CITY LOGISTICS SYSTEM DESIGN UNDER COST UNCERTAINTY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

UTKU CAN KUNTER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

JULY 2015

Approval of the thesis:

CITY LOGISTICS SYSTEM DESIGN UNDER COST UNCERTAINTY

submitted by **UTKU CAN KUNTER** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. M. Gülbin Dural Ünver _____________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Murat Köksalan _____________
Head of Department, **Industrial Engineering**

Assoc. Prof. Dr. Cem İyigün _____________
Supervisor, **Industrial Engineering Dept., METU**

Prof. Dr. Haldun Süral _____________
Co-Supervisor, **Industrial Engineering Dept., METU**

Examining Committee Members:

Assoc. Prof. Dr. Zeynep Pelin Bayındır _____________
Industrial Engineering Dept., METU

Assoc. Prof. Dr. Cem İyigün _____________
Industrial Engineering Dept., METU

Prof. Dr. Haldun Süral _____________
Industrial Engineering Dept., METU

Assist. Prof. Dr. Sakine Batun _____________
Industrial Engineering Dept., METU

Assist. Prof. Dr. Özlem Çavuş _____________
Industrial Engineering Dept., Bilkent University

Date: 23.07.2015

I hereby declare that all the information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: UTKU CAN KUNTER

Signature:

ABSTRACT

CITY LOGISTICS SYSTEM DESIGN UNDER COST UNCERTAINTY

Kunter, Utku Can
M.S., Department of Industrial Engineering
Supervisor : Assoc. Prof. Dr. Cem İyigün
Co-Supervisor: Prof. Dr. Haldun Süral

July 2015, 202 pages

City Logistics (CL) is a quickly developing area of research aiming to develop the methods for designing efficient and effective freight distribution networks. We make an extensive review on CL as well as studies related to CL in order to describe the position of CL in the literature. Based on this review, the location and allocation decisions in a CL system under transportation cost uncertainty is analyzed from a strategic point of view. We use Value of Information analysis to compare different formulations of the problem and measure the difference between the facility location problems in the conventional setting and in CL setting in terms of value of information. Next, we propose two solution methods to handle instances of realistic size. First one is a variation of the L-Shaped method, using scenario-group cuts; and second one is an evolutionary algorithm which makes use of an embedded hybrid heuristic to evaluate chromosomes in reasonable time. The methods proposed have significant advantage over standard solvers, especially when solving large instances.

Keywords: City Logistics, Stochastic Programming, Benders Decomposition, Evolutionary Algorithm

ÖZ

MALİYET BELİRSİZLİĞİ ALTINDA KENT LOJİSTİĞİ SİSTEMİ TASARIMI

Kunter, Utku Can
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Cem İyigün
Ortak Tez Yöneticisi : Prof. Dr. Haldun Süral

Temmuz 2015, 202 sayfa

Kent Lojistiği (KL) hızla gelişen bir araştırma alanıdır ve etkin yük taşıma ağlarını oluşturmak için gereken metotların geliştirilmesini amaçlar. KL konusunun literatürdeki yerini belirlemek amacıyla KL ile ilgili çalışmalar üzerine geniş bir literatür taraması yapılmıştır. Bu taramadan yola çıkarak, KL sistemindeki yer seçimi ve atama kararları ulaştırma maliyeti belirsizliği altında stratejik bakış açısıyla incelenmiştir. Bilginin değeri analiziyle bu problemin farklı formülasyonları karşılaştırılarak, KL sistemlerinin bilginin değeri açısından klasik yer seçimi problemlerinden farkı gösterilmiştir. Ayrıca, uygulamada karşılaşılacak büyüklükteki problemlerin çözülebilmesi için iki çözüm metodu önerilmiştir. Bunlardan ilki, L-Şekilli metodun senaryo-grubu kesileri kullanılan bir varyasyonu, ikincisi ise kromozomların makul zamanda değerlendirilmesi için melez bir sezgisel yöntem kullanan bir evrimsel algoritmadır. Önerilen yöntemlerin, özellikle büyük boyutlu problemlerin çözülmesinde standart çözücülere göre önemli yararlar sağladığı görülmüştür.

Anahtar Kelimeler: Kent Lojistiği, Olasılıksal Programlama, Benders Ayrıştırma, Evrimsel Algoritma

To my family,
for their love and unconditional support

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Assoc. Prof. Dr. Cem İyigün and co-supervisor Prof. Dr. Haldun Süral for their guidance, advice, criticism, encouragements and insight throughout the research.

I also would like to thank Assist. Prof. Dr. Sakine Batun, Assist. Prof. Dr. Özlem Çavuş and Assoc. Prof. Dr. Zeynep Pelin Bayındır for their valuable suggestions and comments.

This study is part of Project 113M121 supported by the TÜBİTAK.

This study was supported by the TÜBİTAK under 2211 National Graduate Fellowship Programme.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Large amounts of freight is transported daily from city outskirts into urban centers. Making these operations efficiently and effectively is crucial especially when one considers the vast amount of resources used in the process and the resulting environmental impact. City Logistics (CL) is a quickly developing area of research aiming to develop the methods and tools to design efficient and effective freight distribution networks for urban areas.

To achieve efficiency in freight distribution, one of the widely accepted approaches is to setup cross-docking facilities where the incoming shipments are combined in order to make the deliveries possible with a smaller fleet. These cross-docking facilities are called City Distribution Centers (CDC) or Urban Distribution Centers (UDC). In 1-echelon CL network design, the process of combining delivery loads in order to achieve higher capacity utilization of distribution vehicles are typical consolidation activities. In addition, for such operations to be effective, all supply and demand points in the system must be considered as a whole. The measures aiming to provide effective communication and collaboration of all the stakeholders are called coordination. Consolidation and coordination are the two main tools employed in the city logistics literature.

The related studies in the literature mostly deal with the CL decisions from a deterministic point of view. The deterministic view of the freight distribution system is a step towards obtaining a more complete model of the freight distribution network, since it reduces the problem complexity and allows the solution of larger instances. However, the deterministic approach is not sufficient to represent the real life systems. In order for the models to provide solutions that are applicable in real life, they must include the uncertainty and risk factors inherent in the nature of the real life systems.

Problems with uncertainty and risk factors are studied extensively in the stochastic programming and robust optimization literature. There are many articles that suggest approaches and methods related to our problem. There are, for example, articles dealing with the stochastic/robust facility location problem (Snyder, 2006); however, to the best of our knowledge, there is stochastic or robust optimization study considering a freight distribution network system in a CL setting. This study aims to provide a stochastic freight distribution network design formulation from a strategic CL perspective, along with the solution methods and analysis of the results.

We use a single echelon setting for the network design problem. We believe the 1-echelon setting would be suitable for our purpose, since the focus of our work is on the strategic and tactical decisions. We analyze the decisions to be made in a 1- echelon CL system in three groups: investment decisions, recourse decisions, and routing decisions. Investment decisions are usually strategic decisions such as determining the locations of facilities. Recourse decisions are made on a tactical level, effective in short/medium-term, such as allocating customers to CDCs. Finally the routing decisions take place on an operational level and are subject to change on a daily basis.

We do not try to build a single optimization model that would make these three sets of decisions all together, as the complexity of such a model prohibits us from obtaining optimal solutions for even small size problem instances. Rather, we divide the problem into location-allocation and vehicle routing decisions. Then, we optimize the critical strategic/tactical decisions while leaving the lower level decisions to be made and updated during operations, according to the changes in uncertain parameters.

Higher level decisions are further divided into two: location decisions are made under uncertainty and allocation decisions are made after the parameters under uncertainty can be observed. In the literature, two-stage stochastic programming models allow for such a division of decisions in a dynamic decision making framework. After making a Value of Information (VoI) analysis comparing different approaches that incorporate uncertainty, we found that a two-stage stochastic model provides a significant cost reduction for all instances we considered. The improvement over the much simpler Expected Value Model is large enough to compensate for the computational complexity of making allocation decisions for each scenario. More importantly, transferring allocation decisions to a recourse problem is meaningful for a CL system because customer allocations can be changed periodically according to the system conditions, in contrast to the number and locations of CDCs.

Using a two-stage stochastic approach to deal with uncertainty, it is meaningful for us to select a kind of uncertainty that can be handled by the recourse problem. Among several causes of uncertainty found in the literature, we choose to deal with the travel time/transportation cost uncertainty, which is expected to have a large impact on our decisions in an urban environment. Other causes of uncertainty such as demand amount, service cost, or unit cost etc. are expected to be largely fixed to the values agreed on contracts. Service/unit cost largely depends on the demand amount and constitute a relatively small part of the system's operating costs. On the other hand, transportation cost is a large part of these costs and is deeply affected from the changes on the transportation network. These changes may be occurring daily or infrequently, may be predictable or unpredictable, may be planned or unplanned. Anticipating those changes leads to more realistic plans. For these reasons, incorporating the transportation cost uncertainty in our models seems to be the best choice.

Since standard solvers usually fail to generate even a feasible solution in reasonable time for the large instances of the problem, we evaluate several alternative exact solution methods. Among these methods, L-Shaped method with scenario-group cuts produces the best results. Unfortunately, it fails to converge within the time limit, even though it finds optimal solutions as its best integer solutions. As an alternative method, we develop an evolutionary algorithm that makes use of a hybrid evaluation heuristic. This heuristic reaches the best solution in shorter time than the other two methods and this solution is optimum in most cases where we know an optimal solution. The solutions obtained by the algorithm also usually agree with the L-Shaped method's solutions for the large instances.

After the location-allocation part, we proceed by handling the routing decisions. Due to the fixed allocation decisions made for each scenario, we only have to consider the vehicle routing problem (VRP) for each CDC, thus the solution space is much smaller than it would be in the Multi-Depot VRP. It is possible for the decision maker to solve these small VRP instances, whenever a change in the higher level decisions are made, or a change in the parameters has been observed. Incorporating the routing decisions into the main problem is also desirable. The routing stage would provide much more accurate information on the cost of making different allocation decisions. To do so, we develop an algorithm to handle the routing stage with a CL perspective and we discuss how the algorithm can be implemented within the methods we proposed.

