043	0.50%
e2-B	6317	7541	19.38%	6710	6.22%	7146	13.12%	6344	0.43%	7160	13.34%	6366	0.78%
e2-C	8335	10537	26.42%	8945	7.32%	9267	11.18%	8477	1.70%	9346	12.13%	8518	2.20%
e3-A	5898	7214	22.31%	6291	6.66%	6401	8.53%	5924	0.44%	6437	9.13%	5941	0.73%
e3-B	7775	9526	22.52%	8228	5.83%	8554	10.02%	7847	0.93%	8606	10.65%	7868	1.20%
e3-C	10292	13178	28.04%	11026	7.13%	11408	10.84%	10386	0.91%	11508	11.81%	10494	1.96%
e4-A	6444	7979	23.82%	6835	6.07%	6913	7.28%	6504	0.93%	7027	9.04%	6530	1.33%
e4-B	8983	10650	18.56%	9825	9.37%	9743	8.46%	9120	1.53%	9841	9.55%	9185	2.25%
e4-C	11596	14157	22.09%	12810	10.47%	12824	10.59%	11886	2.50%	13073	12.73%	11907	2.68%
s1-A	5018	6382	27.18%	5255	4.72%	5912	17.82%	5018	0.00%	5912	17.82%	5025	0.14%
s1-B	6388	8372	31.06%	6666	4.35%	8110	26.96%	6435	0.74%	8110	26.96%	6450	0.97%
s1-C	8518	10259	20.44%	8626	1.27%	9442	10.85%	8518	0.00%	9468	11.16%	8529	0.13%
s2-A	9844	12344	25.40%	10322	4.86%	11115	12.91%	10076	2.36%	11319	14.98%	10140	3.01%
s2-B	13100	16653	27.12%	13880	5.95%	14550	11.07%	13356	1.95%	14697	12.19%	13457	2.72%
s2-C	16425	20665	25.81%	17399	5.93%	18707	13.89%	16752	1.99%	19007	15.72%	16803	2.30%
s3-A	10220	13252	29.67%	10773	5.41%	11934	16.77%	10478	2.52%	12061	18.01%	10519	2.93%
s3-B	13682	17365	26.92%	14511	6.06%	15743	15.06%	13986	2.22%	15897	16.19%	14081	2.92%
s3-C	17230	21055	22.20%	18411	6.85%	19842	15.16%	17538	1.79%	20046	16.35%	17653	2.46%
s4-A	12268	15817	28.93%	13076	6.59%	14350	16.97%	12647	3.09%	14497	18.17%	12737	3.82%
s4-B	16321	19882	21.82%	17553	7.55%	18577	13.82%	16693	2.28%	18854	15.52%	16776	2.79%
s4-C	20517	25214	22.89%	21513	4.85%	23645	15.25%	21071	2.70%	24031	17.13%	21149	3.08%
Avg.			*4.61%*		*6.20%*		*13.12%*		*1.32%*		*14.07%*		*1.75%*

Table 5. Experimental results for *egl* instances.

Table 7 summarizes the aforementioned results in a single table. From the averages in the last row of the table, it can be deduced that the performance of the RandSHARP algorithm is far superior to the performance of the RandPSH algorithm in both considered metrics. In fact, the 87 instance average gap associated with the RandSHARP algorithm for the BEST10 metric is just of 1.10%, which taking into consideration the limited computation time employed (180 s per instance in a standard PC) is a quite competitive result. This is specially the case if we remember that the current version of the algorithm does not employ any local search process, and that it only uses a single parameter – the one associated with the biased random distribution employed. Notice also that, as expected, the performance of both randomized algorithms – which results improve with more computing time – is much better than the performance of the deterministic heuristics. Finally, notice that the performance of our SHARP heuristic is quite superior to that of the classical Path Scanning heuristic.

Figure 8 shows a multiple box plot which allows a visual comparison of the algorithms' performance for the BEST10 metric. A very similar visualization is obtained when considering the AVG10 metric instead –for that reason we did not consider necessary to include the corresponding multiple box plot. The multiple box plot reinforces the idea that results from the RandSHARP algorithm are far superior to

results obtained with the RandPSH and, of course, to those generated by any heuristic. It also shows that the heuristic introduced in this chapter is far superior to the classical Path Scanning heuristic – this is true both for the BEST10 and AVG10 metrics. As a conclusion, the SHARP heuristic constitutes a remarkable candidate to generate the initial solution in most metaheuristic approaches for the CARP.

		Average gap w.r.t. BKS				Average gap w.r.t. BKS in MaxTime = 180s							
		Heuristic				BEST10				AVG10			
		PSH		SHARP		RandPSH		RandSHARP		RandPSH		RandSHARP	
Set	**BKS**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**	**Cost**	**Gap**
val1A	173	188	8.67%	187	8.09%	173	0.00%	173	0.00%	173	0.00%	173	0.00%
val1B	173	188	8.67%	198	14.45%	173	0.00%	175	1.16%	173	0.00%	181	4.62%
val1C	245	287	17.14%	266	8.57%	255	4.08%	247	0.82%	257	4.85%	248	1.27%
val2A	227	269	18.50%	249	9.69%	227	0.00%	229	0.88%	227	0.00%	232	2.33%
val2B	259	284	9.65%	303	16.99%	261	0.77%	260	0.39%	261	0.77%	260	0.42%
val2C	457	550	20.35%	490	7.22%	483	5.69%	462	1.09%	483	5.69%	462	1.09%
val3A	81	86	6.17%	89	9.88%	81	0.00%	81	0.00%	81	0.00%	81	0.37%
val3B	87	98	12.64%	99	13.79%	87	0.00%	87	0.00%	87	0.13%	88	1.15%
val3C	138	157	13.77%	158	14.49%	141	2.17%	139	0.72%	141	2.17%	139	0.72%
val4A	400	472	18.00%	446	11.50%	400	0.00%	404	1.00%	403	0.75%	407	1.68%
val4B	412	508	23.30%	453	9.95%	419	1.70%	424	2.91%	426	3.32%	428	3.81%
val4C	428	539	25.93%	484	13.08%	448	4.67%	438	2.34%	454	5.97%	451	5.42%
val4D	530	710	33.96%	588	10.94%	557	5.09%	543	2.45%	568	7.09%	548	3.45%
val5A	423	487	15.13%	579	36.88%	423	0.00%	427	0.95%	425	0.42%	436	3.10%
val5B	446	496	11.21%	509	14.13%	449	0.67%	450	0.90%	450	0.95%	460	3.21%
val5C	474	524	10.55%	551	16.24%	481	1.48%	485	2.32%	484	2.09%	492	3.71%
val5D	575	707	22.96%	638	10.96%	608	5.74%	594	3.30%	617	7.30%	609	5.83%
val6A	223	258	15.70%	235	5.38%	223	0.00%	225	0.90%	223	0.00%	228	2.33%
val6B	233	258	10.73%	251	7.73%	233	0.00%	233	0.00%	233	0.00%	234	0.52%
val6C	317	386	21.77%	355	11.99%	328	3.47%	321	1.26%	332	4.63%	324	2.33%
val7A	279	303	8.60%	305	9.32%	279	0.00%	279	0.00%	279	0.00%	280	0.39%
val7B	283	320	13.07%	301	6.36%	283	0.00%	286	1.06%	283	0.00%	288	1.66%
val7C	334	390	16.77%	367	9.88%	343	2.69%	342	2.40%	347	3.79%	345	3.23%
val8A	386	434	12.44%	458	18.65%	386	0.00%	391	1.30%	387	0.26%	394	2.07%
val8B	395	450	13.92%	469	18.73%	396	0.25%	406	2.78%	403	2.05%	414	4.78%
val8C	521	617	18.43%	613	17.66%	554	6.33%	541	3.84%	558	7.17%	553	6.18%
val9A	323	375	16.10%	336	4.02%	324	0.31%	326	0.93%	326	0.89%	329	1.70%
val9B	326	359	10.12%	359	10.12%	332	1.84%	333	2.15%	335	2.76%	338	3.56%
val9C	332	371	11.75%	364	9.64%	339	2.11%	341	2.71%	346	4.25%	346	4.22%
val9D	389	474	21.85%	440	13.11%	416	6.94%	402	3.34%	425	9.28%	411	5.63%
val10A	428	451	5.37%	460	7.48%	431	0.70%	435	1.64%	436	1.92%	437	2.10%
val10B	436	473	8.49%	472	8.26%	445	2.06%	447	2.52%	450	3.11%	450	3.17%
val10C	446	484	8.52%	484	8.52%	463	3.81%	459	2.91%	467	4.81%	466	4.39%
val10D	525	625	19.05%	587	11.81%	563	7.24%	543	3.43%	568	8.25%	559	6.55%
Avg.			*14.98%*		*11.93%*		*2.05%*		*1.60%*		*2.78%*		*2.85%*

Table 6. Experimental results for *val* instances.

	Average gap w.r.t. BKS		Average gap w.r.t. BKS in MaxTime = 180s			
	Heuristic		BEST10		AVG10	
Set	**PSH**	**SHARP**	**RandPSH**	**RandSHARP**	**RandPSH**	**RandSHARP**
egl	24.61%	6.20%	13.12%	1.32%	14.07%	1.75%
gdb	7.09%	7.56%	0.54%	0.26%	0.67%	0.53%
kshs	9.87%	9.87%	0.84%	0.62%	0.84%	0.62%
val	14.98%	11.93%	2.05%	1.60%	2.78%	2.85%
Avg.	*15.20%*	*9.05%*	*4.62%*	*1.10%*	*5.20%*	*1.78%*

Table 7. Summary of experimental results by dataset.

Additionally, two ANOVA tests for comparing the performance of each approach using the BEST10 and AVG10 metrics were also performed. **Figure 9** shows the ANOVA table corresponding to the AVG10 metric (a similar output is obtained for the ANOVA associated with the BEST10 metric). The corresponding p-value is 0.000, and thus it is possible to conclude that not all the approaches have the same performance level, i.e. some algorithms perform significantly better than others. Since the individual 95% confidence intervals are clearly disjoint, it seems reasonable to conclude that, in fact, algorithms' average performances are significantly different. A Fisher's test for differences contributed to confirm this point.

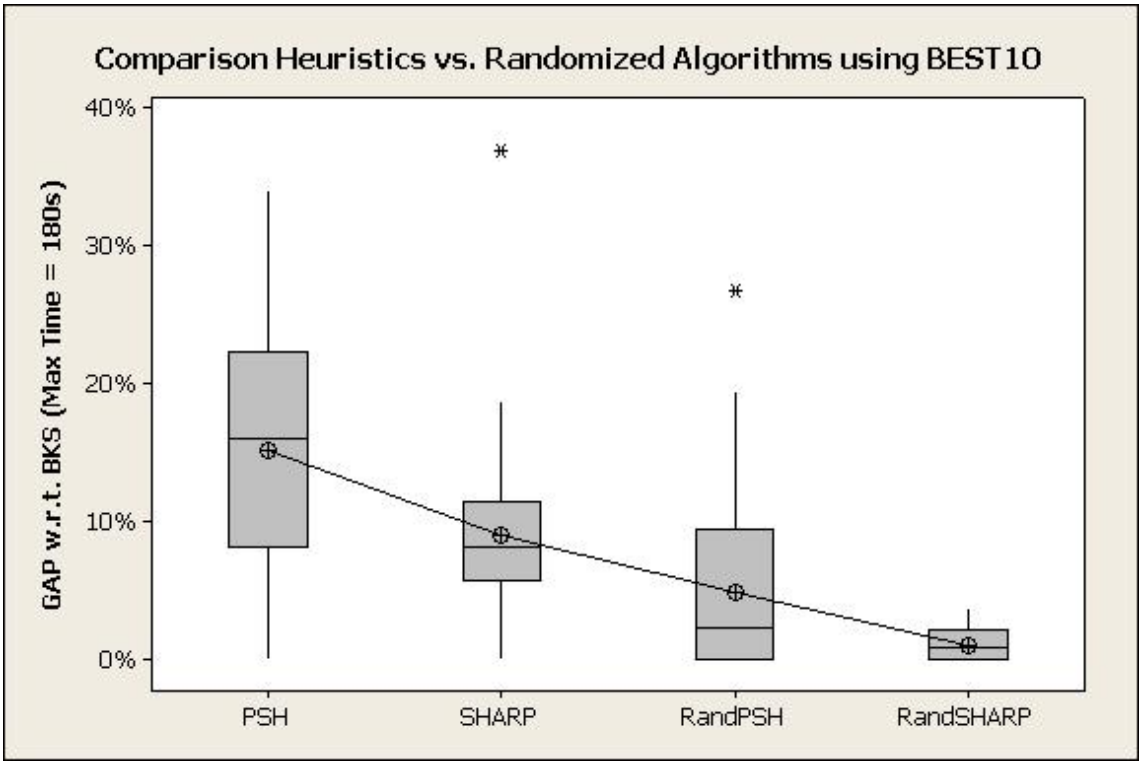

Figure 8. Visual comparison of the different approaches.

3.5.2 A comparison using time-evolution of the gap

In order to discuss the effect of computing time on each approach, several instances have been randomly selected and then solved using each of the approaches considered in this chapter and also different maximum computing times (up to 300 s). Needless to say that both heuristics are very fast, providing immediate feedback – i.e., a solution in about one second or even less – for most 'small' and 'medium' instances. Despite their speed, these heuristics are basically deterministic approaches which generate solutions with a noticeable gap with respect to the BKS. On the contrary, the

randomized algorithms are stochastic procedures which are able to improve their respective solutions with more computing time or more computing resources (e.g., using parallelization techniques).

Figure 10 shows the time-evolution of the gaps associated with each approach for the instance egl-e2-A (77 nodes). Notice that the classical Path Scanning heuristic offers a gap close to 30%, which in general terms can be considered as an enormous gap. This gap can be reduced with time using the RandPSH. In fact, the RandPSH quickly diminishes the gap down to a 10% or so. For this instance, our SHARP heuristic is much more efficient, providing an instantaneous gap close to 5%. Moreover, the RandSHARP algorithm is able to reduce that gap down to almost 0% in just a few seconds. Similarly, **Figure 11** shows the time-evolution of the gaps associated with each approach for the instance gdb8 (27 nodes). Again, both heuristics seem to provide far-from-optimal solutions which can be quickly improved by using the randomized algorithms. In particular, our RandSHARP method is able to diminish the gap down to 1% in just a few seconds of computation.

One-way ANOVA: PSH, SHARP, RandPSH AVG10, RandSHARP AVG10

```
Source   DF       SS       MS      F      P
Factor    3  0.86362  0.28787  77.10  0.000
Error   344  1.28442  0.00373
Total   347  2.14804

S = 0.06110   R-Sq = 40.20%   R-Sq(adj) = 39.68%


Level             N     Mean    StDev
PSH              87  0.15197  0.08849
SHARP            87  0.09051  0.05315
RandPSH AVG10    87  0.05204  0.06307
RandSHARP AVG10  87  0.01779  0.01738

                 Individual 95% CIs For Mean Based on Pooled StDev
Level            ---------+---------+---------+---------+
PSH                                                 (--*--)
SHARP                                    (---*--)
RandPSH AVG10                 (--*--)
RandSHARP AVG10  (--*---)
                 ---------+---------+---------+---------+
                        0.040     0.080     0.120     0.160

Pooled StDev = 0.06110
```

Figure 9. ANOVA for comparing performances.

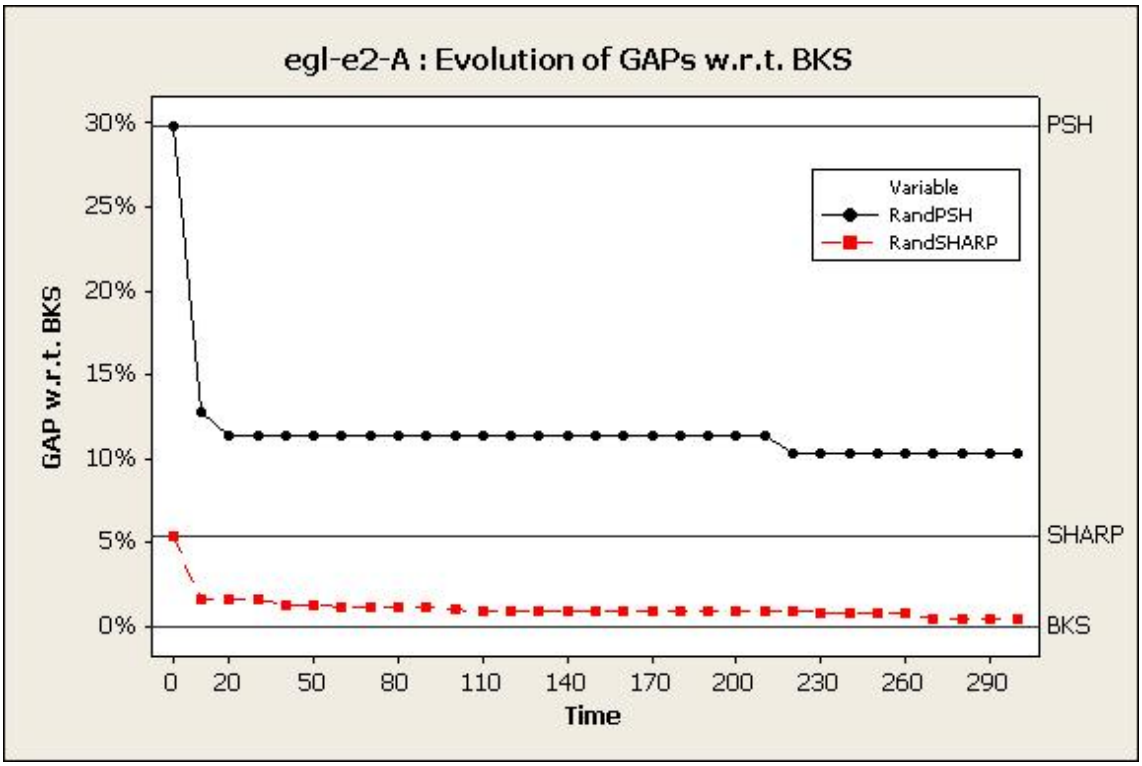

Figure 10. GAP time-evolution for the egl-e2-A instance.

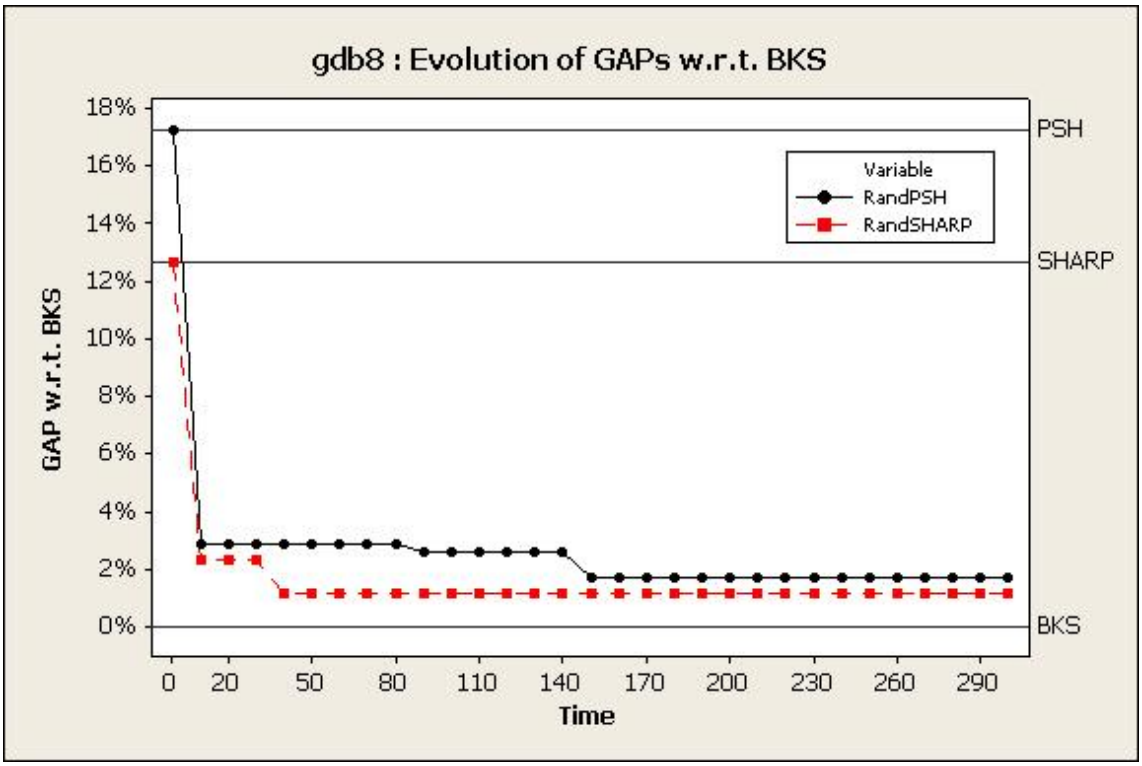

Figure 11. GAP time-evolution for the gdb8 instance.

3.6 Chapter Conclusions

In this chapter we have introduced a first application of the MIRHA framework to the CARP. The CARP is a problem suitable for modeling certain Telecommunication optimization problem. This increases the possible applications of the MIRHA framework which originally was only applied to Transportation & Logistics problems. A new heuristic and two randomized algorithms for solving the CARP have been proposed and evaluated. The SHARP heuristic is based on the savings concept extensively used in the Vehicle Routing Problem arena, and it shows to be highly efficient when compared with the classical Path Scanning heuristic. The randomized algorithms are based on a multi-start biased randomization of a classical heuristic. Thus, the first randomized algorithm, RandSHARP, introduces a biased randomization process into the SHARP heuristic, while the second one randomizes the PSH. As far as we know, it is the first time that a savings based heuristic for solving the CARP is presented in detail. The empirical tests show that employing biased randomization is an efficient way of quickly improving classical heuristics without introducing complexity to them. As a matter of fact, a notable characteristic of our approach is that it only uses one single parameter and, therefore, no complex fine-tuning processes are required. Moreover, the results show that the multi-start biased randomization of the SHARP heuristic is able to provide competitive results in a reasonable computing time period, improving by far other approaches in the literature. Also we can remark that the proposed heuristic SHARP, is better suited for the MIRHA framework than the classical PSH, as results demonstrate.

4 Arc Routing Problem with Stochastic Demands

Parts of this chapter have been taken from the co-autored publications: **Gonzalez-Martin et al. (2012b)** and **Gonzalez-Martin et al. (2014b)**.

Telecommunication optimization problems in real-life scenarios usually have a component which is not deterministic. For instance, the real demand required by the users usually is not a deterministic value that can be anticipated, but has a random component which makes it somehow uncertain. Another example is the background noise which interfere the communication channel, especially for wireless communications. In those cases, a deterministic model is not suitable to properly evaluate the real scenario. However, even though the values cannot be known beforehand, one can infer some probabilistic distribution based on real data from the past which could model the expected value for the random variable.

The Arc Routing Problem with Stochastic Demands (ARPSD) is a variant of the deterministic CARP which generalizes it by considering customers' demands as random variables instead of as deterministic values. In fact, we assume that the random demand associated with each customer can be modeled by a probability distribution, and that its actual value is not determined until the vehicle reaches the customer's arc. Notice that this random behavior of the customers' demands can lead to infeasibility of the planned routing solutions whenever the actual demand in a route exceeds the capacity of the assigned vehicle. This situation generates a "*route failure*" and requires from corrective actions to guarantee that all customers' demands will be satisfied. Of course, corrective actions might increase the total cost of the implemented solution, so they must be taken into account during the routes-design stage.

Considering stochastic demands instead of deterministic ones is a more realistic but also difficult scenario. Unfortunately, most of the existing literature focuses on the deterministic case. Therefore, the main goal of our approach is to contribute to fill the lack of methods for solving the stochastic case by proposing a Simheuristic algorithm that combines Monte Carlo Simulation (MCS) with a metaheuristic that was

originally designed for solving the deterministic CARP. As it will be discussed later, this algorithm deals with the stochastic variant of the problem in a natural and efficient way. Our work is based on the use of a safety stock during the route-design stage. This capacity surplus can be used during the delivery stage to handle unexpected demands. With that, our methodology is able to cover route failures, thus limiting the impact of corrective actions over the total delivery costs. A reliability index is also defined to evaluate the robustness of each solution with respect to possible route failures caused by random demands. This reliability index can be helpful for decision makers when choosing among several distribution plans with similar expected total costs.

4.1 Literature Review

The research body concerning the stochastic version of the CARP is quite limited yet, especially when compared with its deterministic counterpart. As far as we know, the ARPSD was first considered by **Fleury et al. (2002)** and further extended by the same authors in **Fleury et al. (2005)**. In these works, the ARPSD was not approached directly. Instead, the scope was to evaluate the robustness of the solutions obtained for the classical CARP if the demands were stochastic instead of deterministic. In particular, the latter work contains an application to the CARP of the Hybrid Genetic Heuristic proposed by **Lacomme et al.(2001)**. Different solutions were obtained by varying the vehicle capacity in each run of the heuristic. The solutions obtained were then evaluated by means of simulation studies. Although this approach resembles somewhat to the one we are proposing, there are relevant differences among them, e.g.: our proposal uses a different metaheuristic framework, and it employs MCS to estimate the total expected costs –including routing plus corrective actions- as well as reliability indexes of each solution.

The ARPSD with Normally distributed demands was first approached by **Fleury et al. (2004)**. The authors propose a Memetic Algorithm, which is an extension of the algorithm suggested by **Lacomme et al. (2004a)**. For each edge, the Normal distribution describing the demand was truncated to avoid negative values or demands exceeding the vehicle capacity. The problem was further restricted, since a route was only allowed to fail once. An exact method for the ARPSD was proposed by **Christiansen et al. (2009)**. It is motivated on a previous work for solving the VRPSD presented by the same authors in **Christiansen and Lysgaard (2007)**. In particular, they formulated the ARPSD as a Set Partitioning Problem and developed a Branch-and-Price algorithm in which the pricing incorporates demands with stochastic nature. **Laporte et al. (2010)** have developed a local search approach for the stochastic

version of the undirected CARP in the context of garbage collection. In this chapter, a first-stage solution is constructed by means of an Adaptive Large Neighborhood Search Heuristic (ALNS) that takes the expected cost of recourse into account. Closed form expressions were derived for the expected cost of recourse by extending the concept of route failure commonly used in stochastic node-routing problems. Their computational results show that ALNS solutions were better than those obtained by first optimally solving a deterministic CARP and then computing the expected cost of recourse by using random variables for the demands.