This study offers two main contributions to the related literature. First, we describe the City Logistics literature in terms of its relations and overlaps with other fields in the relevant operations research (OR) and industrial engineering (IE) literature. Second, we develop two efficient solution methods that are able to find solutions for instances of realistic size. In addition to these, we report the value of information in a CL setting under transportation cost uncertainty.

Remaining part of the thesis is organized as follows: In the next section, we first provide an overview of the recent developments in the related research areas, and then present an extensive review of the literature on the strategic decisions. Section 3 considers several formulations of the City Logistics Network Design Problem under Uncertainty and provides their comparison in a Value of Information framework. In section 4, we present the exact solution methods based on Benders Decomposition. In section 5, we present the evolutionary algorithm. Section 6 provides the computational results for each solution method and their assessments. In section 7, we discuss how the routing decisions can be incorporated into the proposed solution methods. Section 8 includes concluding remarks and possible extensions of our study.

CHAPTER 2

LITERATURE REVIEW

Achieving a complete view of the CL systems is difficult without an understanding of the current developments in the related research areas. We reviewed the studies that explicitly address CL as well as the studies that are quite relevant to the decisions of a CL system. In order to find methods and approaches that can be employed in CL setting, we also reviewed studies on stochastic/robust optimization.

To present our findings in a structured way, we group these studies into a few categories. However, most studies usually fall into multiple categories, which make it difficult to obtain a general view of trends in the literature. Using the findings of this extensive review, we first provide a recent overview of the related literature.

2.1 Overview

To be able to identify the trends in the literature, we searched how many articles have been published on each individual subject, such as facility location, fleet management and vehicle routing.

All analyses are made using the Web of Science database. We searched for the specific terms in the titles of articles, such as “city logistics”, “transportation”, “robust” etc., specified the research areas and recorded the number of publications on each year. We also specified some important publications that were selected according to their number of citations.

2.1.1 City Logistics

CL is a relatively new subject. The term “City Logistics” became popular in the late 1990s and 2000s. While there are still few articles that specifically focus on CL, research on this subject increased significantly in recent years. This is mainly due to the CL projects applied in European cities. These projects reveal the necessity for the construction of a CL theory. While the decisions of CL systems are in many ways similar to the widely studied network design problems, there are still some differences that require an exclusive CL focus in new studies.

First, CL focuses on the consolidation of freight and coordination of customers and suppliers. This leads to different parameters of the system to have significant effect on the solution performance. For instance, in the classical stochastic location literature, mostly demand uncertainty is considered, but within a CL point of view, demand uncertainty loses its significance as the factor with the largest impact. Second, CL takes place in urban environments, which has characteristics that must be reflected on the solution methods and test instances. For example, it is not possible to locate facilities on a continuous plane, because the probability of a location to be free is very low. Rather, locations must be selected from a predetermined set of available spaces. Also, CL requires the locations to be on the city boundaries, rather than within the city, which has quite different implications than the traditional way of locating facilities. Lastly, the coordination aspect of CL systems make it possible to relax some assumptions of the classical location studies. For instance, when the state of the transportation network changes, we can update the decisions assigning customers to CDCs. Since the system is managed in a centralized way, making such changes is possible and may bring great benefit in terms of transportation cost.

Figure 2.1 summarizes the number of studies that are published each year from 2005 until 2013. There was no entry matching with the search criteria for the years before 2005. The blue line shows studies having “City Logistics” term on their title and the orange line shows studies that consider uncertainty. Since there are several alternative terms for expressing uncertainty, we searched for all the terms by joining them with “OR” operators (uncertainty OR uncertain OR robust OR stochastic). The research area is specified as “Operations Research Management Science”.

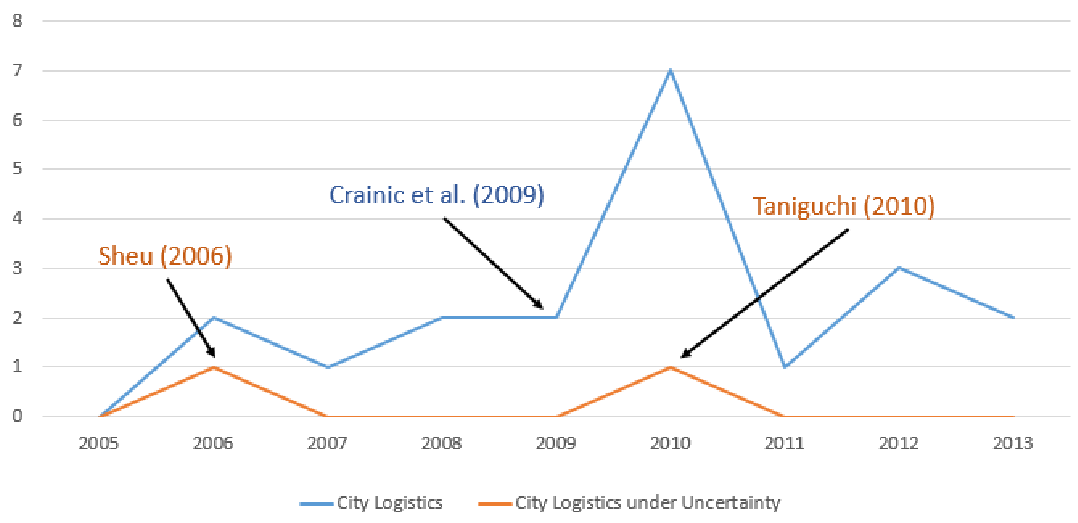


Figure 2.1: Number of Studies on City Logistics in Recent Years (OR&MS)

Crainic et al. (2009) suggest a variety of formulations and solution methods for the deterministic city logistics problem. Due to the high complexity of the formulations, there are no solution methods proposed yet that can consider all decisions at the same time.

When the studies on city logistics under uncertainty are observed, it can be seen that there are only two papers published in the last nine years. These two articles, Sheu (2006) and Taniguchi et al. (2010), are cited 23 times in total.

Although we are mainly interested in the city logistics problems within the ORMS context, we also searched for relevant studies in the CL literature. For example, specifying the research area as "Transportation", we found earlier studies that are mainly concerned about the need for CL systems and about the difficulties of setting up such systems. These articles usually evaluate the solution alternatives from a strategic point of view and do not generally go into the details of how such a system can be constructed. Figure 2.2 shows the number of studies from 2001 to 2013.

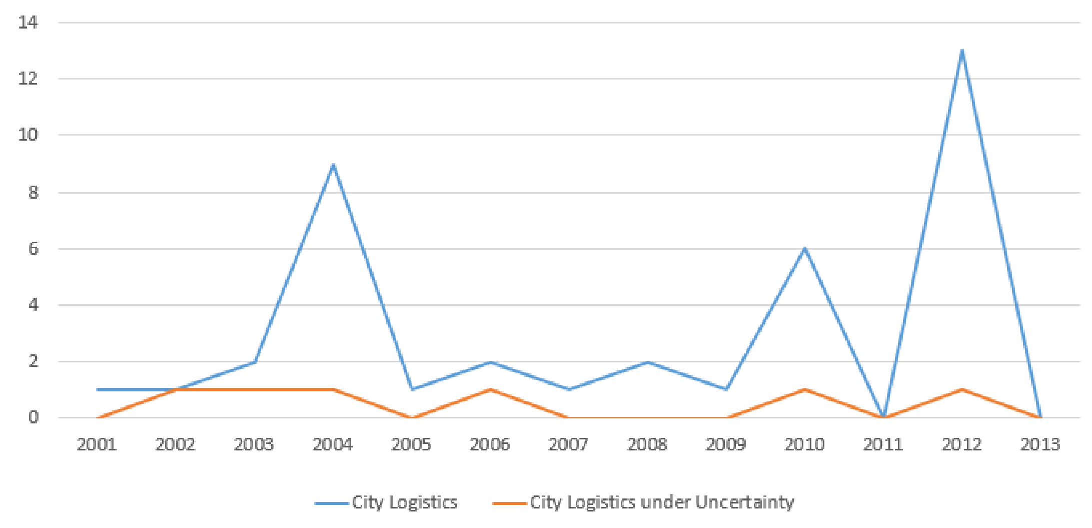


Figure 2.2: Number of Studies on City Logistics in Recent Years (All)

2.1.2 Stochastic/Robust Optimization

We also investigated the use of stochastic and robust optimization in the literature for years after 2001. Stochastic programming literature is considerably larger and more quickly expanding than that of robust optimization, which only recently started to gain popularity due to the higher complexity of robust optimization problems.

While these two approaches towards uncertainty do not generally overlap, it seems that the “robust” and “stochastic” terms are sometimes used interchangeably in the literature. Note that, stochastic programming requires probability distribution of the parameters to be known, so that scenarios can be constructed for different realizations of the parameters. In stochastic programming problems, the objective function usually becomes a weighted average of cost realizations in different scenarios, as the weights being the probabilities of these scenarios. On the other hand, robust optimization does not require probability distributions. It is sufficient to know the upper and lower bounds of the parameters. This way, robust optimization problems mostly try to optimize against the worst-case realization of the parameters under uncertainty.

In Figure 2.3, we provide the number of studies using the terms “Stochastic Programming” and “Robust Optimization”. The research area is specified as “Operations Research Management Science”.