Finally, other works related to the ARPSD that are worth to mention are found in **Chen et al. (2009)** and **Ismail and Ramli (2011)**. In the former, an arc-routing problem motivated by a real world application in small-package delivery was addressed. In this problem, the uncertainty is considered and incorporated to a proposed model called Probabilistic Arc Routing Problem (PARP). The PARP solution procedure incorporates the probabilities into an adapted local search that was primarily designed for the Probabilistic Traveling Salesman Problem by **Bertsimas and Howell (1993)**. Similarly, **Ismail and Ramli (2011)** considered a rich CARP based on waste collection operations. They studied how rain drops affect the weight of the collected waste. These authors also developed a constructive heuristic called Nearest Procedure Based on Highest Demand/Cost.

4.2 Our Approach

Our proposed methodology is based on two main facts: (a) the ARPSD can be seen as a generalization of the CARP, i.e. the CARP can be considered a special case of the ARPSD where the random demands have zero variance; and (b) as discussed in the literature review section, while the ARPSD is yet an emerging research area, efficient metaheuristics do already exist for solving the deterministic CARP. Accordingly, one of the fundamental ideas behind our approach is to transform the challenge of solving a given ARPSD instance into a new challenge which consists in solving several *conservative* CARP instances, each of them characterized by a specific risk (probability) of showing route failures. The term conservative refers here to the fact that only a certain percentage of the total capacity is considered during the route design phase. This latent capacity will be available if the actual demand of the route is greater than expected. This unused capacity can be considered as a safety stock, since it reflects the level of latent capacity that is maintained to buffer against possible route failures.

A similar approach was already introduced by **Juan et al. (2011a)** for the Vehicle Routing Problem with Stochastic Demands, which can be seen as the stochastic version of the classical VRP. In our research, we also use the Randomized Savings Heuristic for the Arc Routing Problem (RandSHARP) described in **Chapter 3**. **Figure 12** shows the flowchart of our Simheuristic approach, an overview of which is given next:

1. Consider an ARPSD instance defined by a network of arcs, a depot, a set of customers, and a capacity W. Assume that each customer has a positive stochastic demand characterized by a specific probability distribution with known mean.
2. Consider a specific value for the parameter k ($0 \leq k \leq 1$), which sets the percentage of W that will be used during the route design stage, i.e.: instead of considering the total capacity, W, we will assume 'virtual' vehicles with capacity given by $W^* = k\,W$ (thus leaving a safety stock for emergencies).
3. Consider the CARP(W*) instance defined by the expected demands of each customer and with capacity W*.
4. Solve the CARP(W*) instance using the RandSHARP algorithm. The obtained solution, s, will be also a feasible solution for the original ARPSD as long as the total route demand computed during the actual delivery stage does not exceed the surplus capacity (i.e., the safety stock).
5. Using the solution s, estimate throughout MCS the expected cost due to possible failures on any route. To this end, random demands for each customer are generated using the associated probability distribution and, whenever a route failure occurs, a corrective action is applied and its cost is registered. In our case, the corrective action consists in performing a round-trip from the arc causing the route failure to the depot, where the vehicle is reloaded so that it can resume the delivery route. After iterating this process some thousands of times, a random sample of costs is obtained, from which an average value can be estimated. Then, the expected total costs can be computed by adding these variable costs due to route failures and the fixed distance-based costs given by s.
6. During the same MCS, it is also possible to estimate the reliability of each route in s. Thus, a route-reliability index can be defined as the probability that a given route will not fail. It should be noticed that, in each route, higher-than-expected demands could sometimes be compensated by lower-than-expected demands.
7. The reliability index for s is then computed as the product of each route-reliability index –under the reasonable hypothesis that customer demands are

independent. This s-reliability index can be considered as a measure of the solution robustness in the stochastic scenario.

8. Repeat the process from **Step 2** with a new value of the parameter k to explore the convenience of using a different level of safety stocks inside each vehicle.
9. Finally, return the solution with the lowest expected total costs found so far (or, alternatively, a list with the best solutions found so far so that the decision maker can choose according to both total expected costs and reliability indices).

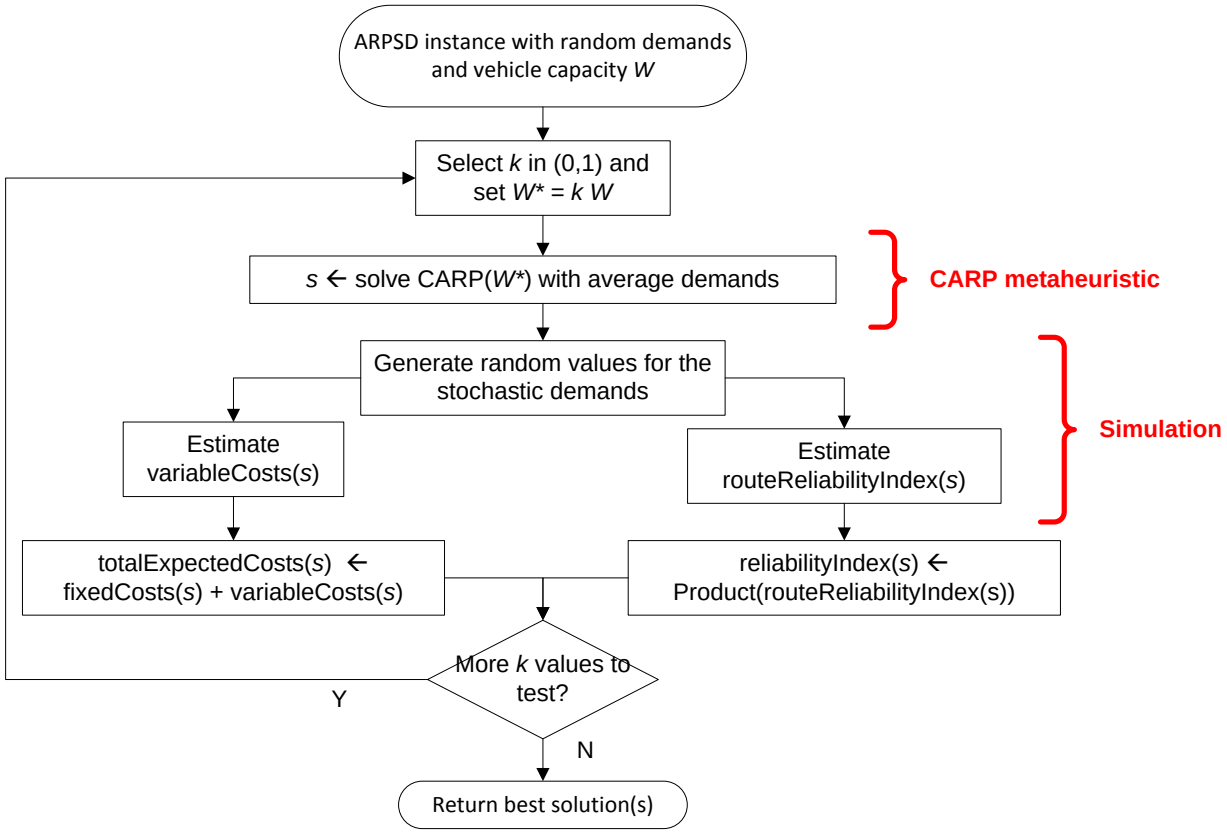

Figure 12. Flowchart diagram of our Simheuristic algorithm

4.3 Implementation details of our Approach

This section provides the pseudo-code details of our algorithm. These details allow other authors and end users to quickly implement our algorithm in order to: (a) reproduce the experiments we have run and compare our approach against other approaches; and (b) use our approach to solve real-life applications of the ARPSD. Thus, **Figure 13** shows the main procedure of our algorithm. This procedure starts by extracting the average value from the probability distribution associated to each customer's demand. Then, after initializing the best solution, it starts a loop to try

different configurations of the safety stock level in each vehicle. Usually, one can start by using k = 1 (zero safety stock, scenario with the highest possible variable costs due to corrective actions) and then decreasing k at each iteration in steps of size 0.1 or 0.2 until the increase in distance-based fixed costs overpass the savings in variable costs. For each value of k, the virtual vehicle capacity is computed as W* = k W, and a nested iterative process starts. At each iteration of this process, the RandSHARP algorithm described in **Chapter 3** is used to generate a new 'good' but random solution to the associated CARP. The fixed cost associated with this solution is given by the distance-based cost of the CARP instance. Then, in order to compute the variable cost, a simulation is employed –more details on this stage are given later. By adding both costs, it is possible to obtain an estimate of the total expected cost. Notice that the same simulation process also allows to obtain the reliability index associated with the current solution. Finally, whenever the new solution outperforms the best-found solution, the latter is updated. In fact, it is probably a good idea to keep a list of 'top' solutions so that the decision maker can choose among them using both the total expected cost as well as the reliability index criteria.

```
procedure SimRandSHARP(arcs, probDist, W, maxTime)
01    avgDemands ← getAverage(probDist)
02    bestSol ← empty
03    for {each desired value of k} do % try different user-defined k
04      W* ← k^W
05      while {elapsed < maxTime} do % time-based stopping criterion
06        sol ← randSHARP(arcs, avgDemands, W*) % biased-radomization
07        fixedCost ← getDistanceBasedCost(sol) % deterministic costs
08        avgVariableCost ← simulation(sol, probDist) % uses MCS
09        reliabilityIndex ← simulation(sol, probDist) % uses MCS
10        totalAvgCost ← fixedCost + avgVariableCost
11        if {totalAvgCost < cost(bestSol)} or {bestSol is empty} then
12          bestSol ← sol
13          cost(bestSol) ← totalAvgCost
14          reliabilityIndex(bestSol) ← reliabilityIndex
15        end if
16      end while
17    end for
18    return bestSol % returns the best(s) solution(s) found so far
end procedure
```

Figure 13. SimRandSHARP algorithm main procedure

Figure 14 shows how a 'good' yet random solution to the CARP(W*) is generated throughout the RandSHARP algorithm. Basically, this algorithm uses a biased-randomized version of the SHARP described in **Chapter 3**. The biased randomization process induces a 'soft randomization' in the order of the sorted savings list by using a skewed (non-symmetric) probability distribution. This allows keeping the logic behind

the savings-based heuristic while, at the same time, helps to generate different solutions each time the randomized heuristic is run. For a more detailed discussion of the biased-randomization process and an example of application to a related field, the reader is addressed to **Juan et al (2010)**.

```
procedure randSHARP(arcs, avgDemands, W*)
01    sol ← getDummySol()% generate a dummy sol
02    savingsList ← genSavingsList(arcs) % generates the savings list
03    randSavingsList ← biasedRand(savingsList)
04    while {savingsList contains edges} do % apply savings-based heu.
05      edge ← extractNextEdge(randSavingsList)
06      sol ← randSHARP(arcs, avgDemands, W*) % biased-radomization
07      iNode ← getOriding(edge)
08      jNode ← getEnd(edge)
09      iRoute ← getRoute(iNode)
10      jRoute ← getRoute(jNode)
11      if {iRoute is not jRoute} and {cap. after merge <= W*} then
12        mergedRoute ← mergeRoutes(iRoute, jRoute, iNode, jNode)
13        sol ← remove(iRoute, sol)
14        sol ← remove(jRoute, sol)
15        sol ← add(mergedRoute, sol)
16      end if
17    end while
18    return sol % returns a random ´good´ solution
end procedure
```

Figure 14. RandSHARP procedure

```
procedure simulation(sol, probDist)
01    nTrials ← 0
02    nFailures ← 0
03    varCosts ← 0
04    solReliability ← 1
05    while {termination condition not met} do
06      nTrials ← nTrials + 1
07      for {each route in sol} do
08        totalRouteDemands ← getRandomVariates(route, probDist)
09        if {totalRouteDemands > vehCapacity} then
10          nFailures(route) ← nFailures(route) + 1
11          varCosts ← varCosts + cost(corrective action)
12        end if
13      end for
14    end while
15    avgVarCost ← varCosts / nTrials
16    for {each route in sol} do
17      routeReliability ← nFailures(route) / nTrials
18      solReliability ← solReliability * routeReliability
19    end for
20    return avgVarCosts and solReliability
end procedure
```

Figure 15. Simulation procedure to estimate variable cost and reliability index

Finally, **Figure 15** illustrates the basic ideas behind the simulation procedure, which is used to obtain estimates for both the expected variable cost of a given solution as well as its reliability index. Notice that the probability distributions associated with each customer's random demands are used to generate sample observations. These sample observations allow determining whether the actual aggregated demand in a single route will exceed the vehicle capacity in the current simulation run. If so, then both the new route failure and the variable cost associated with the corrective action are considered. It is worthy to notice that, despite not explicitly described in this pseudo-code, a route could show more than one failure in cases in which the actual aggregated demand exceeds by far the vehicle real capacity.

4.4 Results

The methodology described in the previous sections has been implemented as a Java application. In our experiments, a standard personal computer was used to perform all tests: an Intel® Core™2 Quad CPU Q9300 at 2.50 GHz and 8 GB RAM running with a Windows® 7 Pro operating system. Four different datasets, originally defined for the CARP, were adapted (generalized) and employed in our tests (details of the datasets can be found in **Section 3.5**). In order to generalize these datasets for the ARPSD, we changed the original deterministic demands by random demands with known probability distributions and means given by the deterministic demands.

Since our approach uses simulation, random demands can be modeled by any probability distribution with a known mean. In this case, we have selected a Log-Normal distribution for modeling the demands. Notice that historical data would be used in a real-life scenario to model each customer's demand by a different probability distribution –the one that best fits the existing observations. The Log-Normal distribution, which is a more natural choice than the Normal distribution when modeling non-negative customers' demands, has two parameters: the location parameter, μ_i, and the scale parameter, σ_i. According to the properties of the Log-Normal distribution, these parameters will be given by the following expressions:

$$\mu_i = \ln(E[Q_i]) - \frac{1}{2}\ln\left(1 + \frac{Var[Q_i]}{E[Q_i]^2}\right) \tag{10}$$

$$\sigma_i = \left|\sqrt{\ln\left(1 + \frac{Var[Q_i]}{E[Q_i]^2}\right)}\right| \tag{11}$$

4.4.1 A Numerical Example

Before performing a complete experiment on all tests, we will discuss an illustrative example based on the instance *egl-s4-B*. First, this instance is generalized by considering the customer demands, Q_i, as random variables following a Log-Normal distribution with mean $E[Q_i]$ and variance $0.25 \cdot E[Q_i]$. Then, we set $k=0.95$, which means that a 5% of the total vehicle capacity is used as a safety stock during the route design stage to deal with unexpected demands during the actual delivery stage. At this point, the RandSHARP algorithm is used to solve the associated CARP instance –notice that any other efficient algorithm for solving the CARP could be used instead. Next, we run a (short-run) simulation over the obtained solution to obtain estimates of both the expected variable cost –due to corrective actions- and the solution reliability index. Once obtained the best solution for the current value of k, other values of this parameter are evaluated. **Table 8** shows the results obtained for different k values. In this case, the value $k=0.94$ will be selected as the one providing the solution with the lowest expected total costs. Notice that, as the value of k decreases (i.e., as higher levels of safety stock are considered), the number of necessary routes (vehicles) increases, and the same can be said for the reliability level and the fixed cost –as opposed to what happens with the expected variable cost.

k	# routes	Fixed cost	Expected variable costs	Expected total costs	Reliability index
0.90	30	18363	379.6	18742.6	0.99
0.91	30	18278	489.6	18767.6	0.99
0.92	30	18279	559.9	18838.9	0.99
0.93	30	18287	415.6	18702.6	0.99
0.94	*29*	*17855*	*475.8*	*18330.8*	*0.99*
0.95	29	17922	700.7	18622.7	0.98
0.96	29	17961	554.0	18515.0	0.98
0.97	28	17579	1024.9	18603.9	0.98

Table 8. Results for the egl-s4-B instance with $Var[Q_i]=0.25 \cdot E[Q_i]$.

4.4.2 Computational Results

In our computational experiments we considered the following four scenarios regarding the variance levels of each instance:

(i) a 'low-variance' scenario with $Var[Q_i] = 0.05 \cdot E[Q_i]$

(ii) a 'medium-variance' scenario with $Var[Q_i] = 0.25 \cdot E[Q_i]$

(iii) a 'high variance' scenario with $Var[Q_i] = 0.75 \cdot E[Q_i]$

(iv) a 'very-high variance' scenario with $Var[Q_i] = 2 \cdot E[Q_i]$.

Notice that, since the expected value of the demand for each arc is set to the deterministic value of the classical CARP benchmarks, we have been able to generalize these CARP benchmarks in a natural way. In other words, the classical benchmarks are retrieved as a particular case of the new ones when $Var[Q_i] = 0$. Obviously, as uncertainty in arc demands increases, total expected costs will tend to increase. This is so because more reliable or robust solutions will be required to avoid unnecessary route failures and their costly recourse actions.

For each of the four scenarios considered, we proceed as follows. First of all, we computed a pseudo-optimal solution for the deterministic CARP using the RandSHARP algorithm. Notice that this solution can be also used as a feasible solution for the ARPSD, although it will probably show 'high' variable costs due to the necessary recourse actions –no safety stocks are considered in this CARP solution. In order to obtain estimates of these variable costs, we used simulation. Note that the fixed costs of the pseudo-optimal solution for the CARP can be considered as a lower bound for the total costs of the ARPSD pseudo-optimal solution. Similarly, total expected costs associated with the CARP pseudo-optimal solution represent an upper bound for the ARPSD pseudo-optimal solution.

Once these lower and upper bounds were established, we used our Simheuristic algorithm to generate solutions for the ARPSD. In our experiments, we varied the *k* parameter from 0.75 to 1.00, using a step of size 0.01. For each k-value and instance, a complete execution of 180 seconds was performed. Then, the k-value providing the lowest total expected cost was selected as the recommended one. **Table 9 to Table 12** shows the average results, for each dataset of instances, obtained in each of the analyzed scenarios. For extended results refer to **Annex A.1**.

In particular these tables show the following columns: (i) dataset name; (ii) fixed cost associated with the pseudo-optimal solution for the deterministic CARP (1); (iii) total expected cost associated with the former solution when random variables are considered and recourse actions are accounted for (2), (iv) percentage gap between values in (1) and (2); (v) total expected cost associated with our best solution (OBS) for the ARPSD (3); (vi) gap between (1) and (3); (vii) gap between (2) and (3); (viii) associated k-value (inverse of the safety stock level); and (ix) reliability value associated with the proposed solution.

	Average values for all instance in each dataset								
	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Dataset	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
egl	9906	10825	9.27%	0.97	10145	2.41%	-6.28%	0.995	1.00
gdb	255	358	40.57%	0.84	322	26.51%	-10.00%	0.943	0.94
kshs	11140	11143	0.03%	0.97	11141	0.01%	-0.02%	0.993	1.00
val	350	395	12.62%	0.93	369	5.15%	-6.64%	0.942	0.99
Avg.			*17.43%*	*0.92*		*8.57%*	*-6.97%*	*0.960*	*0.98*

Table 9. Summary of results when $Var[Q_i] = 0.05 \cdot E[Q_i]$

	Average values for all instance in each dataset								
	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Dataset	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
egl	9906	12462	25.80%	0.91	10715	8.17%	-14.02%	0.984	0.99
gdb	255	380	49.33%	0.80	346	35.69%	-9.13%	0.966	0.90
kshs	11140	11593	4.06%	0.97	11218	0.70%	-3.23%	0.980	1.00
val	350	457	30.33%	0.82	398	13.53%	-12.89%	0.906	0.97
Avg.			*31.41%*	*0.85*		*15.96%*	*-11.36%*	*0.949*	*0.96*

Table 10. Summary of results when $Var[Q_i] = 0.25 \cdot E[Q_i]$

	Average values for all instance in each dataset								
	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Dataset	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
egl	9906	14039	41.71%	0.87	11656	17.66%	-16.97%	0.958	0.96
gdb	255	395	54.95%	0.76	360	41.31%	-8.81%	0.879	0.89
kshs	11140	11938	7.16%	0.96	11387	2.21%	-4.62%	0.973	0.97
val	350	502	43.17%	0.77	439	25.04%	-12.66%	0.826	0.89
Avg.			*42.07%*	*0.81*		*24.40%*	*-12.08%*	*0.886*	*0.94*

Table 11. Summary of results when $Var[Q_i] = 0.75 \cdot E[Q_i]$

	Average values for all instance in each dataset								
	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Dataset	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
egl	9906	16114	62.66%	0.81	12996	31.18%	-19.35%	0.936	0.89
gdb	255	400	57.18%	0.75	366	43.66%	-8.60%	0.877	0.85
kshs	11140	12900	15.80%	0.94	11845	6.33%	-8.18%	0.955	0.97
val	350	550	56.62%	0.71	488	39.00%	-11.25%	0.824	0.88
Avg.			*53.76%*	*0.76*		*34.00%*	*-12.50%*	*0.878*	*0.90*

Table 12. Summary of results when $Var[Q_i] = 2 \cdot E[Q_i]$

Additionally, **Figure 16** shows, for each uncertainty level, the gaps between: (a) the lower bound for the ARPSD (fixed cost of the CARP solution) and the upper bound (total expected cost of the CARP solution); and (b) the lower bound and our best solution for the ARPSD. Similarly, **Figure 17** summarizes, for each scenario, the average reliability indices associated with the CARP solution –when it is considered as a solution for the ARPSD– and our best solution for the ARPSD, respectively.

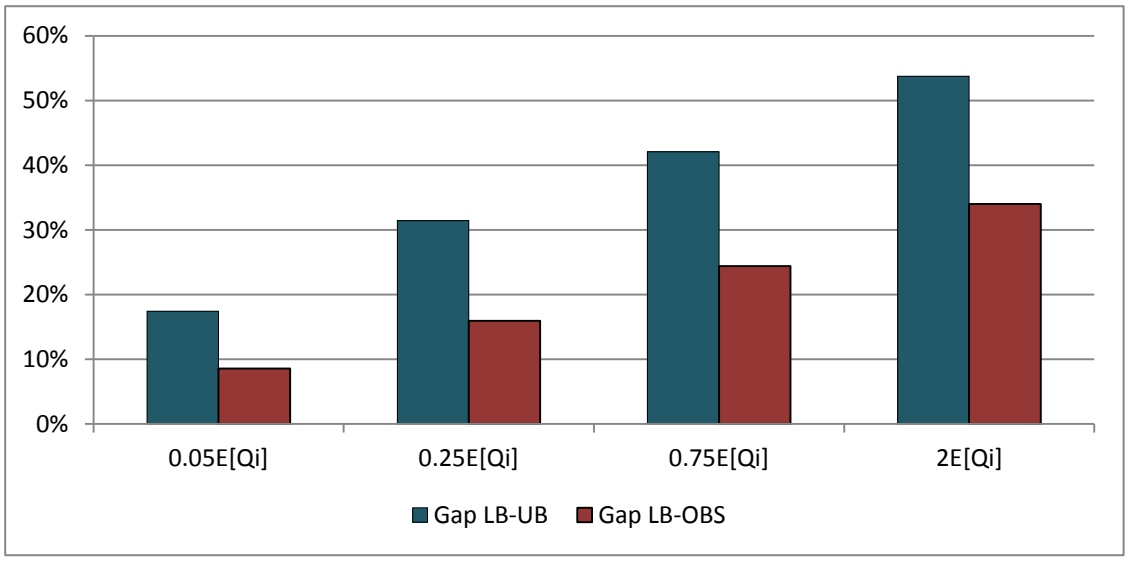

Figure 16. Average gaps with respect to ARPSD lower bound (BKS CARP).

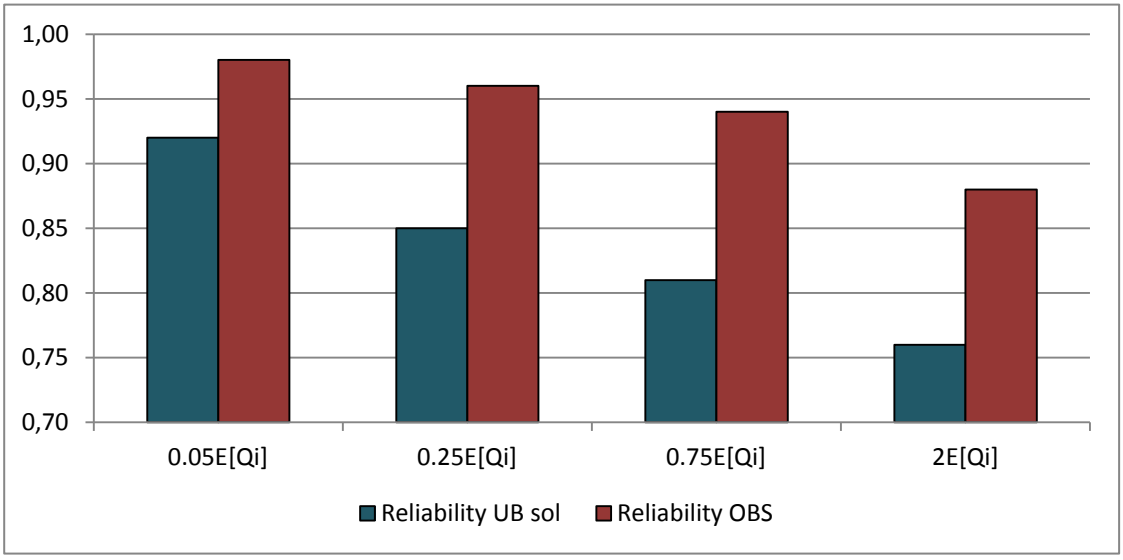

Figure 17. Average reliability indices.

4.4.3 Results Analysis

The aforementioned tables show how averaged total expected costs associated with our best solution for the ARPSD are always bounded by the fixed and total expected costs associated with the CARP solution. Notice also that, as the level of variability increases from one scenario to other, the size of the gaps is also increased while the recommended *k*-value tends to decrease. Put in other words, the more the uncertainty level in the arcs demands', the more safety stock is recommended and the more the pseudo-optimal solution for the ARPSD tends to differ from the pseudo-optimal solution for the deterministic CARP. Notice also that, for a given ARPSD solution, the higher the variance, the lower the reliability index of the solution. Also, the higher the variance, the

larger the gaps in terms of cost between the solution and the upper bound (expected costs for the BKS for the CARP).

Even in the low-variability scenario, results show that when used as solutions for the ARPSD, the pseudo-optimal solutions for the CARP provide lower reliability levels (0.92 on the average) than the ones obtained with our approach (0.98 on the average). This effect can be clearly seen in Figure 6 for the different scenarios. As previously discussed, lower reliability indices imply more routes failures which, at the end, cause higher variable costs. This also explains that, as shown in **Figure 16**, our best solution provides always a lower average gap with respect to the lower bound than the CARP solution when it is used as a solution for the ARPSD.