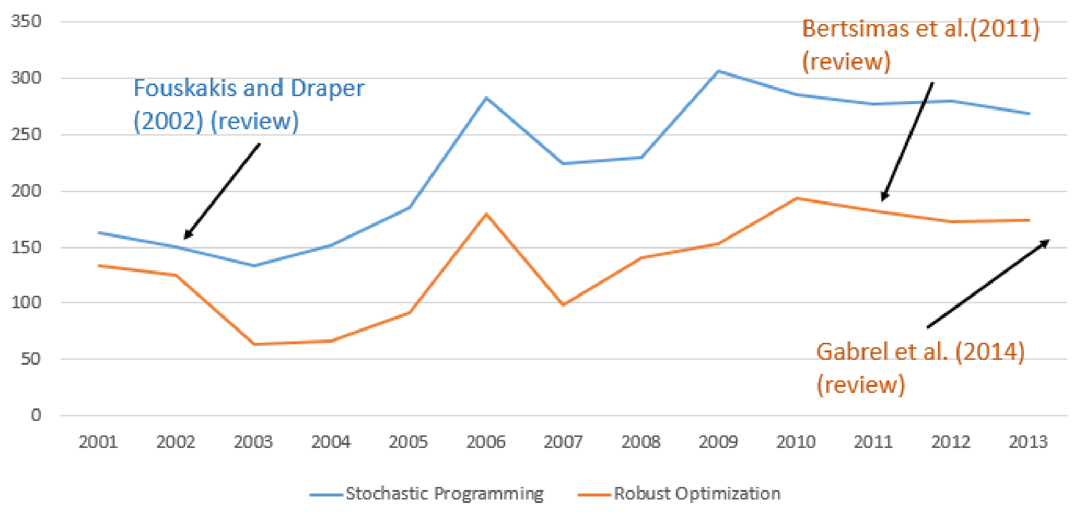


Figure 2.3: Number of Studies on Stochastic Prg. and Robust Opt. in Recent Years

In this area, there are three important review papers. Fouskakis and Draper (2002) is the last comprehensive one on stochastic programming. Since there has been a lot of studies on this subject, the recent reviews choose to focus on only one application area, such as stochastic facility location, stochastic vehicle routing etc.

The other two articles, Bertsimas et al. (2011) and Gabrel et al. (2014a) review the robust optimization literature. Due to the smaller number of papers where robustness is considered, these reviews include a variety of applications in many different areas.

2.1.3 Relevant OR Literature

There are a set of decisions to be made in a CL system design. The following is the list of these decisions:

- Number of city distribution centers (CDC) (strategic level)
- Locations of each CDC (strategic level)
- Number of vehicles to be assigned to each CDC (tactical level)
- Assignment of customers to CDCs (tactical level)
- Vehicle routes to serve the customers (operational level)

Clearly, such a CL system design problem would require many different sub-problems to be solved simultaneously. In our case, these decisions are also subject to uncertainty. Therefore, the complexity of a formulation incorporating all these decisions would be very high. That's why there has been no optimization study yet to address all these decisions at once. Rather, most of the relevant studies consider only a single decision level. Now, we make similar analyses on these studies with respect to the decision level they consider:

- Strategic level: Stochastic Facility Location Problem
- Tactical level: Stochastic Fleet Sizing/Management Problem
- Operational level: Stochastic Vehicle Routing Problem

Stochastic Facility Location Problem

There is a large literature on facility location. However, facility location under uncertainty is not studied that much. Figure 2.4 summarizes the number of facility location articles in recent years. While the literature on the deterministic problems is still growing, we can say that facility location under uncertainty does not draw as much attention. The research area is specified as "Operations Research Management Science" in this search.

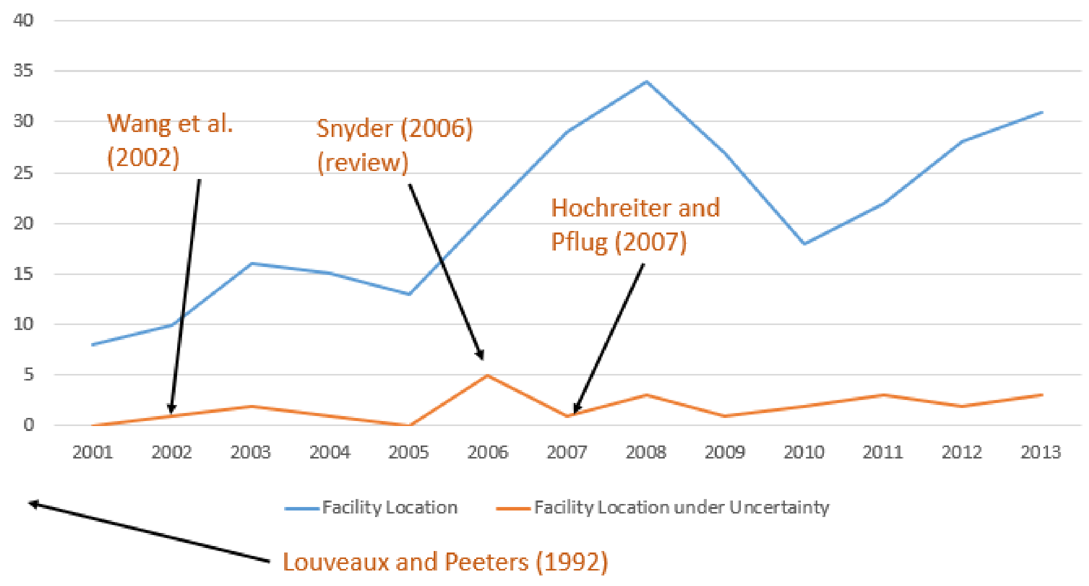


Figure 2.4: Number of Studies on Facility Location in Recent Years

We believe three of these studies are especially important: Louveaux and Peeters (1992) make an important contribution to the stochastic facility location literature with their dual-based procedure. Wang et al. (2002) consider demand uncertainty in a stochastic facility location setting. This is one of the earlier studies that focuses on uncertainty, and proposes three efficient solution methods. Snyder (2006) provides an extensive review of facility location studies that consider uncertainty. These three studies own 44% of the citations (193 out of 444) in the literature of facility location under uncertainty in the specified years.

Fleet Management Problem

There are few studies that consider only the fleet management decision. Nevertheless, Figure 2.5 summarizes the number of studies in the literature on this problem. While searching for the relevant articles, the keywords we used were "fleet sizing", "fleet management" and "fleet planning" connected by "OR" operators, since any of these could be used in the titles of the relevant articles. The research area is specified as "Operations Research Management Science".

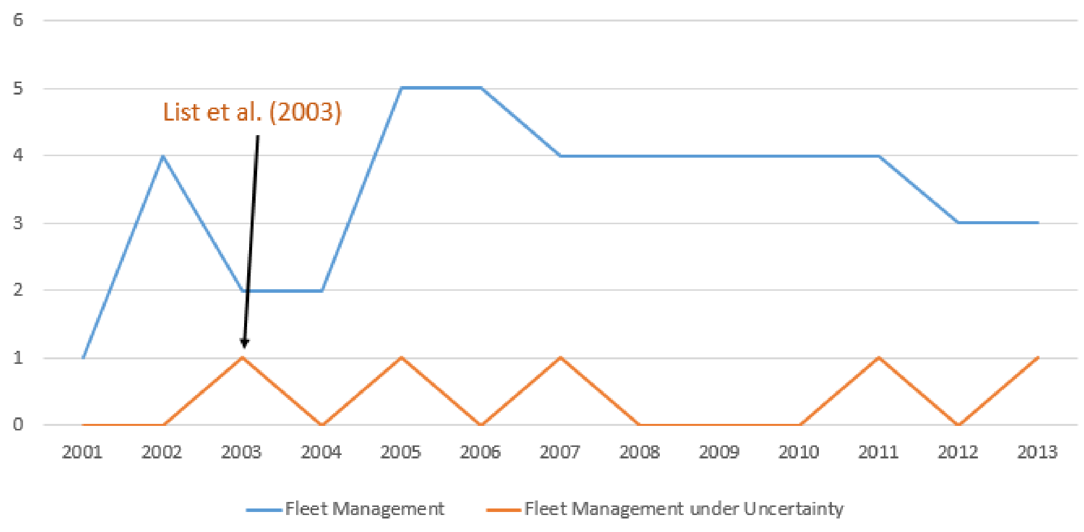


Figure 2.5: Number of Studies on Fleet Management in Recent Years

One of the articles in the relevant literature stands out: List et al. (2003) consider fleet sizing problem under demand uncertainty. They considered the tradeoff between the cost of expanding the vehicle fleet and the additional transportation cost with a smaller fleet. They use a two-stage stochastic model and have taken 60% of the citations (51 out of 84) in the relevant literature.

Vehicle Routing Problem

Literature on the vehicle routing problem is much larger than the problems we have analyzed up to now. Since the nature of the VRP is closer to the operational level decisions, there have been many extensions to the basic VRP throughout the years. Starting decades ago with the simpler Traveling Salesman Problem, the literature expanded into the multi-vehicle problems and recently into the multi-depot multi-vehicle problems. Of course, the complexity of the models increases as they are considering many decisions simultaneously. This requires more effective approaches and new solution methods to be developed. There are also some extensions originating from urban applications, such as hard/soft time windows and eco-friendly vehicles for delivery.

Figure 2.6 summarizes the literature on the VRP and the VRP under uncertainty. For the uncertainty aspect of our search, we looked for the keywords “uncertainty”, “uncertain”, “robust”, “stochastic”, and “reliability”. We added reliability to the set of keywords; because, unlike the previous problem types, reliability plays a greater role in the VRP under uncertainty, especially in studies considering the VRP networks in a graph-theoretical framework. The research area is specified as “Operations Research Management Science”.

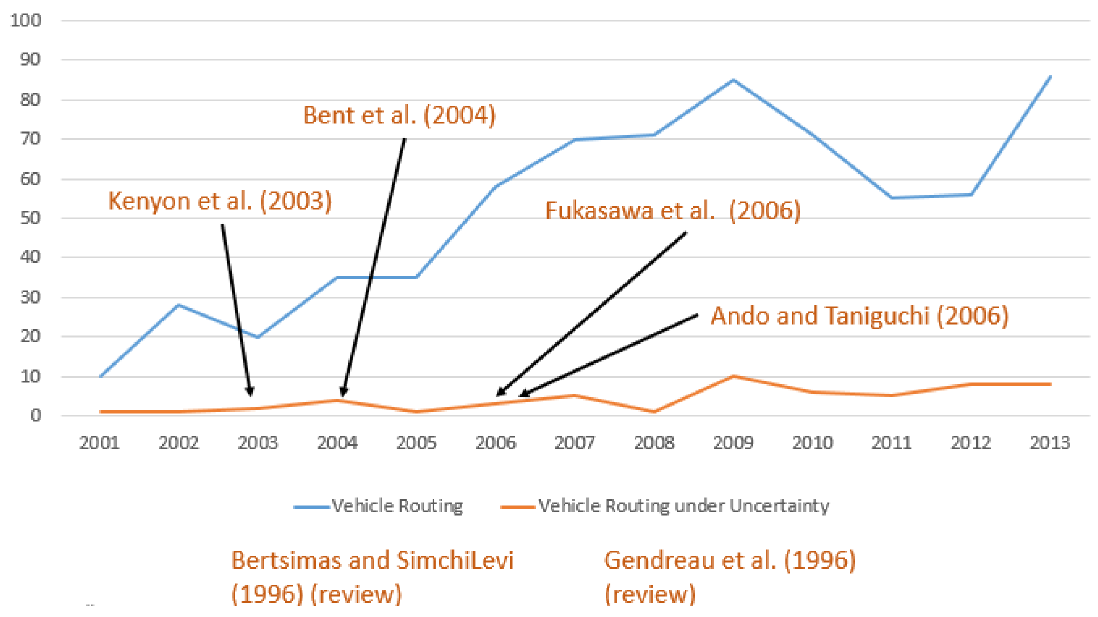


Figure 2.6: Number of Studies on Vehicle Routing in Recent Years

As we mentioned earlier, the VRP under uncertainty has been studied more extensively than the other problems. Naturally, there are a few articles to be specified as important contributions. We selected six of them as particularly relevant. For the relevant literature before 2000, we refer the reader to two review papers, Bertsimas and SimchiLevi (1996) and Gendreau et al. (1996). The former deals with robust algorithms, while the latter is on the stochastic VRP.

Others are as follows: Kenyon et al. (2003) consider random travel times, Bent and Van Hentenryck (2004) make scenario-based plans considering stochastic customers, Fukasawa et al. (2006) suggest a robust branch-and-cut-and-price algorithm for the capacitated VRP. Ando and Taniguchi (2006) look into travel time reliability in the VRP with time windows.

These four articles have got 33% of the citations (276 out of 836) of the literature on the VRP under uncertainty in the years after 2000.

Taş et al. (2014) is a very recent work that considers the VRP with soft time windows under travel time uncertainty. Their assumptions are close to real life systems and the method they present is efficient. However, the high complexity of their model prevents them from solving medium sized instances optimally.

There are also some other studies that consider multiple levels of city logistics decisions. Below, we provide an overview of articles that combine decisions in multiple levels.

Facility Location – Fleet Management

There is one study that considers the facility location and fleet management decisions under uncertainty. Fazel-Zarandi et al. (2013) consider travel time uncertainty subject to probability distributions, so that scenarios can be constructed to represent the uncertainty situation in the system. They use logic-based Benders' decomposition as their solution method. An important novelty in this study is the use of three-level decomposition in this kind of problem.

Fleet Management – Vehicle Routing

Generally, studies on vehicle routing problems try to find the optimal routes to be used by a fixed number of vehicles to serve a set of customers. However, in a smaller number of articles, the number of vehicles is not fixed. Such problems are a generalization of the classical VRP in the sense that a new tradeoff is introduced concerning the cost and benefit of additional vehicles. None of these articles consider uncertainty. Nevertheless, we would like to mention three selected articles as possible references on this field.

Gheysens et al. (1984) compare the performance of different heuristics, propose a new heuristic and provide computational results on the fleet size and mix VRP. Brandao (2009) uses tabu search algorithm to determine the fleet size and vehicle routes simultaneously. Liu et al. (2009) use a genetic algorithm for the fleet size and mix problem.

Facility Location – Vehicle Routing

Facility location and vehicle routing decisions are studied together more frequently due to their practical relationship. The literature on location-routing problems has been expanding for decades. There are also other problems that are structurally similar to location-routing, such as inventory-routing and production-routing. For example, the selection of a facility location in location-routing problems is modeled in a similar way to the selection of periods with positive inventory in inventory-routing problems, and to the selection of periods with production in production-routing problems. This structural similarity allows the modification of algorithms developed for one problem to be used for another.

All these problems deal with a first level decision of locating facilities, stockpiling or scheduling production and a second level decision of routing. The two-level decision making in these problems has led to the use of two-stage stochastic models, where the first stage decisions are made under uncertainty and second stage decisions are made after the parameters under uncertainty have been observed.

We selected three important articles considering the location-routing problem under uncertainty. Laporte et al. (1989) and Berman et al. (1995) develop methods to deal with both decisions simultaneously. Albareda-Sambola et al. (2007) suggest a heuristic algorithm to find the lower bound to the stochastic location-routing problem. Note that this article is still represents state-of-the-art in this area, as even finding a lower bound for the stochastic location-routing problem is hard.

As for the inventory-routing problem, we selected four articles. While Berman and Larson (2001) and Kleywegt et al. (2002) employ stochastic programming as their solution approach, Solyalı et al. (2012) employ robust optimization approach. Coelho et al. (2014) provide an extensive review of the inventory-routing studies in the last three decades.

Among the studies considering production-routing problem, we selected two major ones. Adulyasak et al. (2015a) use Benders' decomposition to solve the production-routing problem under demand uncertainty in a stochastic setting. Adulyasak et al. (2015b) provide a review of production-routing algorithms.

Now, we present a more detailed literature review on the freight distribution network design problem under uncertainty and topics related to the problem. Here we intend to determine the relevant modeling and solution methods in the literature to provide a basis for our study on network design under uncertainty.

Since the main characteristic of our work will be uncertainty, we first review the literature on optimization under uncertainty. Then, we will focus on the causes of uncertainty that have an impact on freight distribution network design. Lastly, we will look at the approaches compatible with our problem.

2.2 Defining Uncertainty in the City Logistics Context

Rosenhead (1972) divides decision making into three categories in terms of nature of information: certainty, risk and uncertainty. In certainty situations, deterministic models can be effectively used. In risk situations, parameters regarding available information are not exactly known, but they are governed by probability distributions which can be used to assign probabilities to different levels of a parameter. Lastly, in uncertainty situations, probability distributions are not available either. In this chapter, we will deal with risk and uncertainty situations. While it may not be possible to obtain probability distributions of disruptions in a large freight distribution system, we may come up with scenarios and assign probabilities to them. For the sake of simplicity, we will refer to risk and uncertainty concepts together as uncertainty.

Taniguchi et al. (2010) provide a review of natural and manmade hazards that have an impact on city logistics (see Figure 2.7). They classify the hazards in two axes: frequency and complexity/uncertainty. They also classify the studies made in this area with respect to the approaches used, such as robustness, stochastic programming, simulation, and multi-objective optimization. The causes of uncertainty mentioned in this article are mostly related to the design decisions on operational level. Since we will be mainly considering design decisions on strategic and tactical levels, only the approaches developed for these levels have been taken into account in this article. Particularly, we will be interested in robust optimization and stochastic programming approaches due to their ability to deal with uncertainty.

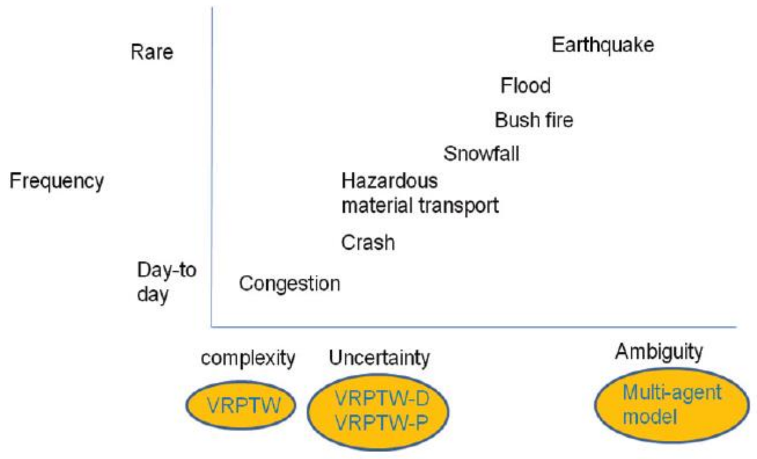


Figure 2.7: Taniguchi's Classification of Reasons for Uncertainty

We investigate the decision making in freight distribution network design in three levels, operational, tactical and strategic. Each level of decision making is affected by uncertainty in a different way. For example, on operational level, an urban transportation vehicle driver may choose to use an alternative road after learning there has been an accident on the road previously planned. On tactical level, allocation of customers to CDCs may be updated according to the changes on the road network connecting the supply and demand nodes of the system. On strategic level, CDC locations may be changed or additional CDCs may be opened according to the changes in demand from different regions of the city. Decision making on all levels are highly dependent on the information obtained from the real life system, and the information is usually just an estimate which is subject to change with time.

Since we want to determine the uncertainty factor most effective and the solution approach most suitable in a CL setting, we start by presenting an overview of the causes of uncertainty and modeling approaches used in the relevant literature. Then we present the reviewed studies categorized accordingly.

2.3 Causes of Uncertainty

In the literature, the main causes of uncertainty and risk in a freight distribution network design problem are specified as follows:

- Demand amount
- Travel time / Transportation cost
- Service time
- Throughput cost
- Unit cost
- Rare events (Manmade and natural disasters) such as link failures and facility disruptions

Demand amount is the uncertainty factor most frequently considered in the literature. This may be due to the fact that capacity constraints have traditionally been included in the widely studied problems, especially in the facility location literature. Thus, in order to ensure feasibility over capacity constraints or to reduce the cost of changes to accommodate the uncertain demand, researchers used robust/stochastic programming to deal with demand uncertainty. In the literature for facility location under uncertainty, vast majority of the studies deal with demand uncertainty.

When the fleet management and vehicle routing problems are considered, however, we see that demand uncertainty is not the single dominant cause of uncertainty in the literature. Half of the few studies dealing with fleet management under uncertainty consider transportation cost uncertainty. If we look at VRP studies, we see that there is roughly one study considering travel time/service time uncertainty for every three studies considering demand amount uncertainty.