4.5 Chapter Conclusions

This chapter has analyzed the stochastic –and more realistic– version of the Arc Routing Problem, which has been seldom discussed in the scientific literature so far. After motivating its importance and reviewing the existing related work, this chapter has proposed a Simheuristic algorithm to solve the ARPSD in a natural way. Our algorithm combines an efficient metaheuristic for the deterministic version of the problem with a simulation stage, which is able to manage the uncertainty presented in the model. The concept of 'safety stock' is used during the route-design stage to reduce the negative effects generated by unexpected demands during the customers' delivery stage, i.e., this capacity surplus can be used to handle possible over demands that may appear due to the random nature of the demands of the customers. With that, our methodology is able to cover route failures, avoiding the over cost of corrective actions –e.g., a round trip to the depot to reload the vehicle. By employing simulation, our approach transforms the challenge of solving a stochastic instance into the challenge of solving a limited set of deterministic ARPs, each of them associated with a different level of safety-stock. This allows using well-tested and efficient metaheuristics –initially designed for solving the deterministic version of the problem– to deal with the stochastic version of the problem. The proposed methodology is also able to generate reliability indices for each solution, which can be interpreted as robustness indicators.

Several research lines remain open at this stage, among others: (a) explore other ways of integrating a simulation stage inside a metaheuristic algorithm; (b) analyze how parallel executions of this algorithm –each of them running with a different simulation seed– can speed up clock times necessary to obtain 'high-quality' solutions; (c) enrich the ARP model even further by including also stochastic costs due to random traveling times; and (d) developing similar Simheuristics for other combinatorial optimization

problems which have traditionally assumed deterministic inputs even when uncertainty is present in most real-life situations.

5 Non-smooth Arc Routing Problem

Parts of this chapter have been taken from the co-autored publications **Ferrer et al. (2013)** and **Gonzalez-Martin et al. (2014c)**.

Another scenario which is also common on real-life Telecommunication optimization problems is that in which the optimized function is non-convex. In the case of the CARP, a non-smooth variation of the problem can be that on which the capacity constraint is converted to a soft-constraint, which can be violated by incurring in some penalty cost. This means, for instance, that the capacity (bandwidth) constraint in a cable is not due to a physical factor but an economical one, which could be violated but incurring in some penalty.

Optimization problems can be classified, from a high-level perspective, as either convex or non-convex. In general, ConVex Optimization Problems (CVOPs) have two parts: a series of constraints that represent convex regions and an objective function to be minimized that is also convex. CVOPs are worth studying because they have a wide variety of applications and many problems can be reduced to them via change of variables. Linear Programming is one well-known example, since linear functions are trivially convex **(Boyd & Vandenberghe, 2004)**. The main idea in convex optimization problems is that every constraint restricts the space of solutions to a certain convex region. By taking the intersection of all these regions we obtain the set of feasible solutions, which is also convex. Due to the structure of the solution space, every single local optimum is a global optimum too. This is the key property that permits us to solve CVOPs exactly and efficiently up to very large instances. However, almost none of the algorithms applied for CVOPs can be extended to non-convex case.

In non-convex optimization (NCVOPs) the objective function, or even the feasible region, are not convex, which results in a far more complex solution space than the case of the CVOPs. In NCVOPs we have many disjoint regions, and multiple locally optimal points within each of them. As a result, if a traditional local search is applied, there is a high risk of ending in the vicinity of a local optimum that may still be far from the global optimum. Another drawback is that it can take exponential time in

the size of the input to determine that the NCVOP is infeasible, that the objective function is unbounded, or that one of the solutions found so far is the actual global optimum.

A function is smooth if it is differentiable and it has continuous derivatives of all orders. Therefore, a non-smooth function is one that is missing some of these properties. Non-smooth optimization problems (NSPs) are similar to NCVOPs in the sense that they are much more difficult to solve than traditional smooth and convex problems. The function for which a global optimum needs to be computed is now non-smooth and the solution space might contain again multiple disjoint regions and many locally optimal points within each of them. The computational techniques that can be used to solve these types of problem are often fairly complex and depend on the particular structure of the problem. While in convex optimization it is possible, sometimes, to explore the problem structure, and build solution methods that provide the global optimum, non-convex optimization problems are often intractable and have to rely on heuristic algorithms that produce only local optima. As a result, developing such techniques is in general time consuming, and the resulting application range is very limited. However, most real-life objective functions are either non-convex, non-smooth or both. Therefore, combinatorial optimization under these complex but common circumstances is an important field to explore.

5.1 Literature Review on Non-Smooth Problems

In the context of combinatorial optimization, probabilistic or randomized algorithms make use of pseudo-random numbers or variants during the construction or local search phases. In addition to the problem's input data, a probabilistic algorithm use random bits to do random choices during its execution. An important property is that for the same input the algorithm can produce different outputs in different runs. Probabilistic algorithms have been widely used to solve many combinatorial optimization problems. Examples are Vehicle Routing Problems **(Laporte, 2009)**, Location and Layout Problems **(Drezner and Hamacher, 2002)** or Covering, Clustering, Packing and Partitions Problems **(Chaves and Lorena, 2010)**.

Despite the great success of application of these methods to the aforementioned combinatorial problems, there exist only a few documented applications of these algorithms to the NCVOPs or NSPs. Some of the existing references are reviewed next. **Bagirov and Yearwood (2006)** present a formulation of the Minimum Sum-of-Squares clustering problem, which is a non-smooth, non-convex optimization problem. The goal of clustering problems is to separate a large set of

objects into groups or clusters based on certain criteria. The authors point out that a large number of approaches, like branch and bound or K-means algorithms, have been used for the clustering problem, but they are efficient only in certain special settings. The author remarks that, in general, better results are obtained when metaheuristics are used for the clustering problem. **Al-Sultan (1995)** proposed a Tabu Search approach that outperforms the K-means. However, this algorithm requires of three parameters, so an extensive study was necessary to find the best settings.

The issue of Optimal Routing in Communication Networks has also received a lot of attention from researchers. The objective is to find the best path for data transmission in short amount of time. The routing strategy can greatly affect the system performance, so there is a high demand for efficient algorithms. Numerous methods that deal with this challenge have been designed. **Hamdan and El-Hawary (2002)** proposed a method which combined Genetic Algorithms with Hopfield networks. **Oonsivialai et al. (2009)** proposed an approach based on Tabu Search. The main drawbacks of most of these methods are either their inability to efficiently explore the solution spacer or very long computational times.

Bagirov et al. (2007) present a non-smooth formulation for the Location Problem in Wireless Sensor Networks. In general, a wireless sensor network can be defined as a distributed collection of nodes that have limited resources and operate autonomously. The goal is to find o accurately estimate the position of the nodes. Most proposed approaches have assumed accurate range measurements, which is unrealistic for Radio Frequency signal strength measurements. **Ramadurai and Sichitiu (2003)** show that a probabilistic approach can be adopted to deal with range measurements inaccuracy.

Finally, in the transportation and logistics arena, **Juan et al. (2013)** presented a non-smooth formulation for the Vehicle Routing Problem. To solve this problem they proposed a hybrid algorithm for solving the problem.

5.2 The Non-Smooth CARP

As described in **Chapter 3**, the CARP is a combinatorial optimization problem defined over an undirected incomplete graph $G = (V, E, C, Q)$. Also, a set of *K* identical vehicles (homogeneous fleet) with restricted capacity *W* are available for serving the customer's demands. Under these circumstances, the usual goal is to find a set of routes which minimize the total delivery costs, computed as the sum of the costs of all the *K* routes, which are equals to the sum of the costs c_{ij} associated to each traversed arc (x_{ij}^k is a binary variable which denotes whether the arc is traversed by the *k*-th route):

$$\gamma_k = \sum_{e_{ij} \in E} c_{ij} \cdot x_{ij}^k \tag{12}$$

$$\min \sum_{k=1}^{K} \gamma_k \tag{13}$$

This minimization is subjected to the following constraints:

1. Every route starts and ends at the depot node so every route is a round-trip.
2. All the demands are satisfied.
3. Each arc with positive demand is served by exactly one vehicle. However, an arc can be traversed as many times as required by any vehicle.
4. The total demand to be served in any route does not exceed the vehicle loading capacity *W*.

As mentioned before, one of the main goals of this chapter is to fill the gap in the CARP literature regarding the discussion and solution of non-smooth objective functions, and to show the efficiency of our approach to deal with this kind of functions in the CARP context. In order to test the effectiveness of our procedure and its efficiency in relation to other existing approaches, we relaxed the constraints by violating some conditions, if necessary. We considered soft constraints, which allow conditions to be violated, by incurring in some penalty costs that must be added to the objective function rather than considering hard constraints, which constraint the problem to never exceed the maximum route costs. According to **Hashimoto et al. (2006)**, in real-world simulations, time windows and capacity constraints can be often violated to some extent. Of course, the same analysis can be applied to constraints associated with maximum route costs. In practice, if a given route exceeds a threshold cost or length, then some penalty cost must be added to the total route costs, and these penalty costs are likely to be defined by a piecewise non-smooth function. These costs will depend on the size of the gap between the actual route costs and the threshold. In this chapter we will define the non-smooth arc routing problem by assuming that the cost of a route is given by:

$$c_k = \begin{cases} \gamma_k & , \gamma_k \leq C_{max} \\ \lambda(\gamma_k, C_{max}) & , \gamma_k \leq C_{max} \end{cases} \tag{14}$$

Where λ represents a non-smooth function –e.g. a piece-wise function representing a variety of penalties.

5.3 The RandSHARP algorithm

For solving the non-smooth CARP we will use the RandSHARP algorithm introduced in **Chapter 3**. It is mainly composed of two parts: (a) the construction of an initial solution

using the classical heuristic; and (b) a biased randomization process applied to the construction of a random solution. To construct the random solution we apply a classical greedy heuristic. We use a classical heuristic as an starting point for several reasons. First of all, there are efficient heuristics for almost every combinatorial optimization problem. They usually are able to compute competitive solutions in a reasonably short amount of time. In addition, classical heuristics build solutions incrementally using well-tested strategies instead of directly using the objective function itself. With that, issues like non-convexity or non-smoothness of the objective function are not likely to have a significant impact on their efficiency. The main idea of these heuristics is to select the next step from a list of possible options of movements, usually following a selection criteria. In the case of the RandSHARP, the base heuristic is the SHARP base heuristic which is an adaption of the CWS for the CVRP, to the CARP.

5.4 Results

To evaluate the performance of the proposed algorithm, we have implemented it as a computer program. Java SE6 over Netbeans IDE was used to develop it for several reasons: (a) being an object-oriented programming language with advanced memory management features such as the garbage collection, and with readily-available data structure, it allows a somewhat faster development of algorithmic software; (b) it offers immediate portability to different platforms; and (c) it offers better replicability and duplicability than other languages.

However, a downside of using Java instead other languages such as C or C++ is the reduction on code execution performance, mainly due to the fact that Java is executed over a virtual machine and it is not a complied language and to the lack of pointer-based optimization. A standard personal computer was used to perform all tests, an Intel® Core2® Quad CPU Q9300 @2.50GHz and 8 GB RAM running the Windows® 7 Pro operating system. For the generation of random number we have employed the **L'Ecuyer (2006)** SSJ library for Java, concretely the LFSR113 random number generation, which offers a period value approximately equal to 2^{113}.

To assess the performance and the quality of the solutions obtained with the proposed algorithms, a complete dataset originally proposed for the standard CARP problem was adapted to make it have a non-smooth objective function. In concrete, the *gdb* (**Golden et al., 1983**) dataset was used. This dataset consists of 23 instances of small to medium size with a mixture of dense and sparse graph networks. For these instances, we have introduced parameter which determines the maximum route cost allowed. This parameter was defined considering the results obtained by the MIRH

algorithm for the CARP instance, rounded to a multiple of 10. But, instead of considering this parameter as a hard constraint, we have considered it as a soft one that could eventually be violated. Following the penalty costs function (14), for our tests we have used the specific non-linear and non-smooth function:

$$\lambda(\gamma_k, C_{max}) = \gamma_k + \min\{\theta(\gamma_k, C_{max}), 8\} \quad (15)$$

$$\theta(\gamma_k, C_{max}) = 0.5 + 100 \cdot \left(\frac{\gamma_k - C_{max}}{\gamma_k}\right)^4 \quad (16)$$

It is worth to mention that these non-smooth functions have been selected for the problem instances which are being tested in our experiments. These values depend on the magnitude of the costs of the instance. In this case the maximum route penalty due to exceeding C_{max} is equal to 8 (15), which approximately is the 10% of the average cost of a route, when considering the RandSHARP solution for the *gdb* CARP instances. The results obtained are displayed in **Table 13**. The table is structured in two halves: the first one showing the characteristics of every problem dataset, and the second one which contain the results. On the first half we display the instance name, the number of arcs (|E|) and nodes (|V|) in the problem instance, the vehicle capacity (W) and the maximum route costs parameter which we have defined. On the second half, the columns contain the following information: best-known solution for the original CARP problem instance; the solution obtained with RandSHARP when applied to the normal CARP problem instance (OBS-CARP) and the gap of this result with respect to the best-known solution; the solution obtained by RandSHARP in the non-smooth ARP when considering soft-constraint during the design phase of the algorithm (OBS-S) and its gap with respect the best-known solution of the CARP; and, finally, the solution of the RandSHARP algorithm when considering hard-constraints during the design phase (OBS-H) and its gap with respect the best-known solution of the CARP.

From the results we can notice first of all that RandSHARP has a good performance with the original CARP problem instance (without maximum route costs constraint and with a smooth objective function). In addition, as it considers soft-constraints during the design phase of the routes, it is able to minimize the effect of having a non-smooth objective function, showing a result closest to the BKS and to the solution obtained by the same algorithm when considering only the CARP. Additionally, we can also notice that when considering hard-constraints in the design of the RandSHARP algorithm, the performance falls down dramatically. This is due to the fact that considering hard-constraints makes the solution to have more routes required,

which means that more overload is obtained in the solution due to the round trips to the depot for refilling.

Remark that the gap with respect to the best known solution showed in the table is computed as follows:

$$Gap(C_{RandSHARP}, C_{BKS}) = 100 \cdot \left(\frac{C_{RandSHARP} - C_{BKS}}{C_{BKS}}\right) \quad (17)$$

Set	|E|	|V|	W	Max. route costs	BKS (1)	OBS-CARP (2)	Gap (1)-(2)	OBS-S (3)	Gap (1)-(3)	OBS-H (4)	Gap (1)-(4)
gdb1	12	22	5	60	316	316	0.00%	317.87	0.59%	343.02	8.55%
gdb2	12	26	5	50	339	339	0.00%	341.76	0.81%	422.56	24.65%
gdb3	12	22	5	50	275	275	0.00%	276.60	0.58%	340.09	23.67%
gdb4	11	19	5	50	287	287	0.00%	289.08	0.72%	483.71	68.54%
gdb5	13	26	5	60	377	377	0.00%	378.85	0.49%	467.00	23.87%
gdb6	12	22	5	60	298	298	0.00%	299.08	0.36%	351.51	17.96%
gdb7	12	22	5	60	325	325	0.00%	325.74	0.23%	356.50	9.69%
gdb8	27	46	27	30	348	350	0.57%	359.43	3.28%	594.91	70.95%
gdb9	27	51	27	30	303	313	3.30%	318.75	5.20%	433.71	43.14%
gdb10	12	25	10	60	275	275	0.00%	276.53	0.56%	283.50	3.09%
gdb11	22	45	50	80	395	395	0.00%	400.01	1.27%	409.00	3.54%
gdb12	13	23	35	60	458	468	2.18%	464.89	1.50%	739.19	61.40%
gdb13	10	28	41	80	536	536	0.00%	545.00	1.68%	580.70	8.34%
gdb14	7	21	21	60	100	100	0.00%	104.00	4.00%	104.00	4.00%
gdb15	7	21	37	50	58	58	0.00%	58.00	0.00%	58.00	0.00%
gdb16	8	28	24	30	127	127	0.00%	127.76	0.60%	129.00	1.57%
gdb17	8	28	41	20	91	91	0.00%	91.00	0.00%	91.00	0.00%
gdb18	9	36	37	30	164	164	0.00%	167.24	1.98%	182.00	10.98%
gdb19	11	11	27	20	55	55	0.00%	55.50	0.91%	63.00	14.55%
gdb20	11	22	27	30	121	121	0.00%	121.53	0.44%	123.00	1.65%
gdb21	11	33	27	30	156	156	0.00%	158.00	1.28%	158.00	1.28%
gdb22	11	44	27	30	200	200	0.00%	201.00	0.50%	202.00	1.00%
gdb23	11	55	27	30	233	233	0.00%	235.00	0.86%	235.00	0.86%
Avg.	*13*	*29*	*12*	*77*			*0.26%*		*1.21%*		*17.23%*

Table 13. Evaluated *gdb* instances and obtained results.

5.5 Chapter Conclusions

In this chapter an overview of non-convex and non-smooth optimization problems has been presented. We have also discussed how different approaches have been used in different non-smooth and non-convex problems in the existing literature. Among others we can find the GRASP, HBSS or Tabu Search. As has been pointed out, our methodology has similarities with some methods already reported in the literature but, at the same time, maintains significant different as previously discussed. In addition, we have defined the non-smooth Arc Routing Problem and described the objective function characteristics which make our approach a good candidate for solving it. We have also presented how the RandSHARP, an algorithm based on the MIRHA framework for solving the problem, can be used for solving the non-smooth Arc Routing

Problem. Finally, we evaluated the performance by using some of the benchmarks available on the CARP literature, but considering the capacity constraint as a soft constraint which could be violated by incurring in some penalty. Results show that the biased-randomized nature of the MIRHA framework, make the algorithm robust for scenarios like this, where the optimized function is non-smooth.

6 Facility Location Problem

Parts of this chapter have been taken from the co-autored publications: **Cabrera et al. (2014)** and **Gonzalez-Martin et al. (2014d)**.

As a closure of the research done within this thesis, another problem has been studied which also has direct applications on the Telecommunications field. The Facility Location Problem (FLP) involves locating an undetermined number of facilities to minimize the sum of the setup costs and the costs of serving customer from these facilities. The problem assumes that the alternative sites where the facilities can be located are predetermined and the demand in each customer is known beforehand. Facility location decisions are costly and difficult to reverse as, once the facility has been installed, the associated cost of opening is actually incurred. This problem is useful to model problems in very disparate areas like transportation and logistics, inventory planning or telecommunication network or computational infrastructures planning. Clear examples of its application on Information technologies are the placement of web-servers in a distributed network which have to provide some service to a given set of customers; or the placement of cabinets in optical fiber networks to server all the customer with a minimal network deployment cost.

We can find some similarities between the FLP and ARP problem families. In the case of the ARP, we can see the goal as deciding how to group the customers in subsets which should be interconnected in a sequential route. On the other hand, for the FLP these customer's subsets are connected to some kind of central node (facility) and the goal is, in addition to determine the subsets of customers which are grouped, determine also the location of this central nodes. One example of application of the FLP to a real scenario is that in which several servers are available for providing a kind of internet service, and the goal is to find the subset of servers to be connected for providing the service to all the customers with the minimum cost possible.

6.1 Literature Review

The FLP was introduced in Operations Research field in the early 60's **(Balinski , 1966, Stollsteimer, 1961)**, originally referred to as Plant Location Problem. This is perhaps the most common location problem, having been widely studied in the literature, both in theory and in practice. In this section we will review some solutions proposed to the problem, as well as different variants of the FLP proposed to cope with different scenarios. For a more extensive literature review on this topic, refer to **Drezner (1995)** or **Fotakis (2011)**.

6.1.1 Solutions to the problem

The facility location problem has been studied from the perspectives of worst case analysis, probabilistic analysis, polyhedral combinatory and empirical heuristics (**Cornuejols et al., 1990**). In the existing literature, we can also find exact algorithms for the problem, but its NP-hard nature makes heuristics a more suitable tool to address larger instances. One of the first works on the FLP was a branch-and-bound algorithm developed by **Efroymson and Ray (1966)**. They used a compact formulation of FLP to take advantage of the fact that its linear programming relaxation can be solved by inspection. However, this linear programming relaxation does not provide tight lower bounds to the problem. The model is therefore known as a weak formulation. Another of the earliest approaches proposed for the problem is the direct search or implicit enumeration method proposed by **Spielberg (1969)**. The author defined two different algorithms based on the same directed search, one considering the facilities initially open and a second one considering the facilities initially closed.

Schrage (1975) presented a tight linear programming formulation for the location problem different from the one defined by **Efroymson and Ray (1966)**. Schrage applied to this formulation a specialized linear programming algorithm for variable upper bound constraints. **Erlenkotter (1978)** presented a dual-based procedure that begins with this tight linear programming formulation but differed from previous approaches by considering a dual objective function. **Körkel (1989)** presented an improved version of the original Erlenkotter algorithm.

One of the first approximation algorithms for the problem was the greedy algorithm proposed by **Hochbaum (1982)**. The first constant factor approximation for this problem was given by **Shmoys et al. (1997)**, later improved by **Chudak (1998)**, being both of these algorithms based on LP-rounding and therefore having high running times. **Jain and Vazirani (1999)** proposed a primal-dual algorithm with faster running times and adapted for solving several related problems. This same algorithm

was later enhanced in **Jain et al. (2003)** and obtained better results. More recently, **Li (2013)** proposed an improved approximation algorithm that outperformed the former results.

Approximation algorithms are very valuable for a theoretical analysis of the problem. However, these algorithms are outperformed in practice by more straightforward heuristic with no performance guarantees when facing more complex problem instances. Constructive algorithms and local search methods for this problem have been used for decades, starting from **Kuehn and Hamburger (1963)**. The authors presented one of the earliest models for the problem and a heuristic procedure solving it. Their heuristic comprised two main phases, first a constructive phase considered as the main program, and an improvement phase.

Following this work, more sophisticated metaheuristics have been applied to the FLP. **Alves and Almeida (1992)** proposed a Simulated Annealing algorithm that was one of the firsts metaheuristics applied to the problem. **Kraticaet al. (2001)** presented a genetic algorithm outperforming previous works. **Ghosh (2001)** presented a neighborhood search heuristics for the problem, using tabu search as local search and obtaining competitive solutions in very low computational times compared to exact algorithms. **Michel and Van Hentenryck (2003)** defined a simple tabu search algorithm, which demonstrated to be robust, efficient and competitive when compared with the previous work with genetic algorithms. The tabu search algorithm used a linear neighborhood, which flipped a single facility at each iteration. **Resende and Werneck (2006)** proposed an algorithm based on the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic. The algorithm combined a greedy construction phase with a local search procedure. It obtained results very close to the best-known solution for a wide range of different instance sets. More recently **Lai et al. (2010)** presented a hybrid algorithm based on Benders' decomposition algorithm and using a genetic algorithm instead of the costly branch-and-bound method, to obtain good suboptimal solutions. The computational results indicated that the algorithm was effective and efficient. However the author only compared the performance with the Benders' original algorithm.

Finally, some work has been presented regarding parallel computing. **Wang et al. (2008)** presented an adaptive version of a parallel Multi population Particle Swarm Optimization (MPSO) implemented with OpenMP. The implementation obtained an important improvement in terms of execution times while obtaining competitive results with a standard computer.

6.1.2 Problem variations

Cooper (1963) studied the problem of deciding locations of warehouses and allocation of customers demand given the locations and demands of customers, which can be considered as the basic facility location problem. After that, many variations of the basic facility location problem have been studied. The first variation defined is by adding a capacity constraint to each of the facilities in the problem, which results in the Capacitated Facility Location Problem (CFLP).

Another immediate generalization of the original FLP is the problem where the delivery of different products is considered. The Multi Commodity Facility Location Problem (MCFLP) was first studied by **Klincewicz and Liss (1987)**, and it studied the problem without any restriction on the number of products at each facility. In the Facility Location with General Cost Function (FLP-GCF), the facility cost is a function on the number of clients assigned to the facility. An additional variant for the problem is this in which the demand points arrive one at a time and the goal is to maintain a set of facilities to service these customers. This is called the Online Facility Location (OFLP) **(Meyerson, 2001)**. **Carrizosa et al. (2012)** present a nonlinear variation of the problem. In it they modified the classical Integer Programming formulation of the problem by adding to the cost a nonlinear function depending on the number of open facilities. This was referred to as the Nonlinear Minsum Facility Location Problem (NMFLP).

An interesting field of study of variations to this problem are those proposals defined under uncertainty **(Snyder, 2006)**, introducing wide variations on any of the parameters of the problem (mainly cost, demands or distances). The goal in these problems is to find a solution that performs well under any possible realization of the random parameters, which means a robust solution. The random parameters can be either continuous or discrete. As an example, **Balachandran and Jain (1976)** presented a CFLP model with piecewise linear production costs that need not be either concave or convex. Demands are random and continuous, described by some joint probability distribution. In this kind of problems, only first-stage decisions are available, so there are no recourse decisions. So, once the locations are set, they cannot be changed after the uncertainty is resolved. The objectives therefore include the expected recourse costs.

Finally, it is worth to mention an extension of the FLP where it is combined with another optimization problem, the Steiner Tree Problem (STP). As a result, **Karger and Minkoff (2000)** defined the Connected Facility Location Problem (ConFLP). The ConFLP introduces an additional constraint to the problem, which is that a Steiner tree

must connect all the open facilities. This variation of the problem is very interesting because it combines location and connectivity problems, which is suitable to model different network design problems.

6.1.3 Applications

Recently the FLP problem found several new applications in digital network design problems. One example is the equipment allocation in Video on Demand (VoD) network deployments **(Thouin and Coates, 2008)**. VoD services are complex and resource demanding, so deployments involve careful design of many mechanisms where content attributes and usage should be taken into account. The high bandwidth requirements motivate distributed architectures with replication of content. An important and complicated task part of the network planning phase of these distributed architectures is resource allocation. The growth of peer-to-peer networks and the use of mobile devices for accessing the contents have made the problem even more complex. Another example of application can be found in **Lee and Murray (2010)**. In this chapter the authors introduce an approach for survivable network design of citywide wireless broadband based on the FLP model. They address two issues: how to locate the Wi-Fi equipment to maximally cover the given demand; and how to connect Wi-Fi equipment to ensure survivable networking on a real case scenario in the city of Dublin (Ohio).

Maric (2013) applied the problem to model the location of long-term health care facilities among given potential sites. The objective is to minimize the maximal number of patients assigned to the established facilities. Examples can be also found in the supply chain management area. **Brahimi and Khan (2013)** show a real case of a company in Pakistan which wanted to outsource part of its warehousing activity to a third party provider. The problem was to decide where to rent space in the third party warehouses.

The Online Facility Location Problem **(OFLP, Meyerson, 2001)** can model a network design problem in which several servers need to be purchased and each client has to be connected to one of the servers. Once the network has been constructed, additional clients may need to be added to the network. In this case additional costs will appear into the problem such as the connection cost of connecting a new customer to the cluster and, if additional capacity is required to accommodate the increase of demand, an additional server should be purchased (which means opening an additional facility).