Until recent years, most studies focused on the strategic decisions of the logistics problems and aimed at making long-term investment decisions such as facility location and capacity planning in an aggregated way. As may be expected, demand uncertainty has been the most critical factor for such decisions by a large margin. However, as more advanced methods are being developed for more complex problems, tactical decisions are coming into consideration along with strategic ones. Uncertainty factors other than demand amount have been increasingly considered through time, as they are critical for the tactical decisions. So, one may expect that the current state of the literature will change and uncertainty factors other than demand amount will gain popularity.

2.4 Approaches for Modeling Uncertainty

Although several other approaches can be found in the stochastic programming and the robust optimization literature, we consider only the ones applicable in our case for incorporating the causes of uncertainty:

- Stochastic programming
    - Mean-outcome models
- Robust optimization
    - Minimax regret / Worst case optimization
    - p-robustness
    - bw-robustness

The main difference between stochastic programming and robust optimization can be specified as follows. Stochastic programming takes into account all the anticipated uncertainty realizations weighted with their corresponding probabilities, however robust optimization tries to find the solutions that would not fail to produce a targeted objective value no matter how the uncertainty is realized. Both approaches have their advantages and disadvantages.

Since stochastic programming optimizes an objective function that involves the expected outcome of the decisions, the probability attributed to scenarios or the probability functions that fit to the uncertain parameters have a large impact on the behavior of the model. For example, stochastic programming models may select solutions that would obtain good performance with 99% probability, but very bad performance with 1% probability. However, the performance that would occur with 1% probability could be unacceptable to the decision maker. Therefore, the main criticism against stochastic programming approach is that it fails to take into account the worst case scenarios and leaves the system vulnerable to such occurrences.

On the other hand, in the robust optimization approach, where mostly the worst case scenarios are considered, the performance of the solution may be unsatisfactory under the steady conditions. Since the decisions are made according to the worst case, performance under normal conditions are often much worse than optimum. Therefore, robust optimization is criticized for over-conservatism, i.e. over-emphasizing the risk in the worst case scenarios. p-Robustness and bw-robustness are methods that try to deal with this problem up to a certain extent, but they also perform unsatisfactorily when the worst case performance is significantly different than the average occurrence.

2.5 Review Articles

Gabrel et al. (2014a) provide a review on robust optimization and robustness. The robustness problems are studied in two groups: static robust optimization where all decisions must be made under uncertainty and multi-stage decision making where some recourse actions are possible after uncertain parameters are observed.

Gabrel et al. (2014a) state that, although a large portion of the static robust optimization literature focuses on applying worst case optimization over a convex uncertainty set, such approaches do not produce satisfactory results under normal conditions. For example, determining the optimal solution according to a worst-case scenario with probability 0.01% may result in operating the system with a solution far from optimal 99.99% of the time. When the performance at the worst-case is not so critical, this approach does not produce applicable results. This characteristic of worst case optimization creates the need for new approaches that take into account the performance of decisions under normal conditions as well as worst case scenarios. Examples of such approaches are p-robustness which obtains solutions to guarantee that the objective function value will always be within c% difference of the overall optimal solution and bw-robustness which provides solutions that guarantee an objective value w and maximizes the probability of achieving an objective value b. Studies using these approaches will be presented later on.

The multi-stage decision making problems are generally studied as two-stage stochastic models. In the first stage, selected decisions are made under uncertainty. These are usually investment decisions with long term effects. In the second stage, recourse decisions can be made after the uncertainty has been resolved. The recourse decisions such as vehicle routing, generally have shorter term effects. Studies using these approaches mostly consider the facility location decisions in the first-stage and vehicle routing decisions in the second stage. We will present several studies using this approach later on.

Snyder (2006) makes a review of articles dealing with the facility location problem in the presence of uncertainty. Articles are divided into two categories: stochastic location and robust location problems. For the stochastic location problems, mean-outcome, mean variance and probabilistic models are discussed. Stochastic models assign probabilities to the elements of their scenario set. Mean-outcome models try to optimize the expected value of the objective function. Mean-variance models consider both the mean-outcome and mean-variance of the objective value, so that the risk aversion of the decision maker can be reflected on the model. Probabilistic models try to maximize the probability that the solution will achieve good performance, where the definition of good performance is selected by the decision maker.

For the robust location problems, minimax cost / minimax regret models are dominant, accompanied with sensitivity analysis and p-robustness. Minimax cost and minimax regret models are the worst case optimization approaches already discussed above. P-robustness is an approach aiming to produce solutions that will always achieve objective value at most within c% difference of the overall optimal.

Bertsimas et al. (2011) offer an extensive review of robust optimization problems, regardless of application domain. They focus on computational attractiveness and modeling power aspects rather than making the conceptual classification of the related articles. Bertsimas et al. (2011) also distinguish between single-stage and multi-stage decision making problems and give a list of application areas of these solution methods such as portfolio optimization and supply chain management.

2.6 Stochastic Programming Approaches

2.6.1 Demand Uncertainty

In this section, we review the articles dealing with demand uncertainty with the help of stochastic programming. Contrary to the definition by Rosenhead (1972), the word uncertainty refers both to the risk and uncertainty in the stochastic programming literature. If it is not possible to represent the distribution of parameters with probability distributions, scenarios are constructed and they are assigned probabilities.

Chan et al. (2001) minimize the expected cost of a stochastic location routing problem with demand uncertainty. Probability of demand that occurs at the nodes is estimated using a queueing process. They solve the problem with the help of stochastic decomposition and space-filling curves.

Daskin et al. (2002) study the facility location problem with demand uncertainty. They obtain a nonlinear program to minimize the expected cost of inventory in distribution centers. The model is solved using Lagrangian relaxation.

Crainic et al. (2012a) propose a detailed two-tiered capacitated location problem minimizing the expected cost of operating the system under demand uncertainty. They follow a two-stage stochastic programming approach with recourse actions. The first stage is for having the nominal plan and the second stage modifies some parts of the plan to adapt to the realization of demand. They consider demand around a point estimation. While they propose a modeling framework, they do not propose a solution method.

Listes and Dekker (2005) apply the stochastic programming approach on the facility location problem in a reverse logistics network. They maximize the expected profit of the system under uncertainty in demand amounts and locations.

Tsiakis et al. (2001) develop a mixed integer programming model that determines the number, location and capacity of distribution centers which minimizes the annual expected cost of the entire network. The model operates under demand uncertainty but differs from the above studies with its capability of handling multi-echelon networks.

Gülpınar et al. (2013) consider the stochastic facility location problem with single commodity and multiple capacitated facilities under demand uncertainty. They minimize the expected cost of the system and claim that their model obtains better results than the models not considering robustness both in terms of worst case and average total cost.

Adulyasak et al. (2015a) deal with the production routing problem under demand uncertainty. They propose a two-stage stochastic production routing problem formulation and use Benders Decomposition as a solution method to solve the formulation. They make several computational enhancements on the classical Benders Decomposition algorithm. They propose using Sample Average Approximation for handling the possible large number of scenarios.

2.6.2 Throughput Cost / Unit Cost Uncertainty

Throughput cost is the variable cost of processing a unit in CDCs. Unit cost is the variable cost of producing and/or shipping one unit of product. These cost parameters are also subject to uncertainty and change through time.

Ricciardi et al. (2002) minimize the expected transportation cost on a network where the throughput costs in distribution centers are random. They solve a nonlinear problem using heuristics. Baldi et al. (2012) study the same problem and apply asymptotic approximation method. They also analyze the effect of using different probability distributions on the solution.

Yu and Li (2000) minimize the expected cost of operations considering a scenario set under uncertainty of unit costs. They transform the robust model into a linear program by adding new variables and demonstrate the computational efficiency of their model on two examples. In the first example, they formulate the production, inventory and transportation plan of a wine company. In the second example, they provide a model scheduling aircraft routes of an airline company.

2.6.3 Travel Time / Service Time Uncertainty

In a city logistics network, shipments are often subject to time constraints. In other words, transportation vehicles are often required to reach their destinations within specified time windows. Therefore, travel time from CDCs to demand points and service time within CDCs are important factors of uncertainty in freight distribution networks. Nevertheless, few studies incorporate these factors into freight distribution network formulations.

Ando and Taniguchi (2006) fit normal distribution to travel time, according to data obtained from real freight operations. They model a probabilistic vehicle routing and scheduling problem with time windows and solve it using genetic algorithms. They also provide a case study that includes a pick-up delivery truck and 11 customers. The problem is formulated as a vehicle routing problem with time-windows.

Binart et al. (2015) propose a two-stage formulation for the multi-depot vehicle routing problem with time windows under stochastic travel and service times. They suggest dividing customers into two groups: the first group consists of mandatory customers and the second group consists of optional customers. They set up the "skeleton" of the network regarding only the mandatory customers and serve optional customers if profitable. The suggestion of grouping customers as mandatory and optional is the defining characteristic of this study. Binart et al. (2015) model the stochastic travel and service times with discrete triangular distributions and minimize the total travel time while visiting as many optional customers as possible.

2.6.4 Other Cases

Santoso et al. (2005) propose using the sample average approximation and accelerated Benders decomposition for handling supply chain network design under uncertainty. The problem of interest includes a set of supply facilities, processing facilities, and customers, and a set of arcs connecting them. Santoso et al. (2005) minimize the total annual cost of operating the system. They claim that their method can deal with the problems of a realistic scale and potentially infinite number of scenarios. The problem is formulated in a two-stage setting. The stochastic model is capable of dealing with all kinds of uncertainty as long as the joint distribution of scenarios involving uncertain parameters are known.

2.7 Robust Optimization Approaches

2.7.1 Minimax Cost / Regret

Problems based on regret are often more difficult than the stochastic problems due to their minimax structure (Snyder, 2006). Still, minimax regret is a widely studied problem in the literature. Their solution methods yield good results when the worst case scenario occurs; however, their results may not achieve satisfactory performance under normal conditions. Since the objective function does not reflect the solution performance under normal conditions, the importance of the performance in worst case is over-emphasized. As a result, the performance under normal conditions is sacrificed for slightly better results in the worst case (over-conservatism).

Demand Uncertainty

Averbakh and Berman (2000) study 1-median problem on a transportation network where node weights (possibly demand) are uncertain. They use weight scenarios to investigate relative regrets at each case.

Ukkusuri et al. (2007) deal with the network design under demand uncertainty and solve their robust model using genetic algorithms. In this paper, the focus is on a traffic network design problem, rather than specifically considering the freight distribution aspect.

Alumur et al. (2012) consider facility setup costs and customer demands to be uncertain and cannot be expressed in a probability distribution. They minimize the maximum regret under all scenarios to determine hub locations in a two-stage strategic setting. In the first stage, hub locations and allocations are determined and in the second stage, routing decisions are made.

Atamtürk and Zhang (2007) construct a two-stage robust optimization model for the multicommodity network design problem under demand uncertainty. They apply the method to the lot-sizing and location transportation problems.

Gabrel et al. (2014b) employ a two-stage robust optimization setting for the location transportation problems under demand uncertainty. They use cutting plane algorithm to solve the problem and propose a tight bound for solving large instances.

Gounaris et al. (2013) study the capacitated vehicle routing problem under demand uncertainty. They try to find the minimum cost solution that will be feasible under all demand realizations by using minimax cost approach.

Remli (2011) studies a two-stage robust location transportation problem focusing on the worst case scenario. In this study, demand is considered uncertain and the proposed model is solved through linear relaxation of the mixed integer formulation of the problem.

Transportation Cost / Travel Time Uncertainty

Yin and Lou (2009) assume demands belong to an uncertainty set, but they don't follow a probability distribution. They use minimax approach to deal with transportation costs.

Han et al. (2013) study the vehicle routing problem under travel time uncertainty. They suggest using range estimates rather than point estimates for the travel times and plan according to the worst case scenarios in these ranges. This way, Han et al. (2013) claim that more meaningful estimations can be made about the uncertain parameters, leading to the solutions to be more applicable.

Demand and Transportation Cost Uncertainty

Mudchanatongsuk et al. (2008) consider both transportation cost and demand uncertainty. They represent uncertainty with independent closed convex sets defined as deviations from a nominal value. They try to minimize the cost for the worst-case and propose a column generation method for the solution of their problem.

Other Factors

Inuiguchi and Sakawa (1995), Mausser and Laguna (1999), and Averbakh (2000) study the minimax regret approach for the linear programming problems with uncertainty on the objective function coefficients. Inuiguchi and Sakawa (1995) use decision theory to determine the final solution so that a reference solution set is used to apply minimax criterion on the candidate solutions. Mausser and Laguna (1999) use mixed integer programming formulation to deal with the minimax relative regret problem where the objective function coefficients are known to take values in an interval. Averbakh (2000) considers the combinatorial optimization problems and illustrates the proposed approach on minimax multifacility location problems under uncertainty of objective function coefficients.

2.7.2 p-Robustness

As mentioned earlier, minimax approach only considers the worst case scenarios, thus producing solutions that will perform badly under normal conditions. To remedy this problem, which is called over-conservatism in the literature, p-robustness measure is developed. p-Robustness guarantees that the solution found by the model will obtain satisfactory results under all scenarios. The objective function will take values between (1-c%) and (1+c%) of the actual optimal objective value in all cases. This way, the solutions performing well under normal conditions are not sacrificed for the worst case, but a solution is found taking into account all scenarios at the same time.

Liu et al. (2010) suggest p-robustness, where p is the upper bound of relative regrets in disruption scenarios. They solve deterministic optimization models for a given set of scenarios and constrain the relative regret of each scenario to be smaller than a percentage of the optimal objective value.

Snyder and Daskin (2006) suggest p-robustness for the facility location models under transportation cost uncertainty. They use Lagrangian relaxation as a solution approach.

Peng et al. (2011) use genetic algorithms to minimize the nominal cost of operations while considering p-robustness under facility disruptions. They refer to Bundschuh et al. (2006) while stressing the difference between reliability and robustness. Bundschuh et al. (2006) define robustness as the ability of a system to perform its intended function relatively well under disruptions and reliability as the probability of a system to perform its function within a given time horizon and environment. In the light of these definitions, Peng et al. (2011) incorporate both reliability and robustness in their model that minimizes costs under possible facility disruptions.

Gutierrez et al. (1996) propose adaptations of the Benders decomposition for finding robust network designs. They search for p-robust solutions to the uncapacitated network design problem and suggest the Multi-Masters Benders algorithm, considering all scenarios to simultaneously generate a number of robust designs.

2.7.3 bw-Robustness

Another robustness measure is bw-robustness which is suggested by Roy (2010). In this measure, determining two values, w and b, is proposed. w is the least acceptable objective value and b is an objective value that is strictly preferred to w. In a bw-robustness model, the objective is to guarantee the objective value w for all scenarios and to maximize the probability of achieving objective value b. Again, bw-robustness measure overcomes the over-conservatism problem and offers the ability to control two parameters b and w according to the preferences of the decision maker.

2.7.4 Other Cases

Koulakezian et al. (2012) consider the general traffic assignment problem under uncertainty. They consider weather conditions and traffic accidents as major reasons for a decrease in traffic supply and add that traffic demand is uncertain. As a result, traffic congestion may occur and travel times may increase. They define network criticality as a graph metric measuring the robustness of a network and try to minimize total travel time using this concept.

Wang et al. (2006) suggest a reliability perspective for the robust transportation network design problem. They provide a detailed optimization model, considering the reliability of a network. They compare the telecommunication networks with the logistics networks and try to apply the telecommunication network reliability concept to logistics systems. They incorporate the spatial distributions of the nodes in a system using density functions.

Crainic et al. (2012b) suggest an approach for dealing with multi-scenario problems. They group similar or dissimilar scenarios using k-means clustering. They also introduce the concept of scenario covers, where each scenario appears in two groups. They conclude that by solving multi-scenario subproblems where scenario covers are used, the number of iterations can be reduced and solution quality can be increased.

2.8 Conclusions on the Literature Review and Framing Our Work

2.8.1 Factors of Uncertainty

Considering the strategic point of view in freight distribution network design under uncertainty, we have the following observations and preferences that will be incorporated into our work.

The demand uncertainty aspect of the problem is usually handled by the aggregate demand forecasts. These are long-term estimations that are expected to be sufficiently accurate for short to medium term. Customers of the system are responsible for reporting their demand correctly to the CL administration, but any changes may be accommodated by the system with some additional cost/fee incurred to the customer. Since we do not consider the relations between customers and CL administration in this study, we can ignore the daily fluctuations in demand. All in all, we will not be considering demand uncertainty.

On the other hand, planning for the changes in transportation cost falls into the responsibility of the CL administration. Moreover, transportation cost uncertainty is a main issue in an urban area with frequent traffic congestion. In a stochastic/robust optimization model dealing with transportation cost uncertainty, it would be possible to meaningfully divide our decisions into two stages. In the first stage the facility location decisions are made under uncertainty and in the second stage, allocation of customers can be made after uncertainty on transportation cost has been realized; as would be the case in a real life CL system. Without making any changes to the formulation of location-allocation models, we can model the transportation cost parameter on travel time, since transportation cost is expected to be proportional to travel time. Also, there are several studies in the literature that model travel time in a city.

Uncertainty on service time is not very different from uncertainty on travel time as the time spent on travel and on service can be aggregated into a single parameter. This parameter is to be constrained by the time-window effective on the city-wide freight transportation. Since we will not be focusing on the operational level decisions, routing and time windows are not represented in our models. Thus, service time uncertainty will not be considered.

Uncertainty of throughput cost and unit cost will not be considered in our study. These cost items are not affected by the decisions on facility locations, customer allocations or vehicle routes.

Link failures and facility disruptions could be considered in the context of disasters (earthquakes, floods etc.), but these are rare events which are outside of the focus of our work. Such a system will be certainly affected in case of a natural disaster and cost of maintaining service will be much higher. However, planning against such rare events at the expense of performance under normal conditions would not be reasonable. Still, small changes in road capacities due to accidents, congestion, etc., which can be considered as link failures, may be represented as causes of the travel time/transportation cost uncertainty.

2.8.2 Solution Approach

In the light of these discussions, we choose to employ the stochastic programming approach. We believe that the over-conservatism problem in robust optimization is difficult to handle, while planning against the worst case could be possible also in stochastic programming by assigning larger probability values to the worst case scenarios. This way, the weight of worst case situations would be larger in the objective function, so that the proposed solutions perform satisfactorily in all scenarios. The amount of increase in the probabilities of worst cases may be determined according to the preferences of the decision makers.

More importantly, robust optimization does not represent a real life system as well as stochastic programming. In the real life CL system, the goal is to find solutions that provide the best long-term performance; bad performance in only one scenario should not prevent the selection of a good solution on average. Robust optimization focuses on the goodness of our results in the worst-cases at the expense of achieving a good long-term solution performance; however, stochastic programming focuses on the average performance over the long-term. For these reasons, we believe that stochastic programming is more suitable approach for CL systems.

2.8.3 Benders Decomposition Method

Regarding the studies reviewed above, a two-stage formulation appears to be suitable for the location-allocation part of our problem. In two-stage stochastic formulations decisions are divided into two groups: first stage decisions and second stage decisions. Factors under uncertainty are observed between the first stage and the second stage.

It seems natural to approach the location-allocation problem in this manner: facility location decisions (which are investment decisions) are to be made under uncertainty and customer allocation decisions (which can be changed in time) are to be made after the actual parameter values are observed. The second stage decisions are also called recourse decisions. In most studies, first stage decisions are represented with binary variables and recourse decisions are represented with continuous variables. This is the case in our problem.

The computational complexity inherent in the nature of this problem forces us to use decomposition methods. Among these methods, Benders decomposition has been extensively applied to two-stage stochastic models. This method is very suitable for two-stage stochastic models, because it decomposes binary and continuous variables, an approach that naturally matches the variable types in a two-stage stochastic setting.