6.2 Problem Description

The (uncapacitated) FLP involves locating an undetermined number of facilities to minimize the sum of setup and serving costs and was first described in **Balinski (1966)** and **Stollsteimer (1961)**. The problem is defined over an undirected strongly connected graph $G = (V, E)$ where *V* is composed of a subset of customers $C \subseteq V$ and a subset of facilities $F \subseteq V$, and *E* is a set of edges connecting the nodes in *V*. Each edge $e \in E$ has an associated cost of using it $c_e \geq 0$, and for all $i \in F$ we are given a facility opening cost $f_i \geq 0$. Furthermore, for every facility i and customer j we have an associated cost of connecting the customer to the facility $c_{ij} \geq 0, i \in F, j \in C$. Under these circumstances, the objective of the problem is to open a subset of the facilities in F and connect each customer with an open facility, so that the total cost is minimized:

$$\min \sum_{j \in C} c_{i(j)j} + \sum_{i \in F} f_i \tag{18}$$

The uncapacitated FLP is considered as the "simple" facility location problem **(Verter, 2011)**, where both the alternative facility locations and the customer positions are considered discrete points in the graph. An example of a FLP problem instance can be found in **Figure 18**. This assumes that the alternative sites have been predetermined and the demand in each customer zone is concentrated at the point representing that region. FLP focuses on the production and distribution of a single commodity over a single time period, during which the demand is assumed to be known with certainty. The distinguishing feature of this basic discrete location problem, however, is the decision maker's ability to determine the size of each facility without any restriction.

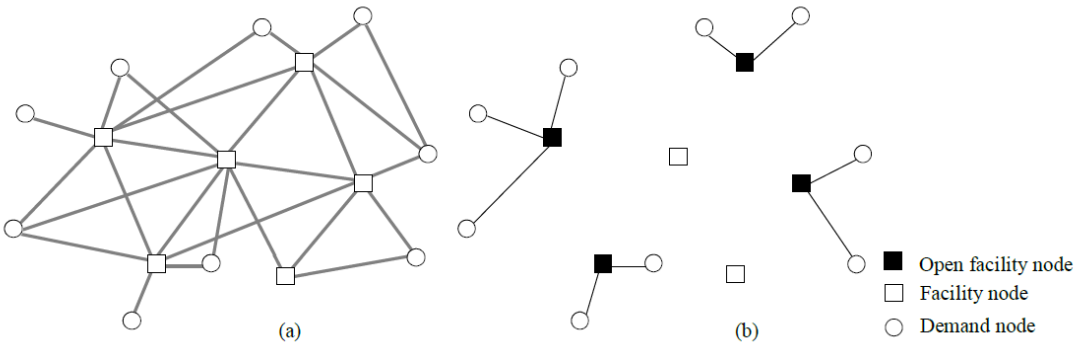

Figure 18. Example of FLP problem instance (a) and solution (b)

6.2.1 Basic notation

When we work with an instance of the FLP, we assume the notation F, C, f_i, c_{ij} as described before. Every customer has an associated demand that needs to be served by the facility, d_j. Notice that this demand would take place in the problem definition for

the case of the Capacitated Facility Location Problem (CFLP), where every facility has a limited facility, so it is required a constraint to avoid surpassing that facility capacitated. Even without this capacity constraint, the FLP is proved to be NP-hard **(Cornuejols et al., 1990)**. For a given instance and a given non-empty subset of facilities $X \subseteq F$, a best assignment $\sigma: C \to X$ satisfying (19) can be computed easily. Therefore, we will often call a nonempty $X \subseteq F$ a feasible solution, with facility cost $c_F(X)$ (20) and service cost $c_S(X)$ (21). The task is to find a nonempty subset $X \subseteq F$ such that the sum of facility cost and service cost is minimized. We denote the optimum by OPT.

$$c_{\sigma(j)j} = \min_{i \in X} c_{ij} \tag{19}$$

$$C_F(X) = \sum_{i \in X} f_i \tag{20}$$

$$C_S(X) = \sum_{j \in C} \min_{i \in X} c_{ij} \tag{21}$$

6.2.2 Mathematical model

The FLP can also be formulated as an Integer Program (IP) as follows **(Vygen, 2009)**:

$$\min \sum_{i \in F} f_i \cdot y_i + \sum_{i \in F} \sum_{j \in C} c_{ij} \cdot x_{ij} \tag{22}$$

Subject to:

$$x_{ij} \leq y_i, \forall i \in F, j \in C \tag{23}$$

$$\sum_{i \in F} x_{ij} = d_j, \forall j \in C \tag{24}$$

$$x_{ij} > 0, \forall i \in F, j \in C \tag{25}$$

$$y_i \in \{0,1\}, \forall i \in F \tag{26}$$

This formulation is considering two decision variables x_{ij} and y_i. x_{ij} represents the amount of flow from a facility *i* to a customer *j*, which would be *0* if the customer will not be served by that facility, or equals to the demand of the customer otherwise. In addition, y_i is a decision variable which is equal to *1* if the facility *i* will be opened and *0* otherwise. Constraint (23) forces that customers can only be assigned to open facilities. And constraint (24) assures that every customer will have its demand satisfied. Note that this formulation has both a binary (y_i) decision variable and a

continuous (x_{ij}) variable. In this case the formulation is called Mixed Integer Linear Program (MILP).

6.3 The RandCFH-ILS Algorithm

For the FLP we are defining an algorithm based on the MIRHA framework (see **Chapter 2**). The proposed algorithm is detailed on **Figure 19** as a flow chart and works as follows. First, it loads the problem instance. Then, it generates an initial random solution, which is the starting point for the ILS procedure. To generate this initial solution, the algorithm chooses a random number p_i between $|F|/2$ and $|F|$, and picks randomly p_i facilities to open. After that, it computes the total cost of the generated solution, which is selected as starting point.

The basis for selecting always more than the half of facilities to open in the initial solution is not casual. Regarding computational costs, the fact of closing a facility on the solution is cheaper than opening a solution. This is due to the way that the costs are calculated. When a facility is selected to be closed, for updating the solution costs we only have to relocate the customers that were assigned to the closed facilities, but the rest of customers assigned to the other facilities remain unmodified. However, when opening a facility in the solution, every single client to facility assignation must be evaluated to determine if the newly open facility is the one with lesser assignment cost for this customer among all the open facilities in the current solution. For this reason, if starting with a solution with a higher number of open facilities, chances are that for improving the solution, closing less costly movements should be performed.

After this initial solution is generated, a local search procedure is applied to refine the initial solution. We propose two different local search procedures in this chapter. The first one is simple and very fast (tiny local search), while the other performs a deeper search with slower execution times (deep local search).

On the one hand, the tiny local search procedure is based only in closing movements. It starts from the current solution and randomly closes one by one each of the open facilities. The facilities to close are selected randomly among all the closed facilities. If the solution is improved (i.e. it has a lower total cost) by closing the selected facility, it is actually removed from the solution; otherwise the facility is kept in.

On the other hand, the deep local search procedure combines both closing and opening movements, which makes this procedure computationally more expensive. This local search procedure is divided in three parts. First, it starts by randomly opening one by one each of the closed facilities. It evaluates the solution at each stage

and only adds the open facility to the solution in the case an improvement is obtained. On the second stage it performs a swap movement, replacing a random number of open facilities on the solution by the same number of closed facilities. Finally, in the third stage, it closes the open facilities in the current solution one by one selected in a random order, effectively closing the facility in the final solution only in the case an improvement is obtained. Further details on both local search procedures are detailed on next sections.

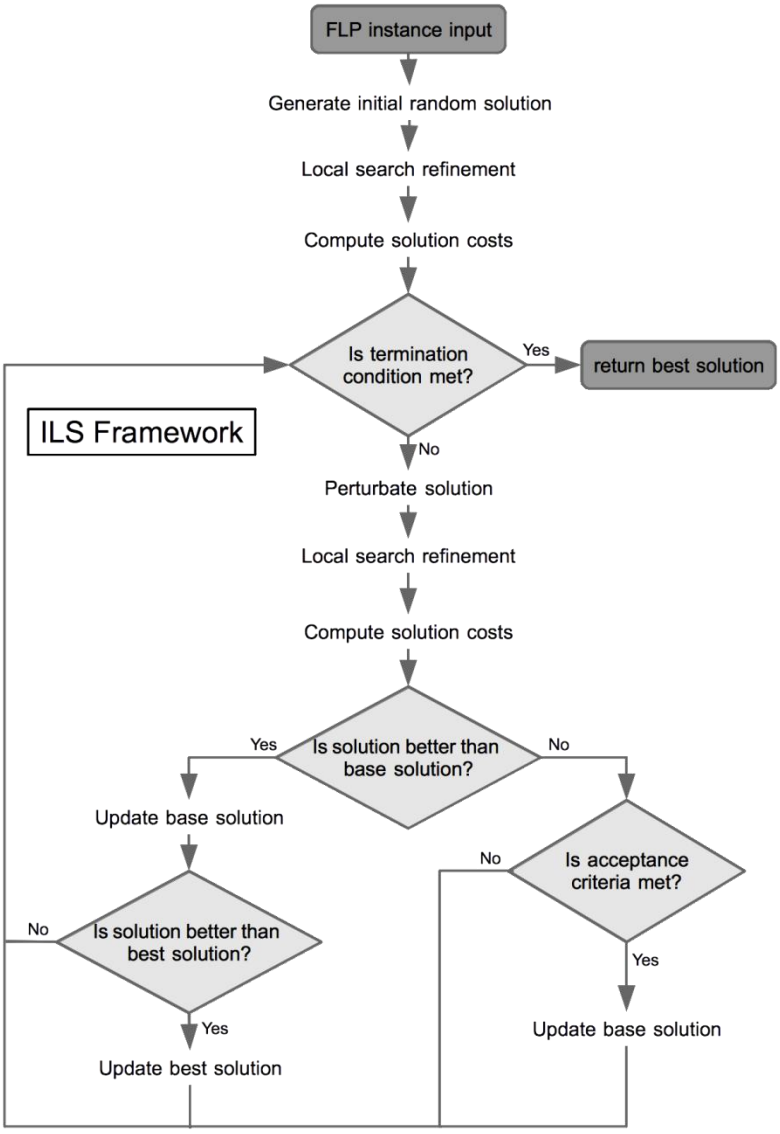

Figure 19. Flow diagram of the proposed approach (RandCFH-ILS)

The solution generated will finally be the starting point for the ILS framework and considered as base solution. The ILS is mainly an iterative process which, at each

iteration, generates a new feasible solution with chances to outperform the base solution. Our ILS consists of three steps:

1. Destruction/Construction of the solution (perturbation)
2. Refine the solution (local search)
3. Acceptance criteria of the solution

On the first step, a perturbation operator is applied to the solution. This operator basically destructs some part of the solution by removing open facilities, and then reconstructs it by opening new facilities. This operator always opens more facilities than the amount of closed ones, so it benefits from the fact that the closing movement is less computationally expensive, as happened with the generation of the initial solution. This solution is then refined by the same local search operator used on the initial solution.

Finally, the last step of the ILS is the acceptance criteria for updating the best and base solutions. Being the best solution the one that will be returned as the result of our methodology and the base solution the solution used as initial solution for the next iteration of the ILS. If the solution obtained from the perturbation and local search procedures improves the best solution found so far, then the best and base solutions are updated.

Additionally, if the solution obtained is worse than the current base solution, an acceptance criterion is defined. This acceptance criterion allows a non-improving solution to be accepted as a new base solution if certain conditions are met. The acceptance criteria define a gap which the solution is allowed to worsen. This gap varies during the execution of the algorithm, allowing greater gaps at the beginning of the execution, and smaller gap as the execution time passes. With that, we enable a method to escape from local minima and explore different regions of the solutions space.

6.3.1 Implementation details

The main method of our proposal is the **RandCFH-ILS** **(Figure 20)**. This method receives as a parameter the facilities and customers which conform the problem instance, the beta parameter used during the randomization process and a parameter used as stopping criterion on the ILS loop.

It first creates an initial random solution by the **genInitRandSol** method and refines it by the local search procedure. As early mentioned in this chapter, we defined two different local search procedures: **localSearchTiny** and **localSearchDeep**. After that, credit values used by the acceptance criteria are initialized. Then, the ILS loop is

started using the base solution obtained at this point. In the iterated loop, the perturbation operator is applied to the current base solution (calling the **perturbate** method) and the obtained solution is refined by the local search procedure.

```
procedure RandCFH-ILS(facilities, clients, beta, maxIter)
01    baseSol ← getInitRandSol(facilities, clients, beta)
02    baseSol ← localSearch(baseSol)
03    bestSol ← baseSol
04    nIter ← 0
05    credit ← 0
06    while {each nIter <= maxIter} do
07      newSol ← perturbate(baseSol, beta) % destruction-construction
08      newSol ← localSearch(baseSol)
09      delta ← cost(newSol) – cost(baseSol)
10      if {delta < 0} then
11        credit ← -(delta)
12        baseSol ← newSol
13        if {cost(newSol) < cost(bestSol)} then
14          bestSol ← newSol
15        end if
16      else if {delta > 0} and {credit >= delta} then %acc. criterion
17        credit ← 0
18        baseSol ← newSol
19      end if
20      nIter ← nIter + 1
21    end while
22    return bestSol % returns the best(s) solution(s) found so far
end procedure
```

Figure 20. RandCFH-ILS algorithm main procedure

This newly obtained solution is then evaluated. If the solution improves the current base solution, then the base solution is updated. Also the credit is updated with the same value as the improvement obtained. This causes credit values to be higher at the beginning of the execution, so bad solutions are more tolerated. As the algorithm obtains smaller improvements, only small degradations on the solution are welcomed by the acceptance criteria. If the new solution improves also the best solution, we update it as well.

In the case that the new solution is worse than the current base solution, the difference between both is still evaluated. The new solution is only accepted as new base solution if this difference is below the credit acceptance threshold. In that case, the new solution is accepted as base solution and the credit is reset to 0, so two degradations in a row are never allowed.

The loop is executed until the termination criterion is met, which can be a time limit or a maximum number of iterations. To conclude, the algorithm returns the best solution found so far.

The **genInitRandSol** **(Figure 21)** method is the responsible of generating the initial solution which serves as starting point for the algorithm. This method receives as parameters the facilities and customers in the problem instance and the beta parameter required for the Random Number Generator (RNG).

```
procedure getInitRandSol(facilities, clients, beta)
% Since closing is fast, start by opening a lot of facilities
01    nFacilToOpen ← rand(size(facilities)/2, size(facilities))
% Biased-randomized selection process using a Geometric(beta) promotes
% the random selection of those facilities with HIGHEST density levels
02    facilToOpen ← biasRandSelect(facilities, nFacilToOpen, beta)
03    sol ← constructSol(facilToOpen, clients) % compute costs
03    return sol
end procedure
```

Figure 21. getInitRandSol method

First, it generates a random number between to total number of facilities and the half of this number. Next, it randomly picks this number of facilities from the list of available ones in the problem instance to be open. Finally it constructs the solution with the selected facilities as open facilities and assigns each customer to that facility among the open with lowest service cost for it.

```
procedure perturbate(baseSol, beta)
01    sol ← copy(baseSol)
02    openFacilities ← getOpenFacilities(sol)
03    nFacilToClose ← rand(0, size(openFacilities))
% Biased-randomized selection process using a Geometric(beta) promotes
% the random selection of those facilities with LOWEST density levels
04    invOpenFacil ← inverseOrder(openFacilities)
05    facilToClose ← biasRandSelect(invOpenFacil, nFacilToClose, beta)
06    sol ← destructSol(sol, facilToClose)
07    closedFacil ← getClosedFacilities(sol)
% Re-construct more than destruct since closeLS is faster than openLS
08    nFacilToOpen ← rand(nFacilToClose, size(closedFacil))
% Biased-randomized selection process using a Geometric(beta) promotes
% the random selection of those facilities with LOWEST density levels
09    facilToOpen ← biasRandSelect(closedFacil, nFacilToOpen, beta)
10    sol ← constructSol(facilToOpen) % Compute assignment cost
11    return sol
end procedure
```

Figure 22. perturbate method

The **perturbate** **(Figure 22)** method complete the perturbation operator used within the main ILS loop. This method receives a base solution and the beta parameter for the RNG. First of all, it creates a copy of the base solution and extracts the list of open facilities from it. Then, it randomly closes a random number of facilities to close from all the open ones in the solution. After that, it generates another random number of facilities to be opened, being always greater than the number of facilities previously

closed. This forces the new solution to always include more open facilities and benefits the algorithm on the later refinement process of the fact that closing a facility is a cheaper operation than opening a new facility. Once we have the number of facilities to open, we randomly pick the right amount from the closed facilities list and reconstruct the solution by adding them.

```
procedure localSearchTiny(baseSol, beta)
% Fast local search based on closing
01    sol ← copy(baseSol)
02    openFacilities ← getOpenFacilities(sol)
03    openFacilitiesSorted <- biasRandSort(openFacilities)
04    for each {oFacility in openFacilities} do
05      newSol ← deleteFacility(sol, oFacility)
06      if {cost(newSol) < cost(sol)} then
07        sol ← newSol
08      else
09        newSol ← addFacility(sol, oFacility)
10      end if
11    end for
12    return sol
end procedure
```

Figure 23. localSearchTiny algorithm main procedure

The **localSearchTiny** **(Figure 23)** method describes the operation of the tiny local search, which can be used to refine FLP solutions. This procedure receives as parameters the base solution and the beta parameter for the RNG.

First of all it, creates a copy of the base solution, extracts the list of open facilities and sorts it randomly. After that, all open facilities are removed from the solution one at a time. If the solution without that facility has a lower global cost than the one including it, the facility is effectively removed from the solution. Otherwise, it is kept in the open facilities list.

The **localSearchDeep** **(Figure 24)** method presents the functioning of the deep local search, which can be used to refine FLP solutions as the **localSearchTiny** method. This procedure receives as parameters the base solution and the parameter for the RNG. It starts by creating a copy of the base solution. Then, it starts a loop structured on three different blocks that will keep running while an improvement is found for the solution.

On the first block, we try to open closed facilities. The list of closed facilities is extracted from the current solution and is randomly sorted. All closed facilities are added one by one to the solution. If the solution with that facility has a lower global cost than the one without it, the facility is effectively added the solution and the improvement control is set to true. Otherwise, it is kept out.

The second block of the loop swaps open and closed facilities. It swaps all the open facilities with all the closed ones one at a time until an improvement is found. If any of the solutions with a swap has a lower global cost than the one without the change, the facilities are effectively swapped from the solution, the improvement control is set to true and the swapping process ends.

```
procedure localSearchDeep(baseSol, beta)
% Local search with exhaustive search
01    sol <- copy(baseSol)
02    while {improvement} do
03      improvement ← false
04      closedFacilities ← getClosedFacilities(sol)
05      closedFacilitiesSorted ← biasRandSort(closedFacilities, beta)
06      for each {cFacility in closedFacilitiesSorted} do
07        newSol ← addFacility(sol, cFacility)
08        if {cost(newSol) < cost(sol)} then
09          sol <- newSol
10          improvement ← true
11        else
12          newSol ← deleteFacility(sol, cFacility)
13        end if
14      end for
15      openFacilities ← getOpenFacilities(sol)
16      for each {oFacility in openFacilities} do
17        newSol ← deleteFacility(sol, oFacility)
18        closedFacilities ← getClosedFacilities(sol)
19        for each {cFacility in closedFacilities} do
20          newSol ← addFacility(sol, cFacility)
21          if {cost(newSol) < cost(sol)} then
22            sol ← newSol
23            improvement ← true
24            break
25          else
26            newSol ← removeFacility(newSol, oFacility)
27          end if
28        end for
29        if {cost(newSol) < cost(sol)} then
30          sol ← newSol
31          improvement ← true
32          break
33        else
34          newSol ← addFacility(newSol, cFacility)
35        end if
36      end for
37      sol ← localSearchTiny(sol, beta)
38    end while
39    return bestSol % returns the best(s) solution(s) found so far
end procedure
```

Figure 24. localSearchDeep algorithm main procedure

Finally in the third block, we try to remove open facilities calling the **localSearchTiny**, willing to reduce the solution cost. All open facilities are removed from the solution one at a time. If the solution without that facility has a lower global cost than the one

including that facility, the facility is effectively removed from the solution and the improvement control is set to true. Otherwise, it is kept in the open facilities list.

6.4 Results

To evaluate and assess the performance of the proposed algorithm, several computational experiments were performed. The proposed algorithm was implemented as a Java® 7SE application. We performed all the tests on a commodity desktop computer with an Intel® CoreTM i5-2400 at 3.20 GHz and 4 GB RAM running Ubuntu GNU/Linux 13.04.

Even though Java is a programming language executed in a virtual machine (JVM) and we are aware it may show poorer performance than others like C or C++, the vast amount of tools available in the standard API (such as advanced structures or garbage collection) and its object-orientation eased the development process. In addition, the execution on the JVM offers better replicability and repeatability than other languages.

The implementation process of the algorithm is not a trivial task, since there are some details which require special attention: (i) The correct design of the different classes so that a convenient level of coupling and cohesion is reached; (ii) The quality of the Random Number Generator, which affects directly in the performance of our algorithm; (iii) The level of precision used to store and operate with numerical values is key for the effectiveness of the algorithm. In order to fix (ii), a state-of-the-art pseudo random number generator has been employed. Specifically, we used the LFSR113 from the SSJ library created by **L'Ecuyer (2002)**. This generator provides a period of 2^{113}, compared to the period of 2^{48} of the generator provided by the standard Java library. Also, to benefit from the fact that the used computer had a 4-core processor, the implementation of the algorithm was done in a multi-threaded application, so at the same time we were able to execute up to four different instances in parallel. With a relatively small change on the implementation, this multi-threaded implementation could be adapted to work all the threads on the resolution of the same instances, so the computational times for obtaining solutions could be reduced.

To test the efficiency of the proposed algorithm, four different classes of problem instances obtained from **Hoefer (2014)** were used. The selected datasets were chosen with the criteria of testing the algorithm against instances of small, medium and big size (in terms of facilities and customers included in the graph). We

briefly describe next the used sets, but the reader is referred to **Hoefer (2014)** for further details of each class of instances.

- ⇒ **BK**: a small-sized class introduced by **Bilde and Krarup (1977)**. Includes 220 instances in total divided in 22 subsets, with the number of facilities varying from 30 to 50 and the number of customers from 80 to 100. These instances were artificially generated by the authors, selecting the assignment costs randomly on the range [0, 1000], and opening costs being always greater than 1000.
- ⇒ **GAP**: a medium-sized class also introduced by **Kochetov and Ivanenko (2003)**. Consists of three subsets, each with 30 instances: GAPA, GAPB and GAPC, being GAPC the hardest. These instances are considered to be especially hard for dual-based methods.
- ⇒ **FPP**: a medium-sized class introduced by **Kochetov and Ivanenko (2003)**. Consists of two subsets, each with 40 instances: FPP11 and FPP 17. Although optimal solutions in this class can be found in polynomial times, the instances are hard for algorithms based on flip and swap local search, since each instance has a large number of strong local optima.
- ⇒ **MED**: big-sized class originally proposed for the p-median problem by **Ahn et al. (1998)**, and later used in the context of the uncapacitated FLP by **Barahona and Chudak (1999)**. Each instance is a set of n points picked uniformly at random in the unit square. A point represents both a user and a facility, and the corresponding Euclidean distance determines connection costs. The set consists of six different subsets with a different number of facilities and customers (500, 1000, 1500, 2000, 2500 and 3000) and three different opening cost schemas for each subset.

In the performed experiments, we tested the algorithm using the two different local search procedures presented in this chapter. So, each dataset was solved twice: one with the algorithm using the **localSearchTiny** procedure, and the second one using the **localSearchDeep** procedure. With this we will be able to compare both procedures, being able to determine which local search has the better performance depending on the characteristics of the solved instance.

The experiments were run by setting a stopping criteria based on a time limit for every problem instance. So every instance will be solved by generating feasible solution until the termination criteria of time has been met (see **Figure 19**). These termination criteria will be different depending on the size and the class of instances being solved. So, the greater the instances are, the more time it is given to the

algorithm (see **Table 14**). We have set these times the great enough to guarantee that the algorithm reaches its best solution. We record the time when the best solution was found for every instance, so we can know how fast the convergence to this best solution was.

Class	Number of facilities (*n*)	Time (secs.)
BK	80-100	30
FPP11	133	600
FPP17	307	600
GAPA	100	180
GAPB	100	180
GAPC	100	180
MED	500-3000	3600

Table 14. Termination criteria for every class of instances.

Next, we show the results obtained when executing our algorithm in the earlier explained benchmarks. We compare our results using both of the local search methods described with the ones declared by **Resende and Werneck (2006)** using a GRASP algorithm, as it is the best performing algorithm available in the literature. We would like to remark that our experiments were performed on a low-end commodity desktop computer, as opposed to the supercomputer used by Resende and Werneck; also, we used a standard Java SE application, instead of the specifically compiled application utilized by Resende and Werneck. Although the theoretically higher performance of our computer, we only used one of its cores for the Java Virtual Machine to run the experiments.

We started evaluating the methodology in the set of tests with simplest and smallest problem instances, the **Bilde and Krarup (1977)** benchmark **(Table 15)**. In this test, our algorithm clearly outperforms the GRASP proposal when using the deep local search method. We find the optimal solution for all the instances in the benchmark in much shorter execution times. Contrarily, we discovered in this first test that the tiny local search method obtains lower quality results in terms of gap with the optima in comparable times with the GRASP.

We performed the next evaluation over the GAP benchmark **(Table 16)**, known as a hard test for dual-based oriented methods. Neither our algorithm nor GRASP was thought specifically for dual-based problems, so finding good quality results in these instances is challenging. In this set of tests, our algorithm obtains better quality solutions using any of the presented local search methods, although the deep one

results in much lower gaps from optima. However, the running times employed by our proposal were much higher than the ones in Resende's experimentation.

			GRASP		RandCFH-ILS *tiny*		RandCFH-ILS *deep*	
Subset	**# customers**	**# facilities**	**Gap**	**t (ms)**	**Gap**	**t (ms)**	**Gap**	**t (ms)**
B	100	50	0.000	310	0.000	794	0.000	0.20
C			0.016	450	0.130	835	0.000	0.05
D01	80	30	0.000	223	0.001	116	0.000	0.02
D02			0.000	211	0.000	317	0.000	0.04
D03			0.000	199	0.000	58	0.000	0.06
D04			0.000	170	0.000	48	0.000	0.08
D05			0.000	162	0.000	41	0.000	0.10
D06			0.000	186	0.000	124	0.000	0.12
D07			0.000	174	0.000	9	0.000	0.15
D08			0.000	166	0.000	20	0.000	0.08
D09			0.000	175	0.000	11	0.000	0.07
D10			0.000	166	0.000	28	0.000	0.21
E01	100	50	0.000	476	0.201	667	0.000	0.03
E02			0.000	588	0.011	1176	0.000	0.07
E03			0.019	512	0.000	286	0.000	0.11
E04			0.000	464	0.000	389	0.000	0.14
E05			0.000	376	0.000	223	0.000	0.18
E06			0.000	408	0.000	247	0.000	0.22
E07			0.000	416	0.000	451	0.000	0.25
E08			0.000	418	0.000	728	0.000	0.28
E09			0.000	352	0.000	35	0.000	0.29
E10			0.000	353	0.000	70	0.000	0.32
Avg.			*0.002*	*316*	*0.015*	*290*	*0.000*	*0.14*

Table 15. Results obtained for the 22 BK subsets of instances.