Unfortunately, the Benders Decomposition approach does not give satisfactory results by itself. There have been many computational enhancements proposed for improving Benders Decomposition, especially in a stochastic setting. These enhancements include L-shaped method, addition of multiple cuts at each iteration, ϵ-optimal solution, cross-decomposition, local-branching, lower bound lifting inequalities, scenario group cuts and pareto-optimal cuts. We work on some of these enhancements that we find applicable for our problem.

2.9 Comparison of Journals

Lastly, we provide the journals where the articles selected in our analyses were published. We believe these journals are the ones that did the most important contribution to the City Logistics literature. The following table lists the most-cited journals in decreasing order of the number of references we used (excluding journals with fewer citations than 2):

Table 2.1: Comparison of Journals

Journal	**Number of Citations**
European Journal of Operational Research	17
Transportation Science	9
Operations Research	8
Annals of Operations Research	6
Computers & Operations Research	6
Transportation Research Part B	5
Mathematical Programming	4
Operations Research Letters	4
Transportation Research Part E	4
IIE Transactions	3
Journal of the Operational Research Society	3
Management Science	3
INFORMS Journal on Computing	2

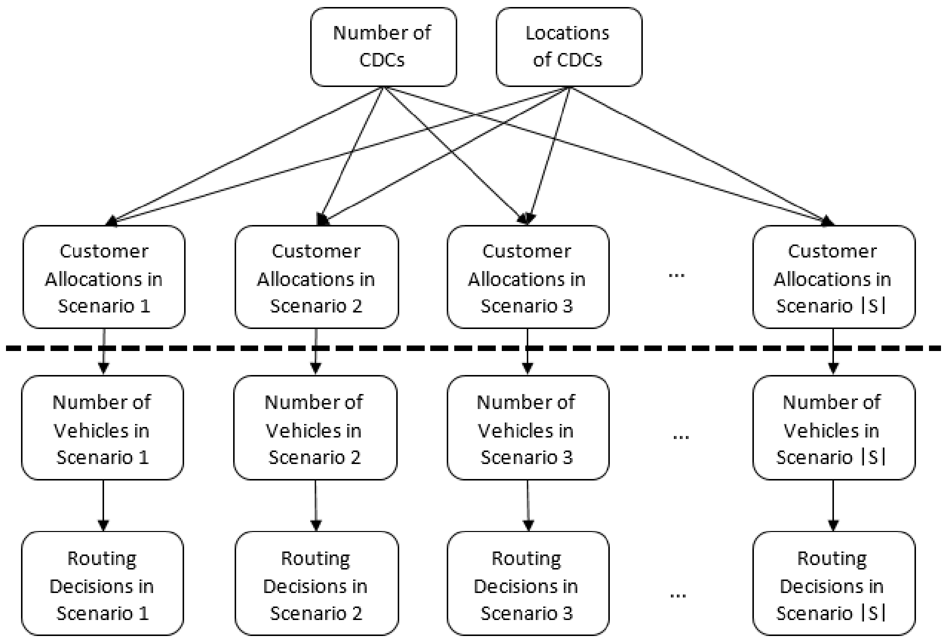
CHAPTER 3

PROBLEM FORMULATION

The problem we are interested in this thesis is the City Logistics Network Design Problem under Transportation Cost Uncertainty. The following major decisions are to be made in such a network: number and locations of CDCs, allocation of customers to the CDCs, and determining fleet size and routing of vehicles.

To the best of our knowledge, there is no study in the literature that considers all these decisions together under uncertainty and manages to find optimal solutions for instances with a realistic size. Our preliminary experiments showed that finding an optimal solution to a realistic size instance would not be possible in reasonable time, even with simplifying assumptions.

Therefore, we divide the problem according to the decision levels we specified. We consider the strategic level decisions on the number and locations of CDCs and the tactical level decisions on allocation of customers to CDCs. Since these decisions are more critical due to their long-term effect, attempting to optimize these decisions is promising the largest benefit for the system. Also, these decisions need to be made earlier in a real-life system and they directly define the solution space for the lower-level decisions. Figure 3.1 illustrates this point in a two-stage stochastic framework; decisions related to CDCs are only made once, while the other decisions are specifically made for each scenario.



Number of CDCs
Locations of CDCs
Customer Allocations in Scenario 1
Customer Allocations in Scenario 2
Customer Allocations in Scenario 3
...
Customer Allocations in Scenario |S|
Number of Vehicles in Scenario 1
Number of Vehicles in Scenario 2
Number of Vehicles in Scenario 3
...
Number of Vehicles in Scenario |S|
Routing Decisions in Scenario 1
Routing Decisions in Scenario 2
Routing Decisions in Scenario 3
...
Routing Decisions in Scenario |S|

Figure 3.1: Decision-Making Process in City Logistics

The logical precedence between these decisions makes it possible to decompose them in a meaningful way and without losing too much information by refusing to optimize the whole system. Once the higher-level decisions are made, it is possible and may be realistic to make the remaining decisions later when the actual parameter values can be observed. When the decisions are divided into two in this way, one major tradeoff arises at each stage:

- 1st stage:
    - CDC operating costs vs.
    - Costs from the 2nd stage model
- 2nd stage:
    - Vehicle operating costs vs.
    - Routing costs of delivery to customers vs.
    - Penalty costs due to the violation of time windows

These two stages are connected through the allocation decisions (assignment of customers to CDCs). Since even the 1st stage decisions take too much time for instances of realistic size, we do not attempt a solution method that iterates between the two stages to reach an optimal solution for the whole system. Rather, we acknowledge that these stages follow a natural sequence. The first stage decisions are preemptive over the whole solution space. Based on these observations, we prefer a sequential solution procedure.

Each of the lower level decisions will be effective in only a single scenario and on a local level within the system. For this reason, we believe that dividing the decisions in this way provides a suitable way of reducing solution complexity without sacrificing solution quality significantly.

Of course, there are more than one way of formulating the location-allocation model under uncertainty, depending on the nature of the system and preferences of the decision maker. The simplest one is the classical location-allocation problem which does not take into account the uncertainty factors. A second way of modeling is the use of scenarios and a stochastic programming approach. When the uncertainty on parameters is represented with the help of scenarios, the objective function becomes the weighted average of costs incurred by the decision through all scenarios; therefore, this formulation is called the expected value model.

We also consider a generalization of the expected value model, called the two-stage stochastic model. In a two-stage setting, first stage decisions (facility location) are made once and are not subject to change through time. However, second stage decisions (assignment of customers to CDCs) are made after the factors under uncertainty have been observed.

Now, we explain the different ways of formulating the strategic level decisions of city logistics network design.

3.1 Deterministic Location-Allocation Problem

In this network model, we use two decision variables:

- z_i determines whether the candidate CDC location i is selected or not
- x_{ij} determines the amount of demand served from CDC i to customer j

Parameters used in the model are the following:

- F_i, the fixed cost of opening CDC i

- C_i, the service capacity of CDC i
- T_{ij}, the travel time/transportation cost between location i and location j
- D_j, the demand amount from customer j

We use the abbreviation DLAP to denote the Deterministic Location-Allocation Problem.

DLAP Model:

minimize

$$\sum_i F_i * z_i + \sum_i \sum_j T_{ij} * x_{ij} \qquad (1)$$

subject to

$$\sum_i x_{ij} = D_j \quad \forall j \qquad (2)$$

$$\sum_j D_j * x_{ij} \leq C_i * z_i \quad \forall i \qquad (3)$$

$$x_{ij} \geq 0 \quad \forall i, j \qquad (4)$$

$$z_i \in \{0,1\} \quad \forall i \qquad (5)$$

The objective function (1) minimizes the total cost which is the sum of two cost items: fixed costs arising from opening and operating CDCs and transportation costs arising from the distance between customers and their assigned CDCs. Constraints (2) satisfy the demand of each customer. Constraints (3) enforce the limits on service capacities of each CDC. Constraints (4) and (5) are domain limitations.

Note that from now on, we will refer to T_{ij} as the distance and the cost of transportation between locations i and j, assuming we spend one unit money per unit distance. We will make the meaning of T_{ij} clear whenever necessary.

When the model is applied on a problem instance we generated, the results can be illustrated as in Figure 3.2. The legend to the bottom left of the figure shows the numbers of selected candidate facilities:

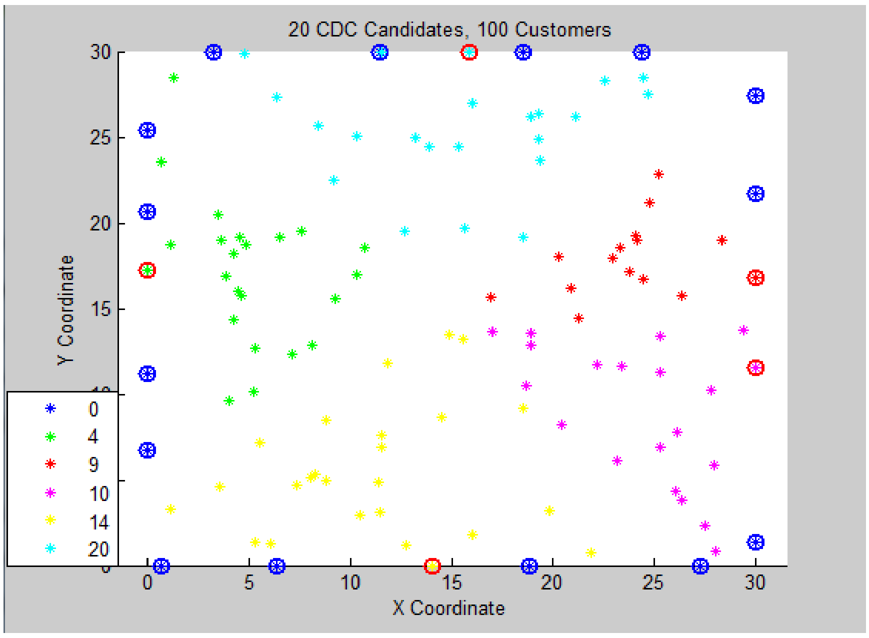


Figure 3.2: Deterministic Solution of a Typical Instance

Figure 3.2, the circled points on the city boundary are the candidate CDC locations. Blue circles indicate that the CDC is closed and red circles indicate that the CDC is open. Other points represent the customer locations. Largest part of the customers' demand is supplied by the CDCs whose color matches their colors. We make use of similar figures many times, to illustrate the changes in optimal solution when a different approach is taken.