			GRASP		RandCFH-ILS *tiny*		RandCFH-ILS *deep*	
Subset	**# customers**	**# facilities**	**Gap**	**t (s)**	**Gap**	**t (s)**	**Gap**	**t (s)**
GAP A	100	100	5.140	1.41	2.470	47.68	1.095	31.25
GAP B			5.980	1.81	2.811	63.56	1.642	40.72
GAP C			6.740	1.89	2.859	70.17	1.207	59.55
Avg.			*5.953*	*1.70*	*3.405*	*54.43*	*1.314*	*43.84*

Table 16. Results obtained for GAP subsets of instances.

The next test was performed over the FPP benchmark **(Table 17)**, known to include very hard instances for swapping algorithms. Our algorithm is not purely in this family, since we close and open an independent number of facilities at each overcoming iteration instead of directly swapping facilities.

			GRASP		RandCFH-ILS *tiny*		RandCFH-ILS *deep*	
Subset	**# customers**	**# facilities**	**Gap**	**t (s)**	**Gap**	**t (s)**	**Gap**	**t (s)**
FPP 11	133	133	8.480	2.58	0.063	127.13	0.000	111.02
FPP 17	307	307	58.270	25.18	70.731	272.94	12.283	253.44
Avg.			*33.375*	*13.88*	*35.397*	*200.03*	*6.142*	*253.44*

Table 17. Results obtained for FPP subsets of instances.

As in the GAP experimentation, our methodology outperforms the results quality of the Resende's proposal. In the small instance, we even discover the optimal solution when using the deep local search, while GRASP remained at an 8.4% gap. In this

benchmark, the tiny local search performs terribly due its simplicity in generating new solutions and the intended complexity of the problem instance.

Finally, we run the MED benchmark **(Table 18)**, the one with largest problem instances. In these tests, the optimal is not known, but lower and upper bounds obtained with exact methods are provided. In this chapter, we show the gap with respect to the lower bound and compare it to the same gap of the average obtained by the GRASP method.

			GRASP		RandCFH-ILS *tiny*		RandCFH-ILS *deep*	
Subset	**# customers**	**# facilities**	**Gap**	**t (s)**	**Gap**	**t (s)**	**Gap**	**t (s)**
0500-10	500	500	0.022	33.2	0.022	213	0.022	7.35
0500-100			0.016	32.9	0.093	676	0.014	345
0500-1000			0.071	23.6	0.071	3168	0.078	2803
1000-10	1000	1000	0.101	173.9	0.399	2038	0.099	69
1000-100			0.048	148.8	0.333	3566	0.088	2898
1000-1000			0.037	141.7	1.165	3468	0.447	4494
1500-10	1500	1500	0.191	347.8	0.236	2418	0.175	1012
1500-100			0.030	378.7	0.687	3582	0.094	1427
1500-1000			0.034	387.2	3.514	3540	0.320	20939
2000-10	2000	2000	0.052	717.5	0.223	3415	0.052	1276
2000-100			0.036	650.8	1.198	3573	0.299	2611
2000-1000			0.031	760.0	5.468	2838	0.403	57678
2500-10	2500	2500	0.164	1419.5	0.622	2988	0.168	2248
2500-100			0.049	1128.2	1.537	3569	0.349	4369
2500-1000			0.052	1309.4	5.964	3533	0.308	108575
3000-10	3000	3000	0.104	1621.1	0.372	3570	0.102	2362
3000-100			0.124	1977.6	1.707	3538	0.545	4904
3000-1000			0.043	2081.4	6.238	3427	0.319	228691
Avg.			*0.067*	*740*	*1.659*	*2951*	*0.938*	*13790*

Table 18. Results obtained for MED subsets of instances.

The results in **Table 18** show that our algorithm performs better than GRASP in the instances with larger setup costs. In the smallest instances, our running times are even competitive with Resende's experiments. However, on larger instances, our algorithm performs much slower, due to the longer list of open facilities included in these solutions. Still, we obtained competitive results, not far from the lower bound.

Notice some of the times employed to find the best solution are larger than the maximum run time given to the algorithm. This happens in the tests with the lowest facility opening costs (the -1000 instances). In these problems, the number of open facilities is considerably large and our algorithm tries to improve iteratively a long list. Thus, a single iteration takes longer than the time set for the stopping criteria.

With all the shown benchmarks, we can conclude our methodology outperforms the existing state-of-the-art heuristics in small and medium-scale scenarios, especially on those with short lists of open facilities. Thus, our algorithm could be a valuable tool for reduced or clustered scenarios, on which a large amount of clients could be served by a small amount of facilities.

6.5 Real Case Scenario: Minimizing Network Distance to Services

Network distance between servers and clients has a great impact on the quality perception for some Internet applications and the overall bandwidth consumption. For instance, video streaming services might be affected if congestion is found in the path from the server to the final client. Distributing services across the network is a good strategy to reduce these phenomena, but it comes at a high cost for operators.

Data intensive applications can also suffer from degradation and can generate high bandwidth demands if data is located far from the processing spot. In this field of study, **Ryden et al. (2013)** proposed a scenario in which user-contributed resources could be gathered to support data-intensive applications. Their proposal included a host selection strategy, based on bandwidth probes, in order to reduce overall bandwidth usage. Also targeting systems composed by non-dedicated resources, **Lazaro et al. (2012)** proposed an availability-aware host selection policy for service deployment in contributory communities. From the network perspective, their replica selection strategy would represent a random selection.

Studying which locations should be selected in a network to place a content or a service can be modeled as a FLP or a p-median problem **(Resende and Werneck, 2004)**. While the p-median problem does not consider any opening costs and restricts the number of facilities to open, the FLP considers the cost incurred when opening a new facility and relates the final number of open ones to the instance size and the opening cost values. In this particular case, the number of replicas each service should deploy is not known in advance. Regarding the network distances, the more service replicas in a network, the closer should be any client to any of them. However, more resources would be utilized and therefore fewer services could be supported in a platform of the same size.

	Average	Minima	Maxima
Network diameter		11	
Node degree	2.0476	1.0000	260.0000
Node degree centrality	0.0018	0.0009	0.2295
Node closeness centrality	0.2442	0.1212	0.4258
Node betweness centrality	0.0028	0.0000	0.7912
Path lengths	4.2244	1.0000	11.0000

Table 19. Network topology trait overview

In order to prove the value of our methodology in a real use case, we simulated a community cloud. To do so, we selected a mesh network topology from a Wireless Community Network **(WCN, Flickenberger 2002)**. This type of networks are

constructed, operated, maintained and owned by the users themselves and pose a great opportunity for community cloud success. We selected a network snapshot with 1134 nodes and 1161 links. Due to space limitations and the low visibility of such a large graph, we cannot depict the actual network topology in this chapter. Instead, we provide in **Table 19** some basic graph statistics to help the reader to have a better understanding of the type of network we are dealing with.

We considered the 53 nodes in the topology with more than one link to be the nodes that could host a service (facility) and all of them (including facilities) to be potential service consumers (customers). Considering facilities to be also consumers is one of the features in the community cloud and contributory communities' proposals, as users originally contribute their resources to support the platforms with the only reward of accessing these services. Although in this example we assume complete knowledge about the network topology, probing techniques like *traceroute* could be used in a real system to explore the underlying network connecting the nodes, in a similar way than **Ryden et al. (2013)** do with available bandwidth used by their allocation strategy.

To transform the network graph into a classical FLP instance, connection costs were established as the number of network hops from one node to another and the opening cost for each facility was defined as $c_i = 1000 \cdot closeness\_centrality_i$. The closeness centrality is a graph measure on how close is a vertex to all other vertices in the same graph. Thus, directly linking the opening cost of a facility at a given spot to this measure intuitively seemed a good strategy to associate higher costs to well connected hosts. We studied the application of our methodology in the described scenario and compared it to the following allocation strategies:

1. A greedy method that selects the top *N* nodes ordered by descending opening cost.
2. A greedy method that selects the top *N* nodes ordered by ascending opening cost.
3. A random selection of *N* nodes from all the available facilities.

In all cases, we set *N* to the same number of nodes selected by our FLP solving method, so the distance and cost comparison is done with the same level of resource usage. For the random selection strategy, we took the average from 100 samples. We gave 1 second of running time to our heuristic, a restricted time that would allow its use in a user-interactive service deployment process. We plot in **Figure 25** the distance of all clients in the network to its closest facility when selected with each of the explained methodologies. **Figure 26** shows the cost incurred by each of the service allocations.

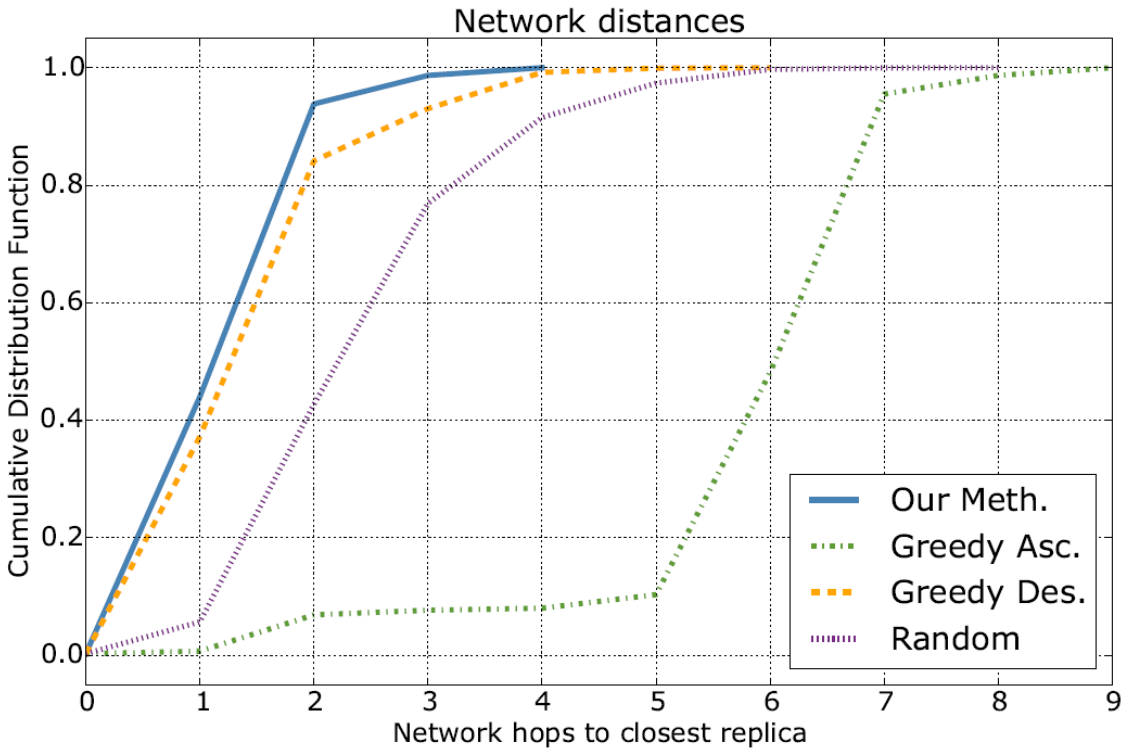

Figure 25. Cumulative distribution function of the distance to the closest replica from each client.

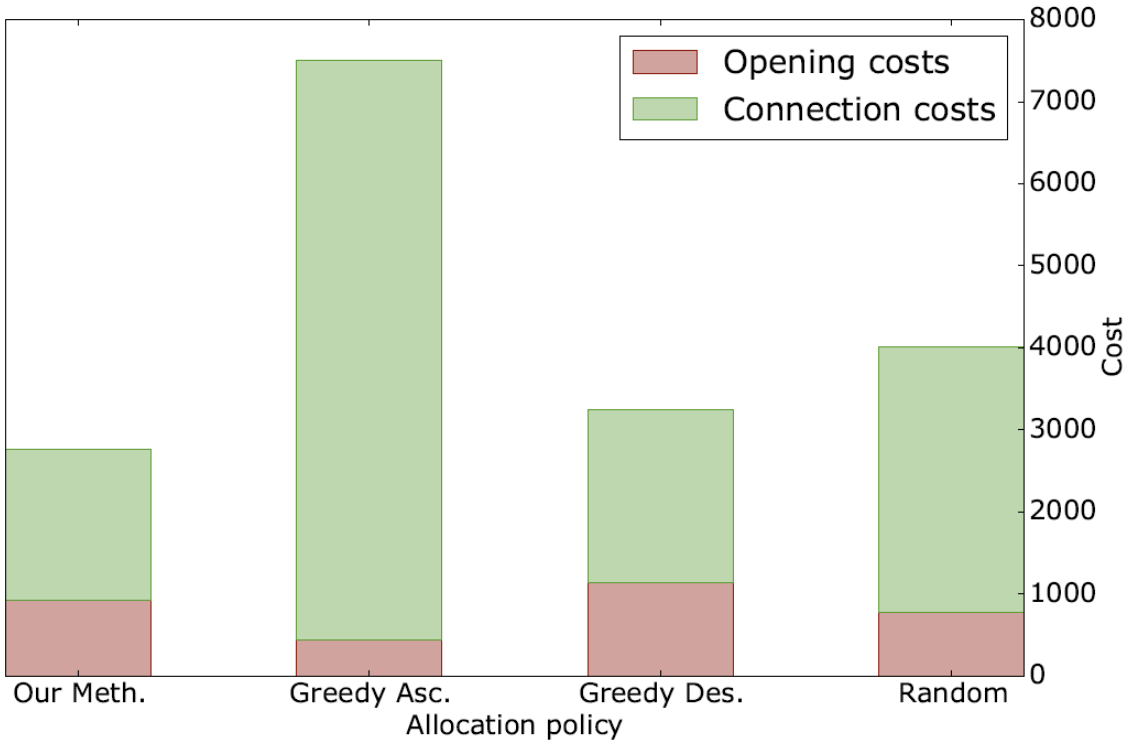

Figure 26. Cost distribution comparison of the different deployments obtained by the described allocation methods

As can be observed in both figures, our simulation-based methodology selects nodes in the network to host a single service that are closer to all other ones while maintaining a low deployment cost. Observing the disperse distances and costs obtained with both

greedy methodologies, we can also deduct that: (a) selecting those nodes with higher closeness centrality values results in low network distances and restrained total deployment costs; and (b) selecting nodes with lower closeness centrality results in very low opening costs but prohibitively high connecting ones and longer network distances. Both facts are a direct consequence of the cost function used to assign open costs for facilities in the FLP instance, but they help to value our proposal on finding low distance and low cost deployments.

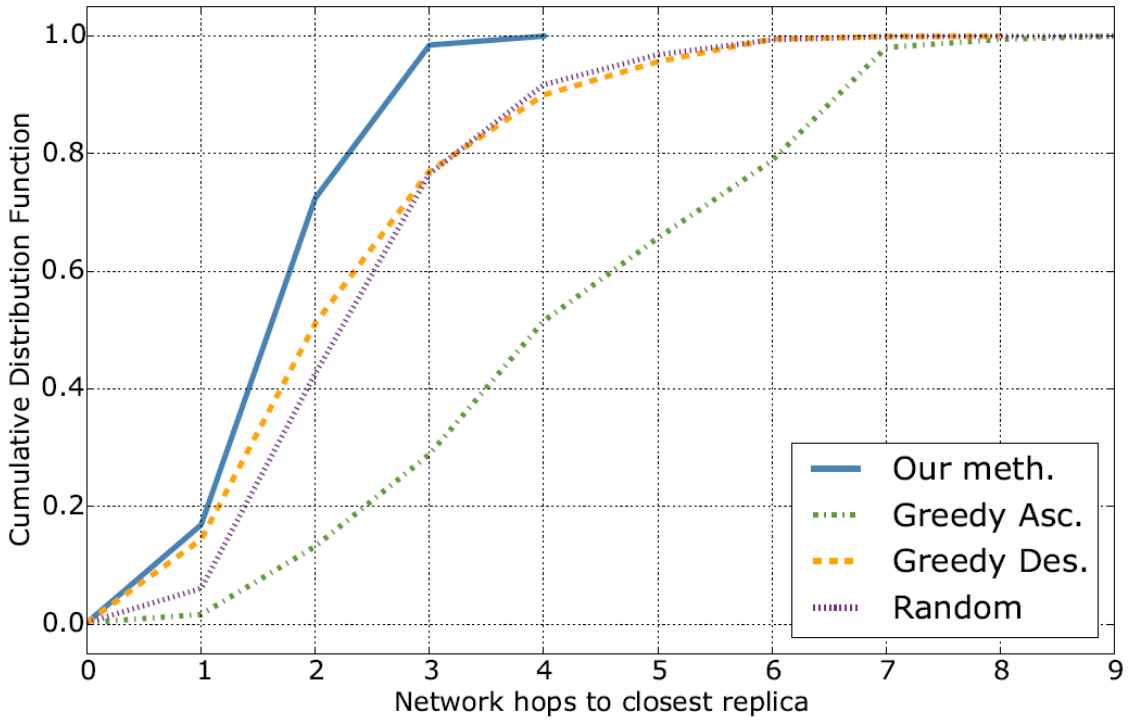

Figure 27. Cost distribution function of the mean network distances from each node to its closes replica.

In a real-life scenario, several services or applications should be concurrently supported in the same network. To show the behavior of our methodology in such case, we evaluated the network distances after five consecutive service allocations, considering only one service replica was supported at a time on each host. We show in **Figure 27** the distance of each node in the network to the closest facility of each of the five deployed services when replicas are allocated with the explained methodologies. **Figure 28** shows a box plot comparison of the total costs incurred by each of the deployed services. **Figure 27** show our methodology consistently allocates replicas closer to all other nodes in the network by using the same number of resources. Specially, if we look at the maximum distance, our proposal is able to allocate replicas at half the distance than other approaches. Thus, we can highlight our methodology does a better resource utilization, getting lower network distances without increasing

the number of consumed resources. Moreover, as the box plot in **Figure 28** reflects, the selected service allocations are also cheaper on average.

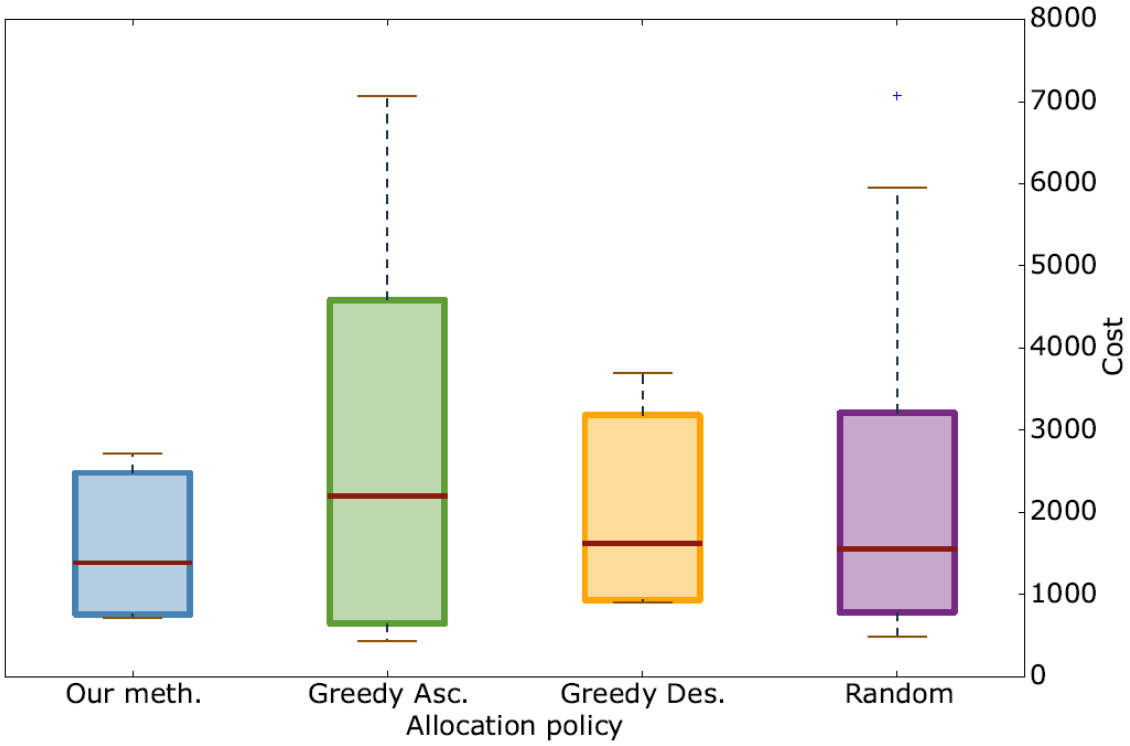

Figure 28. Box plot comparisons of the deployment costs with different allocation methodologies.

6.6 Chapter Conclusions

In the context of the Facility Location Problem applied to computer networks, this chapter has discussed the use of random variates generated from skewed probability distributions to induce a biased-randomized behavior inside a solving metaheuristic. The use of biased randomization helps the algorithm during the local search, thus providing shorter convergence times than the ones obtained by using standard uniform randomization. The resulting algorithm has been tested against a classical and well-studied benchmark for the FLP. As well, we applied our proposal to a real case scenario of a community platform and showed our methodology regularly selects network locations closer than the ones selected by any of the simple methods compared. The proposed algorithm while simple, as it is only requiring a single-parameter during the tuning phase, it is obtaining competitive results. These results confirm that simulation-inspired approaches like the one introduced in this proposal can become an efficient tool for solving FLP and other similar optimization problems in the field of distributed computing systems or telecommunication networks. We expect to

apply the presented methodology to larger network instances and study its application in real system deployments.

7 Conclusions and Contributions Derived from this Thesis

7.1 Conclusions

The Telecommunication sector is generating new optimization problems as technology advances occur. Improvements in the capacity of processing of computer processors, the definition of new concepts of network infrastructures like cloud computing, or the necessity of having faster internet connections which require of the deployment of newer network infrastructure, have associated with them optimization problems which are usually complex (or even impossible) to solve with exact methods. These scenarios usually can be modeled by already defined problems in more mature areas, as for example Transportation and Logistics. Also, solution methods can be reused for the new domain of problems or, at least, ideas behind them can be easily adapted to solve the new problems. One example of this is the MIRHA framework which we have presented in this thesis. The framework offers general concepts which can be followed for define biased randomized algorithms in different problems. In this thesis we have proposed new algorithms based on the MIRHA framework for optimization problems with potential applications in the Telecommunications field.

First of all, we have worked with the CARP. For this, we used the MIRHA framework to define a biased-randomized algorithm, and created two different versions of it, with different base heuristics. One was using the classical Path Scanning heuristic (PSH), and the other our SHARP heuristic. The SHARP heuristic is also an original contribution of this thesis as a heuristic for the Capacitated Arc Routing Problem (CARP). The heuristic being based on the classical Clarke and Wright Savings (CWS) heuristic for the Capacitated Vehicle Routing Problem (CVRP), obtained relatively-good results by itself. Results showed that the RandSHARP algorithm has a better performance than RandPSH. Also, RandSHARP results are competitive when compared with the state of the art. This makes the RandSHARP algorithm a very good alternative for real-life scenarios because, as it is quite easy to implement, and has

only one configuration parameter, the time required to implement the algorithm in a real case is lower than other methauristics with complex configuration steps.

Next, we dealt with the stochastic variation of the CARP, the Arc Routing Problem With Stochastic Demands (ARPSD). The ARPSD differs from the CARP in that the customer's demands are not known beforehand. This makes the problem more suitable for certain real cases in which the demands are not known beforehand but can only be modeled by a random variable. For solving the problem we defined an algorithm which combined the RandSHARP algorithm which we defined for the CARP (based on the MIRHA framework), with the Simheuristics framework. Simheuristics, with the use of biased-random sampling and Monte Carlo simulation, provide solutions which are robust for scenarios with uncertainty. From this we created another original algorithm: the Sim-RandSHARP. Mainly it solves a problem variation which considers safety stocks on the capacity, with RandSHARP, and then evaluates the robustness of the obtained solution with MCS. After several iterations the obtained solution demonstrates to be robust under these uncertain scenarios, while minimizing the total expected costs.

After that, a third variation of the CARP was solved: the non-smooth ARP. In the non-smooth ARP the function to be optimized is non-smooth. In the case of study this was due to the fact that the capacity constraint was changed to a soft constraint. This mean that the constraint can be violated but incurring in some penalty cost. For solving the problem, a variation of the RandSHARP for the problem was used, which consider the soft-constraint in the problem definition. We showed how the algorithm obtained also competitive results also when the cost function is non-smooth.

Finally, we studied the Facility Location Problem (FLP) which also has application in telecommunications problems. For the FLP we contributed an original algorithm also based on the aforementioned MIRHA framework. The biased-randomized algorithm designed demonstrated to be very competitive in terms of execution times and solutions qualities when compared with other algorithms (GRASP). Also, we evaluated the performance of the algorithm on a problem instances extracted from a real network scenario. In this problem instance the algorithm obtained high quality solutions when compared with different methods.

7.2 Future Work

In this thesis we widen the number of successful applications of the MIRHA framework and Simheuristics by studying new problems. However, further research can be

conducted in order to analyze different scenarios of these problems and also to afford new optimization problems:

- Additional problems can be studied for proposing algorithms with MIRHA framework and Simheuristics. In this thesis we have shown how the methodology can successfully be applied to different problems than the ones for which originally was proposed the methodology. Futures researches can propose algorithms based on MIRHA and/or Simheuristics to different problems.
- The proposed algorithms are parallelizable in a natural way. This is due to the nature of the MIRHA framework, which Iterated Local Search step can be easily parallelizable. Also, different steps of the problem specific algorithm can be parallelizable. With parallelization the results obtained by the algorithm in all the problems that we have studied (CARP, ARPSD, non-smooth ARP and FLP) can be improved by decreasing the required execution time when having a computer with multiple execution cores.
- Following the previous point, distributed computing can be also evaluated. In situations in which the computers available for executing the algorithm have not enough execution cores, we can share the capacity of several low-end computers to execute the parallelized version of the algorithm. Here the research would define how the computations of the algorithm would be split and coordinated within the distributed network.
- Study different variations of the ARPSD. The ARPSD models a scenario in which the customer demands is not known beforehand, but there are other stochastic scenarios that can be studied. For instance, another interesting scenario would be that one in which the real cost of traversing an edge is not known beforehand. In this case we have uncertainty in the cost function. A future research line could evaluate if the proposed SimRandSHARP algorithm is also robust when facing this uncertainty scenario.
- Propose an algorithm combining RandCFH-ILS and Simheuristics for the stochastic variation of the FLP. The FLP has a stochastic variation in which new customers appear in the network as the time passes. There is an interesting research line to propose a new algorithm for the problem and evaluate the robustness and quality of the obtained solutions.
- Finally, one problem of special interest for its possible applications in the Telecommunications field is the Connected Facility Location Problem (CFLP, **Gupta et al. 2001**). In the CFLP the goal is, in addition to select the location of

the different central nodes or facilities, to connect them with a Steiner Tree. Mainly it is a combination of a FLP with a Steiner Tree Problem (STP). Thus, another research line could be the adaption of the RandCFH-ILS algorithm, originally proposed to the FLP, to the CFLP. One example of application of the CFLP is the design of Virtual Private Networks (VPN).

7.3 Publications derived from this thesis

As a result of the research conducted within this thesis, several publications have been produced as part of the main contributions of this work. Thus, in this chapter, we present the accepted publications, the *in-process-of-reviewing* publications, some dissemination activities developed in the last four years, and finally there are some extra contributions related to the objectives of this dissertation that must be pointed out.

7.3.1 Publications

Among the publications derived from this thesis, we can remark some parts of this thesis which have been published in the following articles belonging to publications indexed in ISI-JCR or Elsevier-Scopus journals after a *peer-reviewing* process:

- ⇒ Gonzalez-Martin, S.; Juan, A.; Riera, D.; Castella, Q.; Muñoz, R; and Perez, A. (2012a). Development and assessment of the SHARP and RandSHARP algorithms for the arc routing problem. *AI Communications,* 25: 173-189. Indexed in ISI SCI, 2011 IF = 0.500, Q3. ISSN: 1134-5764.
- ⇒ Gonzalez-Martin, S.; Ferrer, A.; Juan, A.; and Riera, D. (2014c). Solving non-smooth arc routing problems throughout biased-randomized heuristics. In De Sousa, J.; and Rossi, R. (eds.), *Advances in Intelligent Systems and Computing*: 451-462. Indexed in ISI Web of Science and Scopus, 2013 SJR = 0.139, Q4. ISSN: 2194-5357.
- ⇒ Gonzalez-Martin, S.; Juan, A.; Riera, D.; Elizondo, M.; and Fonseca, P. (2012b). Sim-RandSHARP: A hybrid algorithm for solving the arc routing problem with stochastic demands. In *Proceedings of the 2012 Winter Simulation Conference*, Berlin, Germany: 1-11. Indexed in ISI Web of Science and Scopus, 2011 SJR=0.372, Q2. ISSN: 08917736.
- ⇒ Gonzalez-Martin, S.; Barrios, B.; Juan, A.; and Riera, D. (2014a). On the use of biased randomization and simheuristics to solve vehicle and arc routing problems. In *Proceedings of the 2014 Winter Simulation Conference*

(accepted), Savannah, USA, December 7-10. Indexed in ISI Web of Science and Scopus, 2011 SJR=0.372, Q2. ISSN: 08917736.

⇒ Cabrera, G.; Gonzalez-Martin, S.; Juan, A; and Marques, J. (2014). Combining biased random sampling with metaheuristics for the facility location problem in distributed computer systems. In *Proceedings of the 2014 Winter Simulation Conference (accepted)*, Savannah, USA, December 7-10. Indexed in ISI Web of Science and Scopus, 2011 SJR=0.372, Q2. ISSN: 08917736.

⇒ Juan, A.; Barrios, B.; Coccola, M.; Gonzalez-Martin, S.; Faulin, J.; and Bektas, T. (2012b). Combining biased randomization with meta-heuristics for solving the multi-depot vehicle routing problem. In *Proceedings of the 2012 Winter Simulation Conference,* Berlin, Germany, December 9-12. Indexed in ISI Web of Science and Scopus, 2011 SJR=0.372, Q2. ISSN: 08917736.

Also, at the moment of writing this dissertation, other parts of this thesis have been submitted to a *peer-reviewing* process of other ISI-JCR publications, but still are under review at the moment of writing this document:

⇒ Gonzalez-Martin, S.; Juan, A.; Riera, D.; Elizondo, M.; and Ramos, J. (2014b). A simheuristic algorithm for solving the arc routing problem with stochastic demands. *Applied Soft Computing.*

⇒ Gonzalez-Martin, S.; Cabrera, G.; and Juan, A. (2014d). BRILSA: A biased-randomized ILS algorithm for the uncapacitated facility location problem. *In work.*

In addition, there are some conference-papers associated to ISI-WOS or Elsevier-Scopus journals which were accepted after a *peer-reviewing* process:

⇒ Gonzalez-Martin, S.; Juan, A.; Riera, D.; and Caceres, J. (2011). A hybrid algorithm combining path scanning and biased random sampling for the arc routing problem. In *Proceedings of the 18th RCRA Workshop, Barcelona, Spain*: 46-54.

⇒ Juan, A.; Faulin, J.; Caceres, J.; and Gonzalez-Martin, S. (2011d). Combining randomized heuristics, monte-carlo simulation and parallel computing to solve the stochastic vehicle routing problem. In: *Proceedings of the international conference on Optimization, Theory, Algorithms and Applications in Economics (OPT2011),* Barcelona, October 24-28.

⇒ Gonzalez-Martin, S.; Ferrer, A.; Juan, A.; and Riera, D. (2013). Solving non-smooth arc routing problems throughout biased-randomzied heuristics. In:

Proceedings of the 16th annual meeting of Euro Working Group on Transportation, Porto, Protugal, September 4-6.

⇒ Ferrer, A.; Juan, A.; Gonzalez-Martin, S., and Lourenço, H. (2013). Randomized algorithms for solving routing problems with non-smooth objective functions. In: *26th European Conference on Operational Research (EURO 2013).* July 1-4, Rome.

⇒ Fernandez-Piñas, D.; Gonzalez-Matin, S.; Juan, A.; and Riera, D. (2013). A heuristic algorithm for the resource assignment problem in satellite telecommunication networks. In: *Proceedings of the 20th RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion*, Roma, Italy, June 14-15.

Additionally, the following book chapter was co-authored:

⇒ Juan, A.; Caceres, J.; Gonzalez-Martin, S.; Riera, D.; and Barrios, B. (2014a). Biased randomization of Classical Heuristics. In Wang, J. (ed.), *Encyclopedia of Business Analytics and Optimization*, IGI Global, 1: 314-324.

7.3.2 Presentations

Some parts of this work have also been presented in several international Congresses, Conferences and Workshops, and published in the following activities:

⇒ Juan, A.; Gonzalez-Martin, S.; Elizondo, M.; Riera, D. (2012). A hybrid algorithm for solving the arc routing problem with stochastic demands. 2012 IN3-HAROSA International Workshop. June 13-15, Barcelona.

⇒ Gonzalez-Martin, S.; Juan, A.; Cabrera, G.; and Riera, D. (2013). Applying MIRHA to the Connected Facility Location Problem. 2013 ICSO-HARSA International Workshop. July 10-12, Barcelona.

⇒ Muñoz, C.; Gonzalez-Martin, S.; Candia, A.; and Juan, A. (2014). Solving Arc Routing Problem with a Hybrid Electromagnetic Mechanism Algorithm. In: *XLVI Brazilian Symposium of Operational Research*, Salvador de Bahia, September 16-19.

7.3.3 Other contributions

During the thesis period, additional activities have been conducted which, not being included on this thesis dissertation, were on topics closely related to the combinatorial optimization and telecommunications fields. In particular, the guidance of a final Master Thesis for the degree of Master in *Software Libre* from the *Universitat Oberta de*

Catalunya (UOC) was conducted in collaboration with Dr. Angel A. Juan (advisor of this PhD thesis). The thesis was developed by the student David Fernandez Piñas (**Fernandez, 2013**), and a part of it was published in the RCRA workshop:

⇒ Fernandez-Piñas, D.; Gonzalez-Matin, S.; Juan, A.; and Riera, D. (2013). A heuristic algorithm for the resource assignment problem in satellite telecommunication networks. In: *Proceedings of the 20th RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion*, Roma, Italy, June 14-15.

Additionally, collaboration was done with the student Carlos Muñoz from the Talca University (Chile) during his stage at UOC, for the work done for his final master thesis in Industrial Engineering. As a result of this collaboration, the following conference paper was produced:

⇒ Muñoz, C.; Gonzalez-Martin, S.; Candia, A.; and Juan, A. (2014). Solving Arc Routing Problem with a Hybrid Electromagnetic Mechanism Algorithm. In: *XLVI Brazilian Symposium of Operational Research*, Salvador de Bahia, September 16-19.

Also the following co-authored paper was presented as a poster in the Winter Simulation Conference:

⇒ Juan, A.; Barrios, B.; Coccola, M.; Gonzalez-Martin, S.; Faulin, J.; and Bektas, T. (2012b). Combining biased randomization with meta-heuristics for solving the multi-depot vehicle routing problem. In *Proceedings of the 2012 Winter Simulation Conference,* Berlin, Germany, December 9-12.

References

Al-Karaki, J; and Kamal, A. (2004) Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications,* 11: 6-28.

Al-Sultan, K. (1995). A tabu search approach to the clustering problem. *Pattern Recognition,* 28(9): 1443-1451.

Alves, M.; and Almeida, M. (1992). Simulated annealing algorithm for the simple plant location problem. In *Revista Investigação Operacional*, 12.

Amberg, A.; Domschke, W.; and Voß, S. (2000). Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees. *European Journal of Operational Research*, 124: 360-376.

Amberg, A.; and Voß, S. (2002). A hierarchical relaxation lower bound for the capacitated arc routing problem. In *Proceedings of the 35th Annual Hawaii Internat. Conference System Science*.

Assad, A.; and Golden, B. (1995). Arc routing methods and applications. In M. G. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, eds. *Network Routing, volume 8 of Handbooks in Operations Research and Management Science*, Elsevier: 375-483.

Bagirov, A.; and Yearwood, J. (2006). A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problem. *European Journal of Operations Research*, 170: 578-596.

Bagirov, A.; Lai, D.; and Palaniswami, M. (2007). A nonsmooth optimization approach to sensor network location. In: Palaniswami, M.; Marusic, M.; and Law, Y. (eds.). *Proceedings of the 2007 international conference on intelligent sensors, sensor networks and information processing*: 727-732.

Balachandran, V.; and Jain, S. (1976). Optimal facility location under random demand with general cost structure. *Naval Research Logistics Quarterly*, 23: 421-436.

Baldacci, R.; and Maniezzo, V. (2006). Exact methods based on node routing formulations for undirected arc-routing problems. *Networks*, 47(1): 52-60.

Balinski, M. (1966). On finding integer solutions to linear programs. In *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problem*: 225-248.

Belenguer, J.; and Benavent, E. (1992) Polyhedral results on the capacitated arc routing problem. *Technical Report TR-01*, Dep. Estadística e Inv. Op., Universidad de Valencia.

Belenguer, J.; and Benavent, E. (1998). The capacitated arc routing problem: Valid inequalities and facets. *Computational Optimization and Applications*, 10(2):165-187.

Belenguer, J.; and Benavent, E. (2003). A cutting plane algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 30(5):705-728.

Belenguer, J.; Benavent, E.; Lacomme, P.; and Prins, C. (2006). Lower and upper bounds for the mixed capacitated arc routing problem. *Computers & Operations Research*, 33-12:3363-3383.

Belenguer, J. (2014). http://www.uv.es/belengue/carp.html, accessed on July 2014.

Beltrami, E.; and Bodin, L. (1974). Networks and vehicle routing for municipal waste collection. *Networks,* 4(1): 65-94.

Bertsimas, D.; and Howell, L. Further results on the probabilistic traveling salesman problem. *European Journal of Operational Research*, 65(1): 68-95.

Beullens, P.; Muyldermans, L.; Cattrysse, D.; and Oudheusden, D. (2003). A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research*, 147(3): 629-643.

Bodin, L.; and Kursh, S. A detailed description of a computer system for the routing and scheduling of street sweepers. *Computers & Operations Research,* 6(4): 181-198.

Boyd, S.; and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, Cambridge, UK.

Brahimi, N.; and Khan, S. (2013). Warehouse location with production, inventory, and distribution decisions: a case study in the lube oil industry. In *4OR Quarterly Journal in Operations Research*, DOI: 10.1007/s10288-013-0237-0: 1-23.

Brandâo, J.; and Eglese, R. (2008). A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 35(4): 1112-1126.

Bresina, J. (1996). Heuristic-biased stochastic sampling. In *Proceedings of the 13th National Conference On Artificial Intelligence and The 8th Innovative Applications of Artificial Intelligence Conference*, 1-2: 271-278.

Cabrera, G.; Gonzalez-Martin, S.; Juan, A; and Marques, J. (2014). Combining biased random sampling with metaheuristics for the facility location problem in distributed computer systems. In *Proceedings of the 2014 Winter Simulation Conference (accepted).*

Cabrera, G.; Gonzalez, S.; Juan, A.; and Grasman, S (2014b). A multi-start based algorithm with iterated local search for the uncapacitated facility location problem. In *IFORS 2014*, July 13-18, Barcelona.

Caceres, J.; Juan, A.; Grasman, S.; Bektas, T.; Fauling, J. (2012). Combining monte carlo simulation with heuristics for solving the inventory routing problem with stochastic demands. In *Proceedings of the 2012 Winter Simulation Conference,* Berlin, Germany.

Carrizosa, E.; Ushakov, A.; and Vasilyev, I (2012). A computational study of nonlinear minsum facility location problem. *Computers & Operations Research*, 39: 2625-2633.

Cattrysse, D.; Lotan, T.; Muyldermands, L.; and Van Oudeheusden, D. (2002). Districting for salt spreading operations. *Europan Journal of Operational research*, 139(3): 521-532.

Chapleau, L.; Ferland, J.; Lapalme, G.; and Rousseau, J. (1984). A parallel insert method for the capacitated arc routing problem. *Operations Research Letters*, 3(2): 95-99.

Chaves, A.; and Lorena, L. (2010). Clustering search algorithm for the capacitated centered clustering problem. *Computers & Operations Research,* 37(3): 552-558.

Choudhary, M.; Khan, N.; Abbas, A.; and Salman, A. (2008). Telecom sector deregulation, business growth and economic development. In *Proceedings of the PICMET 2008, 27-31 July, Cape Town, South Africa.*: 2648-2656.

Christinasen, C.; and Lysgaard, J. (2007). A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters*, 35(6): 773-781.

Christiansen, C.; Lysgaard, J.; and Wøhlk, S. (2009). A brahc-and-price algorithm for the capacitated arc routing problem with stochastic demands. *Operations Research Letters*, 37(6): 392-398.

Chu, F.; Labadi, N.; and Prins, C. (2003). A scatter search for the periodic capacitated arc routing problem. *European Journal of Operational Research*, 169(2):586-605.

Chudak, F. (1998). Improved approximation algorithms for uncapacitated facility location. In: Bixby, R.; Boyd, E.; and Rios-Mercado, R. (3ds), *Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science*, 1412, Springer: 180-194.

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

Cooper, L. (1963). Location-allocation problems. *Operations Research*, 11(3): 331-343.

Cordeau, J.; Gendreau, M.; Laporte, G.; Potvin, J.; and Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of Operations Research Society*, 53: 512-522.

Cormen, R.; Leiserson, R.; Rivest, R.; and Stein, C. (2009) *Introduction to algorithms*, 3rd edition, MIT Press, Cambridge, MA.

Cornuejols, G.; Nemhauser, G.; and Wolsey, L. (1990). The uncapacitated facility location problem. In: Mirchandani, P.; and Francis, R. (eds.), *Discrete Location Theory*, Wiley-Interscience, New York: 119-171.

Del Pia, A.; and Filippi, C. (2006). A variable neighborhood descent algorithm for a real waste collection problem with mobile depots. *International Transactions in Operational Research*, 13(2):125-141.

Doerner, K.; Hartl, R.; Maniezzo, V.; and Reimann, M. (2003). An ant system metaheuristic for the capacitated arc routing problem. In *Preprints of 5th Meta-heuristics International Conference, Kyoto*.

Dorigo, M.; and Stutzle, T. (2010). Ant colony optimization: overview and recent advances. In Gendreau, M.; and Potvin, J. (eds.), *Handbook of metaheuristics. International series in operations research management science,* 146, 2nd edition, Kluwer Academic, Dordrecht: 227-264.

Drezner, Z. (1995). Facility location: a survey of applications and methods. *Springer series in Operations Research and Financial Engineering*.

Drezner, Z.; and Hamacher, H. (2010). *Facility Location: applications and Theory*. Springer, New York.

Dror, M. (2000). *Arc Routing: Theory, Solutions and Application,* Kluwer Academic, Boston, MA.

Efroymson, M.; and Ray, T. (1966). A branch and bound algorithm for plant location. *Operation Research*, 14: 361-368.

Eglese, R.; and Li, L. (1992). Efficient routing for winter gritting. *Journal of Operaltional Research Society*, 43: 1031-1034.

Eglese, R. (1994). Routing winter gritting vehicles. Discrete Applied Mathematics, 48(3): 231-244.

Eglese, R.; and Letchford, A. (2000) Polyhedral theory for arc routing problems. In M. Dror, eds., *Arc Routing: Theory, Solutions and Applications*, Kluwer Academic Publishers, Dordrecht: 199-230.

Eiselt, H.; Gendreau, M.; and Laporte, G. (1995). Arc routing problems ii: The rural postman problem. *Operations Research*, 43(3): 399-414.

Erlenkotter, D. (1978). A dual-based procedure for uncapacitated location. *Operation Research,* 26: 992-1009.

Faulin, J.; Juan, A.; Grasman, S.; Fry, M. (eds.) (2012). *Decision Making in Service Industries: A Practical Approach.* CRC Press – Taylor & Francis, Clermont, FL, USA.

Fernandez-Piñas, D. (2013). A simulation-based algortihm for solving the resource-assignment problem in satellite telecommunication networks. *Final Master Thesis, UOC.*

Fernandez-Piñas, D.; Gonzalez-Matin, S.; Juan, A.; and Riera, D. (2013). A heuristic algorithm for the resource assignment problem in satellite telecommunication networks. In: *Proceedings of the 20th RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion*, Roma.

Ferrer, A.; Juan, A.; Gonzalez-Martin, S.; and Lourenço, H. (2013). Randomized algorithms for solving routing problems with non-smooth objective functions. In *Proceedings of the 26th European Conference on Operational Research (EURO 2013), Rome, Italy.*

Festa, P.; and Resende, M. (2009a). An annotated bibliography of GRASP – Part I: algorithms. *International Transactions in Operational Research*, 16: 1-24.

Festa, P.; and Resende, M. (2009a). An annotated bibliography of GRASP – Part II: applications. *International Transactions in Operational Research*, 16: 1-24.

Fleury, G.; Lacomme, P.; Prins, C. and Ramdane-Cherif, W. (2002). Robustness evaluation of solutions for the capacitated arc routing problem. In: *Conference, AI Simulation and Planning in High Autonomy Systems*: 290-295.

Fleury, G.; Lacomme, P.; and Prins, C. (2004). Evolutionary algorithms for stochastic arc routing problems. In: Raidl, G.; Rothlauf, F.; Smith, G.; Squillero, G.; Cagnoni, S.; Branke, J.; Corne, D.; Drechsler, R.; Jin, Y.; and Johnson, C. (eds.) *Applications of Evolutionary Computing*, volume 3005 of *Lecture Notes in Computer Science*: 501-512.

Fleury, G.; Lacomme, P.; Prins, C. and Ramdane-Cherif, W. (2005). Improving robustness of solutions to arc routing problems. *Journal of the Operational Research Society*, 56(5): 526-538.

Fotakis, D. (2011). Online and incremental algorithms for facility location. In *SIGACT News,* 42(1): 97-131.

Frederickson, G.; Hecht, M.; and Kim, C. (1978). Approximation algorithms for some routing problems. *SIAM Journal of Computing*, 7(2): 178-193.

Ghosh, D. (2003). Neighborhood search heuristics for the uncapacitated facility location problem. *European Journal of Operational Research*, 150: 150-162.

Golden, B.; and Wong, R. (1981). Capacitated arc routing problems. *Networks*, 11(3): 305-315.

Golden, B.; DeArmon, J.; and Baker, E. (1983). Computational experiments with algorithms for a class of routing problems. *Computers & Operational Research*, 10(1): 47-59.

Gomes, C.; and Selman, B. (1997). Practical aspects of algorithm portfolio design. In *Proceedings of the 3rd ILOG International Users Meeting*: 200-201.

Gonzalez-Martin, S.; Juan, A.; Riera, D.; and Caceres, J. (2011). A hybrid algorithm combining path scanning and biased random sampling for the arc routing problem. In *Proceedings of the 18th RCRA Workshop, Barcelona, Spain*: 46-54.

Gonzalez-Martin, S.; Juan, A.; Riera, D.; Castella, Q.; Muñoz, R; and Perez, A. (2012a). Development and assessment of the SHARP and RandSHARP algorithms for the arc routing problem. *AI Communications,* 25: 173-189.

Gonzalez-Martin, S.; Juan, A.; Riera, D.; Elizondo, M.; and Fonseca, P. (2012b). Sim-RandSHARP: A hybrid algorithm for solving the arc routing problem with stochastic demands. In *Proceedings of the 2012 Winter Simulation Conference*, Berlin, Germany: 1-11.

Gonzalez-Martin, S.; Ferrer, A.; Juan, A.; and Riera, D. (2013). Solving non-smooth arc routing problems throughout biased-randomzied heuristics. In: *Proceedings of the 16th annual meeting of Euro Working Group on Transportation*, Porto, Protugal, September 4-6.

Gonzalez-Martin, S.; Barrios, B.; Juan, A.; and Riera, D. (2014a). On the use of biased randomization and simheuristics to solve vehicle and arc routing problems. In *Proceedings of the 2014 Winter Simulation Conference* (accepted).

Gonzalez-Martin, S.; Juan, A.; Riera, D.; Elizondo, M.; and Ramos, J. (2014b). A simheuristic algorithm for solving the arc routing problem with stochastic demands. *Applied Soft Computing (under review).*

Gonzalez-Martin, S.; Ferrer, A.; Juan, A.; and Riera, D. (2014c). Solving non-smooth arc routing problems throughout biased-randomized heuristics. In De Sousa, J.; and Rossi, R. (eds.), *Advances in Intelligent Systems and Computing*: 451-462.

Gonzalez-Martin, S.; Cabrera, G.; and Juan, A. (2014d). BRILSA: A biased-randomized ILS algorithm for the uncapacitated facility location problem. In *XXXXX. (in progress)*.

Greistorfer, P. (2003). A tabu scatter search metaheuristic for the arc routing problem. *Computers & industrial Engineering*, 44(2): 249-266.

Gupta, A.; Kleinberg, J.; Kumar, A.; Rastogi, R.; and Yener, B. (2001). Provisioning a virtual private network: a network design problem for multicommodity flow. In *Proceedings of the 33rd annual ACM symposium on theory of computing*: 389-398.

Hamdan, M.; and El-Hawary, M. (2002). Hopfield-generic approach for solving the routing problem. In: *Computer Networks: Proceedings of the 2002 IEEE Canadian conference on electrical and computer engineering*: 823-827.

Hansen, P.; Mladenovic, N.; Brimberg, J.; and Moreno-Perez, J. (2010). Variable neighborhood search. In, Gendreau, M.; and Potvin, J. (eds.), *Handbook of metaheuristics. International*

series in operations research management science, 146, 2nd edition, Kluwer Academic, Dordrecht: 61-86.

Hashimoto, H.; Ibaraki, T.; Imahori, S.; and Yagiura, M. (2006). The vehicle routing problem with flexible time windows and traveling times. *Discrete Applied Mathematics*, 154(16): 2271-2290.

Hertz, A.; Laporte, G.; and Mittaz, M. (2000). A tabu search heuristic for the capacitated arc routing problem. *Operations Research*, 48(1): 129-135.

Hertz, A.; and Mittazl, M. A variable neighborhood descendent algorithm for the undirected capacitated arc routing problem. *Transportation Science*, 35(4): 425-434.

Hirabayashi, R.; Nishida, N.; and Saruwatari, Y. (1992). Tour construction algorithm for the capacitated arc routing problem. *Asia-Pacific Journal of Operational Research*, 9(2):155-175.

Hochbaum, D. (1982). Approximation algorithms for the weighted set covering and node cover problems. In *SIAM Journal of Computing*, 11(3): 555-556.

Hoefer, M. (2014). http://www.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/, accessed on July 2014.

Ismail, Z.; and Ramli, M. (2011). Implementation of weather-type models of capacitated arc routing problems via heuristics. *American Journal of Applied Sciences*, 8(4): 382-392.

Jain, K.; Mahdina, M.; Markakis, E.; Saberi, A.; and Vazirani, V. (2003). Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50: 795-824.

Jain, K.; and Vazirani, V. (1999). Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamos, California: 2-13.

Juan, A.; Faulin, J.; Ruiz, R.; Barrios, B.; and Caballero, S. (2010). The SR-GCWS hybrid algorithm for solving the Capacitated Vehicle Routing Problem. *Applied Soft Computing,* 10(1): 215-224.

Juan, A.; Faulin, J.; Ferrer, A.; Lourenço, H.; and Barrios, B. (2011a). MIRHA: multi-start biased randomization of heuristics with adaptive local search for solving non-smooth routing problems.

Juan, A.; Faulin, J.; Grasman, S.; Riera, D.; Marull, J.; and Mendez, C. (2011b). Using safety stocks and simulations to solve the vehicle routing problem with stochastic demands. *Transportation Research Part C*, 19: 751-765.

Juan, A.; Faulin, J.; Jorba, J.; Caceres, J.; and Marques, J. (2011c). Using parallel & distributed computing for solving real-time vehicle routing problems with stochastic demands. *Annals of Operations Research,* DOI 10.1007/s10479-011-0918-z.

Juan, A.; Faulin, J.; Caceres, J.; and Gonzalez-Martin, S. (2011d). Combining randomized heuristics, monte-carlo simulation and parallel computing to solve the stochastic vehicle

routing problem. In: Proceedings of the international conference on Optimization, Theory, Algorithms and Applications in Economics (OPT2011), Barcelona, October 24-28.

Juan, A.; Lourenço, H.; Mateo, M.; Castella, Q.; and Barrios, B. (2012). ILS-ESP: An efficient, simple and parameter-free algorithm for solving permutation flow-shop problem. In *Proceedings of the 6th Workshop on Statistics, Mathematics and Computations,* ISBN: 978-972-9473-62-3, July 3-4, Covilha, Portugal.

Juan, A.; Barrios, B.; Coccola, M.; Gonzalez-Martin, S.; Faulin, J.; and Bektas, T. (2012b). Combining biased randomization with meta-heuristics for solving the multi-depot vehicle routing problem. In *Proceedings of the 2012 Winter Simulation Conference,* Berlin, Germany, December 9-12.

Juan, A.; Caceres, J.; Gonzalez-Martin, S.; Riera, D.; and Barrios, B. (2014a). Biased randomization of Classical Heuristics. In Wang, J. (ed.), *Encyclopedia of Business Analytics and Optimization*, IGI Global, 1: 314-324.

Karger, D.; and Minkoff, M. (2000). Building Steiner Trees with Incomplete Global Knowledge, In: *Proceedings of the 21st annual symposium in Foundations of Computer* Science: 613-623.

Kennedy, J.; and Eberhart, R. (1995). Particle swarm optimization. In *1995 IEEE international conference on neural networks proceedings*, 1(6): 1942-1948.

Kiuchi, M.; Shinano, Y.; Hirabayashi, R.; and Saruwatari, Y. (1995). An exact algorithm for the capacitated arc routing problem using parallel branch and bound method. *Abstracts of the 1995 Spring National Conference of the Oper. Res. Soc. of Japan*: 28-29.

Klincewicz, J.; and Luss, H. (1987). A dual based algorithm for multiproduct uncapacitated facility location. *Transportation Science*, 21: 198-206.

Körkel, M. (1989). On the exact solution of large-scale simple plant location problems. *European Journal of Operational Research*, 39: 157-173.

Kratica, J.; Tosic, D.; Filipovic, V.; and Ljubic, I. (2001). Solving the simple plant location problem by genetic algorithm. In *RAIRO Operations Research*, 35: 127-142.

Kuehn, A., and Hamburger, M. (1963). A heuristic program for locating warehouses. *Management Science*, 9(4): 643-666.

L'Ecuyer, P. (2006). Random number generation in simulation. *Elsevier, Amsterdam*.

Lam, P.; and Shiu, A. (2010). A bi-criteria approach for Steiner's tree problems in communications networks. In *Proceedings of the 2011 International Workshop on Modeling, Analysis and Control of Complex Networks, San Francisco, Califronia*: 37-44.

Lacomme, P.; Prins, C.; and Ramdane-Chérif, W. (2001). Competitive genetic algorithms for the capacitated arc routing problem and its extensions. *Lecture Notes in Computer Science*, 2037: 473-483.

Lacomme, P.; Prins, C.; and Ramdane-Chérif, W. (2004a). Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, 131(1): 159-185.

Lacomme, P.; Prins, C.; and Tanguy, A. (2004b). First competitive ant colony scheme for the CARP. *Lecture Notes in Computer Science*, 3172: 426-427.

Lacomme, P.; Prins, C.; and Ramdane-Chérif, W. (2005). Evolutionary algorithms for periodic arc routing problems. European Journal of Operational Research, 165(2):535-553.

Lai, M.; Sohn, H.; Tseng, T.; and Chiang, C. (2010). A hybrid algorithm for capacitated plant location problem. *Expert Systems with Applications*, 37: 8599-8605.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43(4): 408-416.

Laporte, G.; Musmanno, R.; and Vocaturo, F. (2010). An adaptive large neighborhood search heuristic for the capacitated arc routing proble with stochastic demands. *Transportation Science*, 44(1): 125-135.

Lazaro, D.; Kondo, D.; and Marques, J. (2012). Long-term availability prediction for groups of volunteer resources. *Journal of Parallel and Distributed Computing,* 72-2: 281-296.

Lee, G.; and Murray, A. (2010). Maximal covering with network survivability requirements in wireless mesh networks. *Computers. In Environment and Urban Systems*, 34: 49–57.

Letchford, A.; and Oukil, A. (2009). Exploiting sparsity in pricing routines for the capacitated arc routing problem. *Computers & Operations Research*, 36(7): 2734-2742.

Li, L. (1992). Vehicle routeing for winter gritting. *Ph. D. thesis*, Department of Management Science, Lancaster University.

Li, S. (2013). A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222: 45-58.

Loiola, E.; de Abreu, N.; Boaventura-Netto, P.; Hahn, P.; and Querido, T. (2007). A survey for the quadratic assignment problem. *European Journal in Operational Research*, 176(2): 657-690.

Lokketangen, A.; and Glover, F. (1998). Solving zero-one mixed integer programming problems using tabu search. *European Journal in Operational Research,* 106(2-3): 624-658.

Longo, H.; Aragão, M.; and Uchoa, E. (2006). Solving capacitated arc routing problems using a transformation to the CVRP. *Computers & Operations Research*, 33(6): 1823-1837.

Lourenço, H.; Martin, O.; and Stutzle, T. (2010). Iterated local search: framework and applications. In,Gendreau, M.; and Potvin, J. (eds.), *Handbook of metaheuristics. International series in operations research management science,* 146, 2nd edition, Kluwer Academic, Dordrecht: 363-397.

Maniezzo, V.; Roffilli, M. (2008). Algorithms for large directed capacitated arc routing problem instances. In C. Cotta, J. van Hemert eds., *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, volume 153 of *Studies in Computational Intelligence*. Springer Berlin / Heidelberg: 259-274.

Maric, M.; Stanimirovic, Z.; and Bozovic, S. (2013). Hybrid metaheuristic method for determining locations for long-term health care facilities. *Annals of Operations Research*, DOI: 10.1007/s10479-013-1313-8: 1-21.

Meyerson, A. (2001). Online facility location. In *FOCS '01 Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*: 426-432.

Michel, L.; and Van Hentenryck, P. (2003). A simple tabu search for warehouse location. *European Journal on Operations Research*, 157(3): 576-591.

Montoya, J.; Juan, A.; Huaccho, L.; Faulin, J.; and Rodriguez-Verjan, G. (eds.) (2011). *Hybrid algorithms for service, computing and manufacturing systems: routing and scheduling solutions.* IGI Global, Hershey.

Muñoz, C.; Gonzalez-Martin, S.; Candia, A.; and Juan, A. (2014). Solving Arc Routing Problem with a Hybrid Electromagnetic Mechanism Algorithm. In: *XLVI Brazilian Symposium of Operational Research*, Salvador de Bahia, September 16-19.

Nawaz, M.; Enscore, E.; and Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. *OMEGA,* 11: 91-95.

Nikolaev, A.; and Jacobson, S. (2010). Simulated annealing. In, Gendreau, M.; and Potvin, J. (eds.), *Handbook of metaheuristics. International series in operations research management science,* 146, 2nd edition, Kluwer Academic, Dordrecht: 1-40.

Novoa, C.; and Storer, R. (2009). An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 196: 509-515.

Oonsivilai, A.; Srisuruk, W.; Marungsri, B.; and Kulworawanichpong, T. (2009). Tabu search approach to solve routing issues in communication netowkrs. In: *World academy of science, engineering and technology*: 1174-1177.

Partridge, C. (2011). Forty data communications research questions. *ACM SIGCOMM Computer Communication Review,* 41(5): 24-35.

Pearn, W.; Assad, A.; and Golden, B. (1987) Transforming arc routing into node routing problems. *Computers & Operations Research*, 14(4): 285-288.

Pearn, W. (1988). New lower bounds for the capacitated arc routing problem. *Networks*, 18(3): 181-191.

Pearn, W. (1989). Approximate solutions for the capacitated arc routing problem. *Computers & Operations Research*, 16(6): 589-600.

Pearn, W. (1991). Argument-insert algorithms for the capacitated arc routing problem. *Computers & Operations Research*, 18(2): 189-198.

Peruyero, E.; Juan, A.; and Riera, D. (2011). A hybrid algorithm combining heuristics with monte carlo simulation for solving the stochastic flow shop problem. In *Proceedings of the 2011 ALIO/EURO Workshop,* Porto, Portugal: 127-130.

Pinedo, M. (ed.) (2008). *Scheduling theory, algorithms and systems.* Springer, New York.

Ramadurai, V.; and Sichitiu, M. (2003). Localization in wireless sensor networks: a probabilistic approach. In: *International conference on wireless networks (ICWN03)*: 275-281.

Reeves, C. (2010). Genetic Algorithms. In, Gendreau, M.; and Potvin, J. (eds.), *Handbook of metaheuristics. International series in operations research management science,* 146, 2nd edition, Kluwer Academic, Dordrecht: 109-140.

Reghioui, M.; Prins, C.; and Labadi, N. (2007) GRASP with path relinking for the capacitated arc routing problem with time windows. *Lecture Notes in Computer Science*, 4448: 722-731.

Resende, M.; and Werneck, R. (2006). A hybrid heuristic for the p-median problem. *Journal of heuristics*, 10(1): 59-88.

Resende, M.; and Werneck, R. (2006). A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, 174(2006): 54-68.

Ryden, M.; Chandra, A.; and Weissman, J. (2013). Nebula: Data intensive computing over widely distributed voluntary resources, *Technical Report*, Citeseer.

Santos, L.; Coutinho-Rodrigues, J.; and Current, J. (2009) An improved heuristic for the capacitated arc routing problem. *Computers & Operations Research*, 36(9): 2632-2637.

Schrage, L. (1975). Implicit representation of variable upper bound in liner programming. In *Mathematical Programming Study*, 4: 118-132.

Shang, R.; Wang, J.; Jiao, L.; and Wang, Y. (2014). An improved decomposition-based memetic algorithm for multi-objective capacitated arc routing problem. *Applied Soft Computing*, 19: 343-361.

Shmoys D.; Tardos, E.; and Aardal, K. (1997). Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Sumposium on Theory of Computing*, ACM, New York: 265-274.

Snyder, L. (2006). Facility location under uncertainty: a review. *IIE Transactions*, 38(11): 971-985.

Spielberg, K. (1969). Algorithms for the simple plant location problem with some side considerations. *Operations Research*, 17: 85-111.

Stern, H.; and Dror, M. (1979). Routing electric meter readers, *Computers & Operations Research*, 6(4): 209-233.

Stollsteimer, J. (1961). The effect of technical change and output expansion on the optimum number, size and location of pear marketing facilities in a California pear producing region. *Ph.D. dissertation,* University of California at Berkeley.

Tagmouti, M.; Gendreay, M.; and Potvin, J. (2007). Arc routing problems with time-dependent service costs. *European Journal of Operations Research,* 181(1): 30-39.

Tagmouti, M.; Gendreau, M.; and Potvin, J. (2011). A dynamic capacitated arc routing problem with time-dependent service costs. *Transportation Research Part C*, 19: 20-28.

Talbi, E (ed.) (2009). *Metaheuristics: From Design to Implementation*, Wiley.

Thouin, F.; and Coates, M. (2008). Equipment allocation in video on demand network deployments. *ACM Transaction on Multimedia Computing, Communications and Applications*, 5(1): 1-24.

Toth, P.; and Vigo, D. (2002). *The Vehicle Routing Problem*. SIAM, Phildeplhia, PA, USA.

Ulusoy, G. (1985). The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research,* 22(3): 329-337.

Van Hentenryck, P.; and Bent, R. (2010). *Online Stochastic Combinatorial Optimization.* The MIT Press. Boston. USA.

Verter, V. (2011). Uncapacitated and capacitated facility location problems. In: Eiselt H.; and Marianov V. (eds.), *Principles of Location Science*, Springer: 25-37.

Vieira, J.; Moura, F.; and Viegas, J. (2007). Transport policy and environmental impacts: The importance of multi-instrumentality in policy integration. *Transportation Policy,* 14(5): 421-432.

Vygen, J. (2005). Approximation algorithm for facility location problems *(Lecture Notes). Technical report 05950*, Resarch Institute for Discrete Mathematics, University of Bonn, 2005.

Wang, D.; Wang, D.; Yan, Y.; and Wang, H. (2008). An adaptive verison of parallel MPSO with OpenMP for uncapacitated facility location problem. In *Proceedigns of 2008 Chinese Control and Decision Conference*: 2303-2307.

Welz, S. (1994). Optimal solutions for the capacitated arc routing problem using integer programming. *Masther's thesis, University o Cincinnati*.

Win, Z. (1988). Contributions to routing problems. *Ph. D. thesis*, University of Augsburg, Germany.

Wøhlk, S. (2005). Contributions to arc routing. *Ph. D. thesis,* Faculty of Social Sciences, University of Southern Denmark.

Wøhlk, S. (2008). A decade of capacitated arc routing. In B. Golden, S. Raghavan, E. Wasil, eds. *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces Series*, Springer, Boston: 29-48.

Zachariadis, E.; and Kiranoudis, C. (2011), Local search for the undirected capacitated arc routing problem with profits. *European Journal of Operational Research*, 210: 358-367.

A. Appendix

A.1. Extended results for ARPSD

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
e1-A	3548	3578.7	0.87%	1.00	3562.1	0.40%	-0.46%	0.99	1.00
e1-B	4498	4572.7	1.66%	1.00	4520.6	0.50%	-1.14%	1.00	1.00
e1-C	5632	6388.6	13.43%	0.96	5669.9	0.67%	-11.25%	1.00	1.00
e2-A	5022	5366.5	6.86%	0.97	5087.2	1.30%	-5.20%	1.00	1.00
e2-B	6344	6867.7	8.26%	0.97	6436.5	1.46%	-6.28%	1.00	1.00
e2-C	8477	9674.4	14.13%	0.96	8634.0	1.85%	-10.75%	1.00	1.00
e3-A	5924	7121.9	20.22%	0.93	6051.6	2.15%	-15.03%	1.00	1.00
e3-B	7847	8503.6	8.37%	0.98	7976.8	1.65%	-6.20%	1.00	1.00
e3-C	10386	11693.9	12.59%	0.97	10641.4	2.46%	-9.00%	0.99	1.00
e4-A	6504	6710.5	3.17%	0.99	6622.6	1.82%	-1.31%	0.98	1.00
e4-B	9120	9303.4	2.01%	1.00	9311.9	2.10%	0.09%	1.00	1.00
e4-C	11886	12225.5	2.86%	0.99	12018.9	1.12%	-1.69%	1.00	1.00
s1-A	5018	5887.3	17.32%	0.95	5130.0	2.23%	-12.86%	1.00	1.00
s1-B	6435	6484.9	0.78%	1.00	6484.9	0.78%	0.00%	1.00	1.00
s1-C	8518	9433.0	10.74%	0.97	8692.0	2.04%	-7.86%	1.00	0.99
s2-A	10076	11747.6	16.59%	0.93	10433.6	3.55%	-11.19%	1.00	0.99
s2-B	13356	13951.9	4.46%	0.99	13710.6	2.65%	-1.73%	1.00	1.00
s2-C	16752	18229.4	8.82%	0.97	17279.8	3.15%	-5.21%	0.98	1.00
s3-A	10478	11457.1	9.34%	0.98	10696.1	2.08%	-6.64%	0.99	1.00
s3-B	13986	15981.4	14.27%	0.97	14525.3	3.86%	-9.11%	0.99	1.00
s3-C	17538	19300.7	10.05%	0.96	18222.2	3.90%	-5.59%	1.00	0.99
s4-A	12647	15105.4	19.44%	0.95	13011.9	2.89%	-13.86%	0.99	1.00
s4-B	16693	17971.1	7.66%	0.98	17083.4	2.34%	-4.94%	1.00	1.00
s4-C	21071	22244.9	5.57%	0.99	21682.7	2.90%	-2.53%	0.97	1.00
Avg.			*9.27%*	*0.97*		*2.41%*	*-6.28%*	*0.995*	*1.00*

Table 20. Results for *egl* dataset when $Var[Q_i] = 0.05 \cdot E[Q_i]$

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
gdb1	316	541.1	71.23%	0.73	482.0	52.53%	-2.01%	0.95	0.87
gdb2	339	598.5	76.55%	0.73	517.7	52.71%	-7.06%	0.99	0.87
gdb3	275	474.1	72.40%	0.85	424.9	54.51%	-3.93%	0.98	0.87
gdb4	287	526.0	83.28%	0.98	466.1	62.40%	-5.59%	0.82	0.87
gdb5	377	628.6	66.74%	1.00	599.1	58.91%	-5.79%	0.93	0.86
gdb6	298	521.4	74.97%	0.88	469.7	57.62%	-1.92%	0.85	0.82
gdb7	325	561.7	72.83%	0.92	521.0	60.31%	-5.39%	0.90	0.82
gdb8	350	467.2	33.49%	0.82	397.4	13.54%	0.00%	1.00	0.96
gdb9	313	451.1	44.12%	0.98	359.4	14.82%	0.00%	1.00	0.97
gdb10	275	400.7	45.71%	0.78	356.2	29.53%	0.00%	1.00	0.91
gdb11	395	459.2	16.25%	0.90	418.5	5.95%	-1.27%	0.87	0.99
gdb12	468	522.4	11.62%	0.71	485.9	3.82%	-2.08%	0.99	0.99
gdb13	536	546.2	1.90%	0.92	544.9	1.66%	-0.04%	0.96	1.00
gdb14	100	129.2	29.20%	0.92	109.8	9.80%	-0.27%	0.92	0.98
gdb15	58	63.2	8.97%	0.90	58.3	0.52%	0.00%	1.00	0.99
gdb16	127	161.8	27.40%	0.93	133.9	5.43%	-0.37%	0.97	0.98
gdb17	91	94.1	3.41%	0.73	91.1	0.11%	-0.33%	0.99	1.00
gdb18	164	212.1	29.33%	0.69	178.7	8.96%	-1.05%	0.87	0.98
gdb19	55	69.7	26.73%	0.75	59.2	7.64%	0.00%	1.00	0.92
gdb20	121	138.4	14.38%	0.73	124.4	2.81%	-0.08%	0.98	0.99
gdb21	156	175.4	12.44%	0.73	163.9	5.06%	-0.30%	0.86	0.99
gdb22	200	235.2	17.60%	0.73	207.1	3.55%	-0.58%	0.96	0.99
gdb23	233	258.6	10.99%	0.91	243.2	4.38%	-1.06%	0.89	0.99
Avg.			*40.57%*	*0.84*		*26.51%*	*-10.00%*	*0.943*	*0.94*

Table 21. Results for *gdb* dataset when $Var[Q_i] = 0.05 \cdot E[Q_i]$

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
kshs1	14661	14661.0	0.00%	0.83	14661.0	0.00%	0.00%	1.00	1.00
kshs2	9863	9863.0	0.00%	1.00	9863.0	0.00%	0.00%	1.00	1.00
kshs3	9666	9666.0	0.00%	1.00	9666.0	0.00%	0.00%	1.00	1.00
kshs4	11498	11499.2	0.01%	1.00	11499.9	0.02%	0.00%	1.00	1.00
kshs5	10957	10967.2	0.09%	1.00	10959.2	0.02%	0.00%	1.00	1.00
kshs6	10197	10203.6	0.06%	1.00	10197.0	0.00%	-0.02%	0.96	1.00
Avg.			*0.03%*	*0.97*		*0.01%*	*-0.02%*	*0.993*	*1.00*

Table 22. Results for *kshs* dataset when $Var[Q_i] = 0.05 \cdot E[Q_i]$

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
val1A	173	184.1	6.42%	1.00	184.1	6.42%	-5.49%	1.00	1.00
val1B	175	186.9	6.80%	0.93	186.9	6.80%	-1.02%	1.00	0.93
val1C	247	276.0	11.74%	0.91	276.0	11.74%	-7.54%	1.00	0.91
val2A	229	234.3	2.31%	0.85	234.3	2.31%	-0.13%	1.00	0.85
val2B	260	296.1	13.88%	0.89	296.1	13.88%	-10.44%	0.98	0.89
val2C	462	501.5	8.55%	0.94	501.5	8.55%	-2.31%	0.98	0.94
val3A	81	99.8	23.21%	1.00	99.8	23.21%	-14.73%	0.94	1.00
val3B	87	122.2	40.46%	0.96	122.2	40.46%	-18.33%	0.98	0.96
val3C	139	209.4	50.65%	1.00	209.4	50.65%	-17.57%	0.93	1.00
val4A	404	422.4	4.55%	0.94	422.4	4.55%	-1.92%	0.98	0.94
val4B	424	481.2	13.49%	0.97	481.2	13.49%	-8.87%	0.98	0.97
val4C	438	505.5	15.41%	0.81	505.5	15.41%	-6.49%	0.99	0.81
val4D	543	737.9	35.89%	0.80	737.9	35.89%	-21.32%	0.99	0.80
val5A	427	443.8	3.93%	0.83	443.8	3.93%	0.00%	0.99	0.83
val5B	450	478.3	6.29%	0.93	478.3	6.29%	-1.40%	0.98	0.93
val5C	485	535.5	10.41%	0.88	535.5	10.41%	-5.30%	0.98	0.88
val5D	594	679.1	14.33%	0.87	679.1	14.33%	-6.29%	0.99	0.87
val6A	225	231.0	2.67%	0.85	231.0	2.67%	-0.65%	1.00	0.85
val6B	233	241.6	3.69%	0.99	241.6	3.69%	-1.08%	0.98	0.99
val6C	321	342.4	6.67%	0.96	342.4	6.67%	-2.25%	1.00	0.96
val7A	279	289.6	3.80%	0.95	289.6	3.80%	-3.63%	1.00	0.95
val7B	286	308.6	7.90%	0.95	308.6	7.90%	-5.99%	1.00	0.95
val7C	342	393.5	15.06%	0.98	393.5	15.06%	-8.87%	0.99	0.98
val8A	391	415.0	6.14%	0.98	415.0	6.14%	-1.90%	0.99	0.98
val8B	406	443.3	9.19%	0.97	443.3	9.19%	-2.96%	0.98	0.97
val8C	541	650.1	20.17%	1.00	650.1	20.17%	-9.71%	0.99	1.00
val9A	326	344.0	5.52%	0.94	344.0	5.52%	-4.24%	1.00	0.94
val9B	333	360.7	8.32%	0.95	360.7	8.32%	-4.57%	0.97	0.95
val9C	341	370.3	8.59%	0.95	370.3	8.59%	-2.97%	0.99	0.95
val9D	402	516.6	28.51%	0.93	516.6	28.51%	-14.00%	0.97	0.93
val10A	435	454.9	4.57%	0.95	454.9	4.57%	-2.07%	0.96	0.95
val10B	447	481.5	7.72%	0.94	481.5	7.72%	-4.15%	1.00	0.94
val10C	459	524.7	14.31%	0.91	524.7	14.31%	-8.06%	1.00	0.91
val10D	543	671.8	23.72%	0.90	671.8	23.72%	-11.51%	0.97	0.90
Avg.			*12.62%*	*0.93*		*5.15%*	*-6.64%*	*0.942*	*0.99*

Table 23. Results for *val* dataset when $Var[Q_i] = 0.05 \cdot E[Q_i]$

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
e1-A	3548	4050.7	14.17%	0.94	3648.4	2.83%	-9.93%	1.00	0.99
e1-B	4498	4601.9	2.31%	0.99	4601.9	2.31%	0.00%	1.00	0.99
e1-C	5632	7061.8	25.39%	0.93	5742.7	1.97%	-18.68%	1.00	1.00
e2-A	5022	6141.8	22.30%	0.93	5288.8	5.31%	-13.89%	1.00	0.98
e2-B	6344	8209.7	29.41%	0.90	6647.0	4.78%	-19.03%	0.94	1.00
e2-C	8477	10785.4	27.23%	0.93	8866.6	4.60%	-17.79%	0.99	0.99
e3-A	5924	8281.6	39.80%	0.84	6397.9	8.00%	-22.75%	1.00	0.99
e3-B	7847	9455.0	20.49%	0.95	8286.0	5.59%	-12.36%	1.00	0.99
e3-C	10386	12680.9	22.10%	0.93	11110.4	6.97%	-12.38%	0.99	0.99
e4-A	6504	7933.9	21.98%	0.91	6951.1	6.87%	-12.39%	0.99	0.99
e4-B	9120	12862.7	41.04%	0.88	9721.7	6.60%	-24.42%	0.98	1.00
e4-C	11886	13461.4	13.25%	0.96	12697.0	6.82%	-5.68%	1.00	0.98
s1-A	5018	7699.9	53.45%	0.80	5403.0	7.67%	-29.83%	0.98	0.99
s1-B	6435	8434.7	31.08%	0.91	6881.0	6.93%	-18.42%	0.99	0.98
s1-C	8518	10176.2	19.47%	0.94	9208.1	8.10%	-9.51%	0.96	0.99
s2-A	10076	15242.6	51.28%	0.82	11349.4	12.64%	-25.54%	1.00	0.96
s2-B	13356	16598.4	24.28%	0.92	14525.3	8.75%	-12.49%	0.99	0.99
s2-C	16752	20389.6	21.71%	0.92	18433.9	10.04%	-9.59%	0.97	0.98
s3-A	10478	12657.9	20.80%	0.94	11260.7	7.47%	-11.04%	0.97	0.98
s3-B	13986	19262.1	37.72%	0.89	15425.4	10.29%	-19.92%	1.00	0.96
s3-C	17538	20320.8	15.87%	0.95	19561.4	11.54%	-3.74%	0.98	0.98
s4-A	12647	16466.4	30.20%	0.91	13704.7	8.36%	-16.77%	0.96	0.99
s4-B	16693	20151.9	20.72%	0.94	18330.8	9.81%	-9.04%	0.94	0.99
s4-C	21071	26170.5	24.20%	0.93	23125.9	9.75%	-11.63%	0.98	0.98
Avg.			*25.80%*	*0.91*		*8.17%*	*-14.02%*	*0.984*	*0.99*

Table 24. Results for *egl* dataset when $Var[Q_i] = 0.25 \cdot E[Q_i]$

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
gdb1	316	514.7	62.88%	0.75	491.1	55.41%	-4.59%	0.98	0.82
gdb2	339	578.0	70.50%	0.65	526.8	55.40%	-8.86%	0.99	0.86
gdb3	275	450.1	63.67%	0.77	433.9	57.78%	-3.60%	0.98	0.87
gdb4	287	488.9	70.35%	0.94	474.1	65.19%	-3.03%	0.99	0.87
gdb5	377	662.6	75.76%	0.94	605.3	60.56%	-8.65%	0.95	0.82
gdb6	298	506.1	69.83%	0.79	476.5	59.90%	-5.85%	0.98	0.82
gdb7	325	542.8	67.02%	0.86	531.7	63.60%	-2.04%	0.95	0.82
gdb8	350	596.3	70.37%	0.83	490.8	40.23%	-17.69%	0.95	0.95
gdb9	313	559.8	78.85%	0.86	438.9	40.22%	-21.60%	0.95	0.93
gdb10	275	446.7	62.44%	0.76	389.8	41.75%	-12.74%	0.95	0.86
gdb11	395	521.8	32.10%	0.84	463.8	17.42%	-11.12%	0.94	0.91
gdb12	468	577.1	23.31%	0.74	547.2	16.92%	-5.18%	0.95	0.97
gdb13	536	613.6	14.48%	0.84	574.8	7.24%	-6.32%	0.99	0.96
gdb14	100	146.7	46.70%	0.85	125.7	25.70%	-14.31%	0.96	0.91
gdb15	58	68.0	17.24%	0.83	63.4	9.31%	-6.76%	0.99	0.91
gdb16	127	166.2	30.87%	0.81	145.8	14.80%	-12.27%	0.95	0.93
gdb17	91	102.2	12.31%	0.75	95.8	5.27%	-6.26%	0.99	0.97
gdb18	164	231.2	40.98%	0.71	202.8	23.66%	-12.28%	0.97	0.91
gdb19	55	71.6	30.18%	0.74	66.4	20.73%	-7.26%	1.00	0.90
gdb20	121	158.1	30.66%	0.75	135.5	11.98%	-14.29%	0.96	0.95
gdb21	156	196.4	25.90%	0.75	182.1	16.73%	-7.28%	0.99	0.93
gdb22	200	252.0	26.00%	0.81	221.2	10.60%	-12.22%	0.91	0.94
gdb23	233	298.1	27.94%	0.72	266.6	14.42%	-10.57%	0.95	0.93
Avg.			*49.33%*	*0.80*		*35.69%*	*-9.13%*	*0.966*	*0.90*

Table 25. Results for *gdb* dataset when $Var[Q_i] = 0.25 \cdot E[Q_i]$

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
kshs1	14661	14834.5	1.18%	1.00	14675.7	0.10%	-1.07%	1.00	1.00
kshs2	9863	9868.2	0.05%	1.00	9870.5	0.08%	0.02%	0.98	1.00
kshs3	9666	9666.4	0.00%	1.00	9666.1	0.00%	0.00%	0.96	1.00
kshs4	11498	11507.9	0.09%	1.00	11506.1	0.07%	-0.02%	1.00	1.00
kshs5	10957	13463.9	22.88%	0.86	11377.7	3.84%	-15.49%	1.00	0.97
kshs6	10197	10216.1	0.19%	1.00	10212.3	0.15%	-0.04%	0.94	1.00
Avg.			*4.06%*	*0.97*		*0.70%*	*-3.23%*	*0.980*	*1.00*

Table 26. Results for *kshs* dataset when $Var[Q_i] = 0.25 \cdot E[Q_i]$

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
val1A	173	192.2	11.10%	0.80	177.2	2.43%	-7.80%	0.99	1.00
val1B	175	206.1	17.77%	0.80	188.1	7.49%	-8.73%	1.00	0.99
val1C	247	316.7	28.22%	0.77	288.4	16.76%	-8.94%	0.93	0.97
val2A	229	301.4	31.62%	0.79	242.8	6.03%	-19.44%	0.96	0.94
val2B	260	347.4	33.62%	0.86	292.3	12.42%	-15.86%	0.89	0.94
val2C	462	648.3	40.32%	0.94	583.6	26.32%	-9.98%	0.98	0.94
val3A	81	103.9	28.27%	0.91	93.6	15.56%	-9.91%	0.98	0.96
val3B	87	141.9	63.10%	0.85	117.4	34.94%	-17.27%	0.85	0.91
val3C	139	266.9	92.01%	0.86	229.6	65.18%	-13.98%	0.78	0.91
val4A	404	450.2	11.44%	0.87	428.7	6.11%	-4.78%	0.77	0.99
val4B	424	490.3	15.64%	0.77	460.4	8.58%	-6.10%	0.78	0.99
val4C	438	657.3	50.07%	0.72	495.4	13.11%	-24.63%	0.85	0.96
val4D	543	853.4	57.16%	0.70	645.3	18.84%	-24.38%	0.91	0.95
val5A	427	504.0	18.03%	0.84	461.9	8.17%	-8.35%	0.96	0.95
val5B	450	581.7	29.27%	0.84	492.9	9.53%	-15.27%	0.84	0.99
val5C	485	657.0	35.46%	0.71	550.0	13.40%	-16.29%	0.88	0.97
val5D	594	884.8	48.96%	0.79	727.4	22.46%	-17.79%	0.88	0.98
val6A	225	258.8	15.02%	0.84	235.5	4.67%	-9.00%	1.00	1.00
val6B	233	286.1	22.79%	0.83	254.7	9.31%	-10.98%	1.00	0.99
val6C	321	415.7	29.50%	0.79	382.7	19.22%	-7.94%	0.89	0.97
val7A	279	298.0	6.81%	0.83	286.6	2.72%	-3.83%	0.89	0.97
val7B	286	359.4	25.66%	0.83	297.6	4.06%	-17.20%	0.92	0.98
val7C	342	454.9	33.01%	0.84	404.9	18.39%	-10.99%	0.99	0.93
val8A	391	471.6	20.61%	0.88	416.7	6.57%	-11.64%	0.97	0.98
val8B	406	497.8	22.61%	0.96	451.9	11.31%	-9.22%	0.85	0.97
val8C	541	758.2	40.15%	0.79	673.7	24.53%	-11.14%	0.90	0.96
val9A	326	374.2	14.79%	0.88	338.3	3.77%	-9.59%	0.91	0.99
val9B	333	387.1	16.25%	0.83	357.7	7.42%	-7.59%	0.94	0.97
val9C	341	425.5	24.78%	0.82	374.6	9.85%	-11.96%	0.97	0.96
val9D	402	560.3	39.38%	0.83	496.3	23.46%	-11.42%	0.81	0.97
val10A	435	494.3	13.63%	0.78	454.2	4.41%	-8.11%	0.96	0.98
val10B	447	540.1	20.83%	0.81	472.7	5.75%	-12.48%	0.99	0.97
val10C	459	567.7	23.68%	0.80	503.4	9.67%	-11.33%	0.76	0.98
val10D	543	792.9	46.02%	0.81	665.7	22.60%	-16.04%	0.83	0.96
Avg.			*30.33%*	*0.82*		*13.53%*	*-12.89%*	*0.906*	*0.97*

Table 27. Results for *val* dataset when $Var[Q_i] = 0.25 \cdot E[Q_i]$

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
e1-A	3548	4695.7	32.35%	0.87	3803.4	7.20%	-19.00%	0.98	0.99
e1-B	4498	4958.8	10.24%	0.97	4833.5	7.46%	-2.53%	1.00	0.98
e1-C	5632	6629.5	17.71%	0.94	6212.6	10.31%	-6.29%	0.99	0.98
e2-A	5022	7260.1	44.57%	0.83	5708.3	13.67%	-21.37%	0.96	0.98
e2-B	6344	9565.6	50.78%	0.86	7117.6	12.19%	-25.59%	0.96	0.98
e2-C	8477	12362.4	45.83%	0.88	9457.9	11.57%	-23.49%	0.93	0.99
e3-A	5924	7723.8	30.38%	0.89	6656.3	12.36%	-13.82%	0.94	0.99
e3-B	7847	11217.7	42.96%	0.86	8958.0	14.16%	-20.14%	1.00	0.96
e3-C	10386	13753.2	32.42%	0.90	12044.5	15.97%	-12.42%	0.99	0.96
e4-A	6504	10187.1	56.63%	0.80	7456.0	14.64%	-26.81%	0.95	0.98
e4-B	9120	12116.3	32.85%	0.89	10611.2	16.35%	-12.42%	1.00	0.96
e4-C	11886	14871.9	25.12%	0.92	13991.2	17.71%	-5.92%	1.00	0.95
s1-A	5018	7643.1	52.31%	0.79	5871.7	17.01%	-23.18%	0.88	0.98
s1-B	6435	8896.8	38.26%	0.89	7497.8	16.52%	-15.72%	1.00	0.95
s1-C	8518	11922.6	39.97%	0.88	10442.9	22.60%	-12.41%	1.00	0.94
s2-A	10076	15880.5	57.61%	0.79	11736.8	16.48%	-26.09%	0.93	0.98
s2-B	13356	18779.7	40.61%	0.85	16119.8	20.69%	-14.16%	1.00	0.93
s2-C	16752	23492.6	40.24%	0.88	19866.6	18.59%	-15.43%	0.91	0.98
s3-A	10478	17002.1	62.26%	0.78	12113.0	15.60%	-28.76%	0.88	0.98
s3-B	13986	19824.0	41.74%	0.87	17547.8	25.47%	-11.48%	1.00	0.91
s3-C	17538	25061.4	42.90%	0.88	21263.8	21.24%	-15.15%	0.95	0.95
s4-A	12647	18372.6	45.27%	0.84	14789.3	16.94%	-19.50%	0.90	0.98
s4-B	16693	25140.3	50.60%	0.86	19867.0	19.01%	-20.98%	0.91	0.97
s4-C	21071	29573.2	40.35%	0.89	25774.1	22.32%	-12.85%	0.93	0.96
Avg.			41.71%	0.87		*17.66%*	*-16.97%*	*0.958*	*0.96*

Table 28. Results for *egl* dataset when $Var[Q_i] = 0.75 \cdot E[Q_i]$

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
gdb1	316	501.2	58.61%	0.77	471.5	49.21%	-5.93%	0.95	0.83
gdb2	339	546.4	61.18%	0.79	516.0	52.21%	-5.56%	0.97	0.87
gdb3	275	449.5	63.45%	0.70	428.5	55.82%	-4.67%	0.95	0.88
gdb4	287	481.3	67.70%	0.87	463.3	61.43%	-3.74%	1.00	0.74
gdb5	377	613.7	62.79%	0.86	593.3	57.37%	-3.32%	0.98	0.83
gdb6	298	475.6	59.60%	0.78	459.6	54.23%	-3.36%	0.92	0.84
gdb7	325	524.1	61.26%	0.80	514.6	58.34%	-1.81%	0.85	0.84
gdb8	350	639.3	82.66%	0.75	563.9	61.11%	-11.79%	0.89	0.87
gdb9	313	622.6	98.91%	0.87	506.6	61.85%	-18.63%	0.76	0.91
gdb10	275	409.4	48.87%	0.70	391.2	42.25%	-4.45%	0.82	0.86
gdb11	395	574.3	45.39%	0.76	509.5	28.99%	-11.28%	0.85	0.92
gdb12	468	686.3	46.65%	0.78	629.1	34.42%	-8.33%	0.98	0.92
gdb13	536	718.0	33.96%	0.69	620.2	15.71%	-13.62%	0.88	0.94
gdb14	100	147.2	47.20%	0.72	136.4	36.40%	-7.34%	0.82	0.91
gdb15	58	74.4	28.28%	0.76	68.8	18.62%	-7.53%	0.85	0.90
gdb16	127	188.3	48.27%	0.73	154.7	21.81%	-17.84%	0.81	0.92
gdb17	91	106.4	16.92%	0.77	101.6	11.65%	-4.51%	0.85	0.94
gdb18	164	240.3	46.52%	0.74	218.9	33.48%	-8.91%	0.79	0.90
gdb19	55	89.2	62.18%	0.78	71.8	30.55%	-19.51%	0.98	0.93
gdb20	121	178.7	47.69%	0.77	144.5	19.42%	-19.14%	0.91	0.90
gdb21	156	221.2	41.79%	0.77	195.0	25.00%	-11.84%	0.81	0.93
gdb22	200	268.1	34.05%	0.76	234.2	17.10%	-12.64%	0.82	0.91
gdb23	233	323.3	38.76%	0.66	285.9	22.70%	-11.57%	0.77	0.92
Avg.			*54.95%*	*0.76*		*41.31%*	*-8.81%*	*0.879*	*0.89*

Table 29. Results for *gdb* dataset when $Var[Q_i] = 0.75 \cdot E[Q_i]$

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
kshs1	14661	15182.6	3.56%	0.93	14780.4	0.81%	-2.65%	1.00	1.00
kshs2	9863	10038.0	1.77%	0.99	9947.8	0.86%	-0.90%	1.00	1.00
kshs3	9666	9721.1	0.57%	1.00	9676.1	0.10%	-0.46%	0.98	1.00
kshs4	11498	11693.7	1.70%	0.99	11682.2	1.60%	-0.10%	1.00	0.99
kshs5	10957	14592.4	33.18%	0.85	11919.3	8.78%	-18.32%	0.87	1.00
kshs6	10197	10402.4	2.01%	0.99	10314.8	1.16%	-0.84%	0.99	0.99
Avg.			*7.16%*	*0.96*		*2.21%*	*-4.62%*	*0.973*	*0.97*

Table 30. Results for *kshs* dataset when $Var[Q_i] = 0.75 \cdot E[Q_i]$

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
val1A	173	205.8	18.96%	0.71	179.1	3.53%	-12.97%	0.80	0.99
val1B	175	210.7	20.40%	0.73	201.6	15.20%	-4.32%	0.88	0.94
val1C	247	361.2	46.23%	0.72	331.2	34.09%	-8.31%	0.89	0.94
val2A	229	286.8	25.24%	0.72	267.3	16.72%	-6.80%	0.96	0.86
val2B	260	382.9	47.27%	0.80	322.6	24.08%	-15.75%	0.83	0.97
val2C	462	801.7	73.53%	0.90	715.9	54.96%	-10.70%	0.94	0.90
val3A	81	119.1	47.04%	0.85	104.3	28.77%	-12.43%	0.82	0.91
val3B	87	152.1	74.83%	0.84	140.2	61.15%	-7.82%	0.77	0.84
val3C	139	285.7	105.54%	0.81	249.2	79.28%	-12.78%	0.76	0.87
val4A	404	509.3	26.06%	0.77	444.1	9.93%	-12.80%	0.86	0.96
val4B	424	569.7	34.36%	0.71	484.5	14.27%	-14.96%	0.76	0.98
val4C	438	655.4	49.63%	0.66	547.6	25.02%	-16.45%	0.81	0.95
val4D	543	908.2	67.26%	0.65	752.7	38.62%	-17.12%	0.81	0.94
val5A	427	547.3	28.17%	0.75	478.7	12.11%	-12.53%	0.76	0.97
val5B	450	647.8	43.96%	0.72	533.6	18.58%	-17.63%	0.87	0.94
val5C	485	698.9	44.10%	0.68	607.2	25.20%	-13.12%	0.76	0.96
val5D	594	963.2	62.15%	0.78	828.2	39.43%	-14.02%	0.78	0.94
val6A	225	291.6	29.60%	0.78	243.3	8.13%	-16.56%	0.95	0.97
val6B	233	319.2	37.00%	0.74	272.7	17.04%	-14.57%	0.86	0.93
val6C	321	510.2	58.94%	0.75	448.1	39.60%	-12.17%	0.79	0.95
val7A	279	305.5	9.50%	0.78	295.3	5.84%	-3.34%	0.75	0.98
val7B	286	354.2	23.85%	0.76	319.2	11.61%	-9.88%	0.96	0.92
val7C	342	519.3	51.84%	0.72	464.1	35.70%	-10.63%	0.81	0.93
val8A	391	505.6	29.31%	0.81	436.3	11.59%	-13.71%	0.75	0.96
val8B	406	546.9	34.70%	0.90	492.8	21.38%	-9.89%	0.85	0.92
val8C	541	951.6	75.90%	0.85	788.6	45.77%	-17.13%	0.86	0.91
val9A	326	361.7	10.95%	0.81	345.2	5.89%	-4.56%	0.77	0.96
val9B	333	428.6	28.71%	0.74	374.3	12.40%	-12.67%	0.75	0.96
val9C	341	457.6	34.19%	0.75	403.2	18.24%	-11.89%	0.84	0.97
val9D	402	650.2	61.74%	0.73	570.4	41.89%	-12.27%	0.75	0.92
val10A	435	514.6	18.30%	0.90	466.4	7.22%	-9.37%	0.99	0.95
val10B	447	556.3	24.45%	0.74	505.6	13.11%	-9.11%	0.79	0.93
val10C	459	626.9	36.58%	0.73	540.2	17.69%	-13.83%	0.76	0.94
val10D	543	871.6	60.52%	0.73	761.0	40.15%	-12.69%	0.79	0.90
Avg.			*43.17%*	*0.77*		*25.04%*	*-12.66%*	*0.826*	*0.89*

Table 31. Results for *val* dataset when $Var[Q_i] = 0.75 \cdot E[Q_i]$

Set	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
	Fixed Cost (1)	Expected Cost (2)	Gap (1)-(2)	Reli-ability	Expected Cost (3)	Gap (1)-(3)	Gap (2)-(3)	k	Reli-ability
e1-A	3548	5337.4	50.43%	0.82	4099.6	15.55%	-23.19%	0.95	0.96
e1-B	4498	6791.2	50.98%	0.86	5273.3	17.24%	-22.35%	1.00	0.94
e1-C	5632	7734.2	37.33%	0.90	6819.5	21.08%	-11.83%	0.99	0.96
e2-A	5022	8179.7	62.88%	0.78	6103.5	21.54%	-25.38%	0.90	0.95
e2-B	6344	10299.7	62.35%	0.81	7908.6	24.66%	-23.22%	0.97	0.93
e2-C	8477	13556.3	59.92%	0.82	10479.6	23.62%	-22.70%	0.96	0.94
e3-A	5924	9606.1	62.16%	0.78	7323.6	23.63%	-23.76%	0.81	0.98
e3-B	7847	10631.8	35.49%	0.87	10029.9	27.82%	-5.66%	1.00	0.91
e3-C	10386	15948.7	53.56%	0.86	13148.1	26.59%	-17.56%	0.96	0.95
e4-A	6504	10246.4	57.54%	0.78	8070.6	24.09%	-21.23%	0.91	0.97
e4-B	9120	15899.0	74.33%	0.78	11672.0	27.98%	-26.59%	0.99	0.92
e4-C	11886	18596.6	56.46%	0.85	15357.9	29.21%	-17.42%	0.97	0.93
s1-A	5018	8545.8	70.30%	0.76	6558.3	30.70%	-23.26%	0.94	0.93
s1-B	6435	9802.4	52.33%	0.80	8492.2	31.97%	-13.37%	0.95	0.93
s1-C	8518	14835.1	74.16%	0.79	11637.2	36.62%	-21.56%	0.99	0.90
s2-A	10076	17859.2	77.24%	0.74	12786.3	26.90%	-28.40%	0.86	0.96
s2-B	13356	22263.4	66.69%	0.77	17762.5	32.99%	-20.22%	0.98	0.93
s2-C	16752	27749.0	65.65%	0.81	22873.4	36.54%	-17.57%	0.91	0.93
s3-A	10478	17770.4	69.60%	0.77	14164.0	35.18%	-20.29%	0.98	0.86
s3-B	13986	22534.1	61.12%	0.80	18908.7	35.20%	-16.09%	0.90	0.93
s3-C	17538	29729.8	69.52%	0.82	24320.6	38.67%	-18.19%	0.90	0.93
s4-A	12647	21239.4	67.94%	0.78	16455.8	30.12%	-22.52%	0.86	0.95
s4-B	16693	26434.8	58.36%	0.82	22625.9	35.54%	-14.41%	0.91	0.92
s4-C	21071	35151.6	66.82%	0.81	29021.4	37.73%	-17.44%	0.88	0.94
Avg.			*62.66%*	*0.81*		*31.18%*	*-19.35%*	*0.936*	*0.89*

Table 32. Results for *egl* dataset when $Var[Q_i] = 2 \cdot E[Q_i]$

Set	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
	Fixed Cost (1)	Expected Cost (2)	Gap (1)-(2)	Reli-ability	Expected Cost (3)	Gap (1)-(3)	Gap (2)-(3)	k	Reli-ability
gdb1	316	474.8	50.25%	0.80	458.5	45.09%	-3.43%	0.96	0.85
gdb2	339	527.8	55.69%	0.71	507.4	49.68%	-3.87%	0.81	0.88
gdb3	275	429.7	56.25%	0.67	414.0	50.55%	-3.65%	0.98	0.86
gdb4	287	465.4	62.16%	0.77	443.4	54.49%	-4.73%	1.00	0.77
gdb5	377	590.8	56.71%	0.79	569.4	51.03%	-3.62%	1.00	0.80
gdb6	298	460.6	54.56%	0.74	445.0	49.33%	-3.39%	0.88	0.86
gdb7	325	509.8	56.86%	0.76	496.7	52.83%	-2.57%	1.00	0.80
gdb8	350	662.5	89.29%	0.77	594.0	69.71%	-10.34%	0.82	0.88
gdb9	313	618.2	97.51%	0.73	536.5	71.41%	-13.22%	0.94	0.84
gdb10	275	407.5	48.18%	0.68	381.8	38.84%	-6.31%	0.77	0.90
gdb11	395	599.7	51.82%	0.75	556.7	40.94%	-7.17%	0.87	0.79
gdb12	468	847.9	81.18%	0.80	685.5	46.47%	-19.15%	0.95	0.91
gdb13	536	715.5	33.49%	0.76	651.2	21.49%	-8.99%	0.89	0.91
gdb14	100	159.0	59.00%	0.75	139.9	39.90%	-12.01%	0.81	0.90
gdb15	58	75.8	30.69%	0.73	71.7	23.62%	-5.41%	0.93	0.87
gdb16	127	177.8	40.00%	0.69	160.4	26.30%	-9.79%	0.83	0.92
gdb17	91	119.7	31.54%	0.79	107.4	18.02%	-10.28%	0.77	0.89
gdb18	164	248.2	51.34%	0.77	232.2	41.59%	-6.45%	0.83	0.86
gdb19	55	91.8	66.91%	0.79	75.6	37.45%	-17.65%	0.93	0.91
gdb20	121	176.8	46.12%	0.80	150.6	24.46%	-14.82%	0.78	0.94
gdb21	156	231.2	48.21%	0.79	199.9	28.14%	-13.54%	0.76	0.91
gdb22	200	277.9	38.95%	0.75	242.4	21.20%	-12.77%	0.84	0.88
gdb23	233	340.9	46.31%	0.67	297.0	27.47%	-12.88%	0.78	0.90
Avg.			*57.18%*	*0.75*		*43.66%*	*-8.60%*	*0.877*	*0.85*

Table 33. Results for *gdb* dataset when $Var[Q_i] = 2 \cdot E[Q_i]$

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
kshs1	14661	16794.4	14.55%	0.95	15391.5	4.98%	-8.35%	1.00	0.98
kshs2	9863	10750.0	8.99%	0.97	10181.7	3.23%	-5.29%	0.98	0.99
kshs3	9666	9826.5	1.66%	0.99	9781.0	1.19%	-0.46%	0.85	0.99
kshs4	11498	13019.8	13.24%	0.96	12739.1	10.79%	-2.16%	1.00	0.99
kshs5	10957	14093.8	28.63%	0.86	12159.0	10.97%	-13.73%	0.97	0.99
kshs6	10197	12915.8	26.66%	0.89	10818.2	6.09%	-16.24%	0.93	0.96
Avg.			*15.80%*	*0.94*		*6.33%*	*-8.18%*	*0.955*	*0.97*

Table 34. Results for *kshs* dataset when $Var[Q_i] = 2 \cdot E[Q_i]$

	Solution for the deterministic CARP				Our Best Solution for the ARPSD				
Set	**Fixed Cost (1)**	**Expected Cost (2)**	**Gap (1)-(2)**	**Reli-ability**	**Expected Cost (3)**	**Gap (1)-(3)**	**Gap (2)-(3)**	**k**	**Reli-ability**
val1A	173	208.2	20.35%	0.67	189.6	9.60%	-8.93%	0.80	0.95
val1B	175	254.4	45.37%	0.67	222.6	27.20%	-12.50%	0.92	0.93
val1C	247	454.2	83.89%	0.69	377.9	53.00%	-16.80%	0.78	0.91
val2A	229	317.8	38.78%	0.66	291.8	27.42%	-8.18%	0.83	0.94
val2B	260	416.8	60.31%	0.76	370.0	42.31%	-11.23%	0.94	0.91
val2C	462	897.6	94.29%	0.78	799.2	72.99%	-10.96%	0.95	0.86
val3A	81	121.5	50.00%	0.77	114.9	41.85%	-5.43%	0.79	0.86
val3B	87	156.3	79.66%	0.76	150.5	72.99%	-3.71%	0.89	0.77
val3C	139	288.8	107.77%	0.75	256.3	84.39%	-11.25%	0.85	0.83
val4A	404	553.9	37.10%	0.72	475.6	17.72%	-14.14%	0.77	0.91
val4B	424	589.3	38.99%	0.67	534.5	26.06%	-9.30%	0.83	0.92
val4C	438	721.4	64.70%	0.64	611.3	39.57%	-15.26%	0.78	0.90
val4D	543	1031.6	89.98%	0.65	864.7	59.24%	-16.18%	0.93	0.81
val5A	427	619.2	45.01%	0.66	515.8	20.80%	-16.70%	0.79	0.91
val5B	450	676.4	50.31%	0.70	596.9	32.64%	-11.75%	0.80	0.88
val5C	485	773.5	59.48%	0.65	684.4	41.11%	-11.52%	0.87	0.84
val5D	594	1128.9	90.05%	0.70	967.6	62.90%	-14.29%	0.78	0.88
val6A	225	288.3	28.13%	0.69	262.3	16.58%	-9.02%	0.85	0.91
val6B	233	346.1	48.54%	0.71	306.6	31.59%	-11.41%	0.79	0.91
val6C	321	582.4	81.43%	0.69	511.7	59.41%	-12.14%	0.79	0.88
val7A	279	329.5	18.10%	0.70	309.6	10.97%	-6.04%	0.77	0.95
val7B	286	372.5	30.24%	0.75	340.4	19.02%	-8.62%	0.76	0.92
val7C	342	578.4	69.12%	0.70	531.6	55.44%	-8.09%	0.78	0.90
val8A	391	533.9	36.55%	0.72	480.9	22.99%	-9.93%	0.90	0.88
val8B	406	601.3	48.10%	0.85	542.2	33.55%	-9.83%	0.90	0.85
val8C	541	1017.7	88.11%	0.81	910.4	68.28%	-10.54%	0.78	0.88
val9A	326	411.7	26.29%	0.74	370.1	13.53%	-10.10%	0.86	0.95
val9B	333	459.1	37.87%	0.69	403.4	21.14%	-12.13%	0.86	0.91
val9C	341	486.8	42.76%	0.69	445.7	30.70%	-8.44%	0.77	0.91
val9D	402	741.8	84.53%	0.67	650.1	61.72%	-12.36%	0.81	0.86
val10A	435	549.9	26.41%	0.69	493.2	13.38%	-10.31%	0.76	0.89
val10B	447	587.9	31.52%	0.69	541.0	21.03%	-7.98%	0.77	0.91
val10C	459	643.7	40.24%	0.68	593.4	29.28%	-7.81%	0.75	0.88
val10D	543	940.7	73.24%	0.64	864.2	59.15%	-8.13%	0.82	0.82
Avg.			*56.62%*	*0.71*		*39.00%*	*-11.25%*	*0.824*	*0.88*

Table 35. Results for *val* dataset when $Var[Q_i] = 2 \cdot E[Q_i]$