We can modify the model to put an upper limit to the longest distance to be traveled between a customer and a CDC to reflect the service time performance. For this purpose, we define an additional continuous decision variable and a parameter.

- LD represents the longest distance to be travelled from a CDC to a customer
- RHS represents the upper limit on the allowed distance in the system

And add the constraints:

$$LD \geq T_{ij} * x_{ij} \ \forall \ i, j \qquad (6)$$

$$LD \ \leq RHS \qquad (7)$$

The first constraint set (6) forces LD to be equal to the largest of the distances between a customer and its assigned CDC. The second constraint set (7) forces LD to be smaller than a specified RHS value. The selection of the RHS value should be made carefully.

Since all candidate CDC locations are on the city boundary, small values would make the model infeasible.

An alternative to constraining the longest distance is penalizing its cost in the objective function. We introduce another cost item to represent penalty due to the largest distance between a customer and its assigned CDC.

We define:

- pt, the penalty due to unit distance in the longest travel distance between a customer and a CDC in the system

It is clear that estimating *pt* would not be trivial. It may be used to express a strong preference on LD, or to make an analysis on the tradeoff between LD and total transportation cost. We use the abbreviation DLAP-P to denote the Deterministic Location-Allocation Problem with Longest Distance Penalty.

DLAP-P Model:

minimize

$$\sum_i F_i * z_i + \sum_i \sum_j T_{ij} * x_{ij} + pt * LD \qquad (8)$$

subject to

$$\sum_i x_{ij} = D_j \quad \forall\, j \qquad (2)$$

$$\sum_j D_j * x_{ij} \leq C_i * z_i \quad \forall\, i \qquad (3)$$

$$LD \geq T_{ij} * x_{ij} \quad \forall\, i, j \qquad (6)$$

$$x_{ij} \geq 0 \quad \forall\, i, j \qquad (4)$$

$$z_i \in \{0,1\} \quad \forall\, i \qquad (5)$$

Note that we only modified the objective function. This time, the objective function (8) minimizes the total cost of function (1) together with the penalty incurring from the longest travel time in the system. In this case, as *pt* increase, its impact on the LD will increase the objective function value. As *pt* decreases, the system will converge to the first model (1)-(5).

3.2 Modeling Uncertainty

In this section, we explain how the transportation cost parameter is taken into account in our approach and consequently we develop the scenarios to solve the problem under uncertainty. While there are many studies on modeling travel time, transportation cost has never been modeled on its own to the best of our knowledge. Therefore, we focus on the factors affecting travel time.

The real-life causes of travel-time uncertainty are the following:

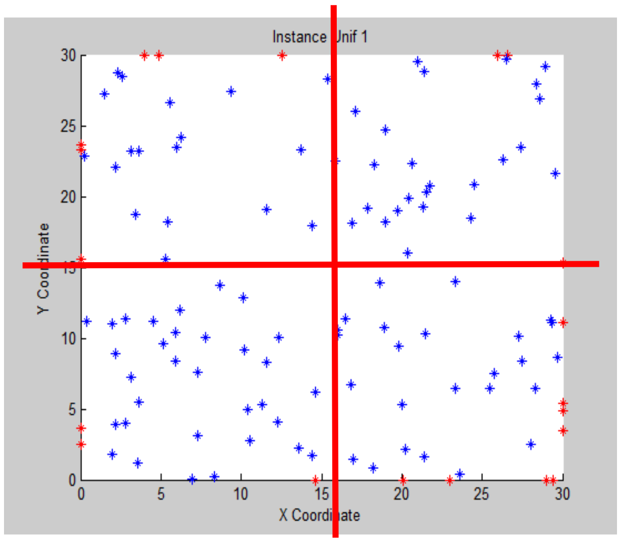
- Rush-hours
- Occasional traffic congestion
- Accidents
- Maintenance/construction

These causes may be divided into two groups: time-dependent traffic congestion and disruptions to traffic. Time-dependency can also be divided into two: fluctuations of congestion within a day and fluctuations through several days.

Rush hours, for example, are a kind of daily time-dependency. During the morning hours when commuters are going to work, there is significantly higher traffic congestion. Similarly, during the evening hours when commuters are coming back from work, traffic congestion is observed. Effect of time dependency can be observed also in other hours to a smaller extent.

Second, we observe significant difference in traffic congestion through the week. For instance, weekly delivery of freight is made mostly on Mondays and Fridays, while less congestion is expected to occur during the week-days between them. Naturally, people rarely encounter traffic congestion on weekends, except on long weekends and holidays.

Figure 3.3: Division of City Space



Disruption conditions include all other causes of traffic congestion. Accidents, construction projects, weather, social events etc. all have an impact on the travel time. There is, however, an important difference between disruption type and time-dependent causes. While time-dependent causes have city-wide effects, disruptions often affect a smaller area within the city. We will call this "local congestion" from now on.

For constructing meaningful scenarios, all these causes of traffic congestion must be taken into account properly.

To separately represent city-wide congestion and local congestion, we divide the cities into four regions: northwest, northeast, southeast, and southwest (see Figure 3.3). These can be considered as neighborhoods within a city. Vehicles traveling within a neighborhood use regular or narrow roads, while vehicles traveling between neighborhoods use highways or wider roads.

Deterministic travel time between two points in a city is proportional to the Euclidean distance between these points. But modeling the stochasticity of travel time is a different issue. In the literature, there are several alternatives for modeling travel times. Branston (1976) makes a review of the most frequently used travel time models.

In this study, we use the Bureau of Public Roads (BPR) function. This function is widely known and frequently used in the transportation literature. It does not require any distribution parameters to be estimated by the user. As we do not have real-life data from which such estimations can be made, there is a risk that the result of using statistical methods would be inaccurate. On the other hand, BPR function requires only the estimate of the proportion of used road capacity, which is much easier to estimate correctly.

Below, we give the Bureau of Public Roads function:

$$S_a(v_a) = t_a * \left(1 + 0.15 * \left(\frac{v_a}{c_a}\right)^4\right)$$

where

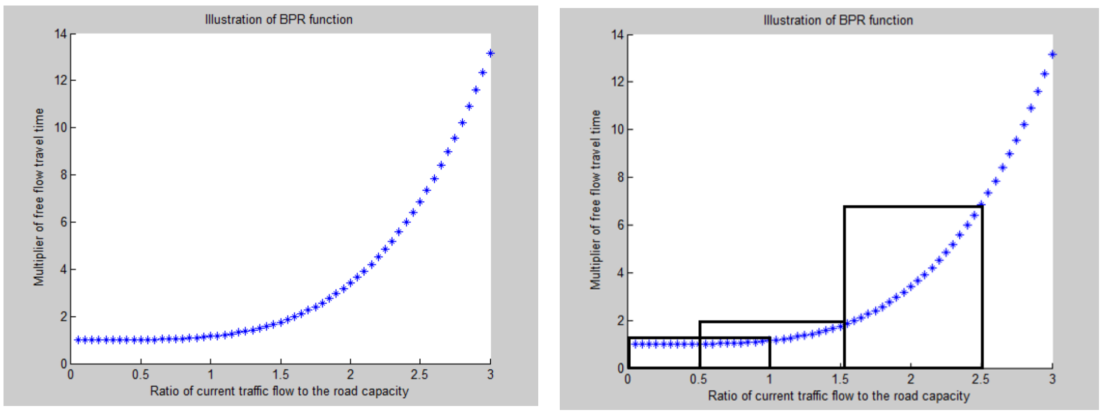
$S_a(v_a)$ is the average travel time on link

t_a is the free flow travel time on link

v_a is the volume of traffic on link per unit time

c_a is the capacity of link per unit time

In this function, the only parameter to be estimated is the ratio of current traffic flow to the road capacity. We illustrate the function's behaviour with respect to this ratio with the following plots in Figure 3.4:



(a) BPR Function (b) Selected Regions on the BPR Funct.

Figure 3.4: BPR Function and the Selected Regions

Observe that the travel time does not increase a lot until the traffic flow exceeds link capacity. This first part, between no usage and full capacity usage, provides a good estimation of traffic on weekends. However, on regular weekdays, the traffic flow is surely larger. Considering the roads will rarely be empty and flow sometimes exceed capacity, we choose the region between 50-150% capacity usage to represent the traffic on regular weekdays. Lastly, we select a region for the heavy traffic case. For most major cities, it would not be surprising to see that the flow/capacity ratio take very high values, especially during rush-hours on busy days. We select the region between 150-250% capacity usage to represent the heavy-traffic.

Since we are dealing with a freight distribution network design problem, we are mostly interested in the heavy-congestion days. Therefore, we generate a higher number of scenarios from the corresponding regions of the BPR function. The following list reflects the composition of the scenarios we generate:

- Low congestion days (20%)
- Average days (40%)
    - No local traffic (20%)
    - With local traffic (20%)
        - SE region congested (5%)
        - NE region congested (5%)
        - NW region congested (5%)
        - SW region congested (5%)
- Congested days (40%)
    - No local traffic (20%)
    - With local traffic (20%)
        - SE region congested (5%)
        - NE region congested (5%)
        - NW region congested (5%)
        - SW region congested (5%)

These scenario sets represent the changes in travel time due to time-dependency. The term “time-dependency” is used extensively in the literature, especially in the recent years, to describe the fluctuations of the travel times in a city. The effect of city traffic and the limitations of time windows made it necessary for the models to take into account the real-life conditions on the urban road network. The time-dependency concept essentially uses the same logic we did when introducing the rush hour congestion and expands it to include all hours of the day. The graph on Figure 3.5 taken from Van Woensel et al. (2008) illustrates the time-dependency of travel times.

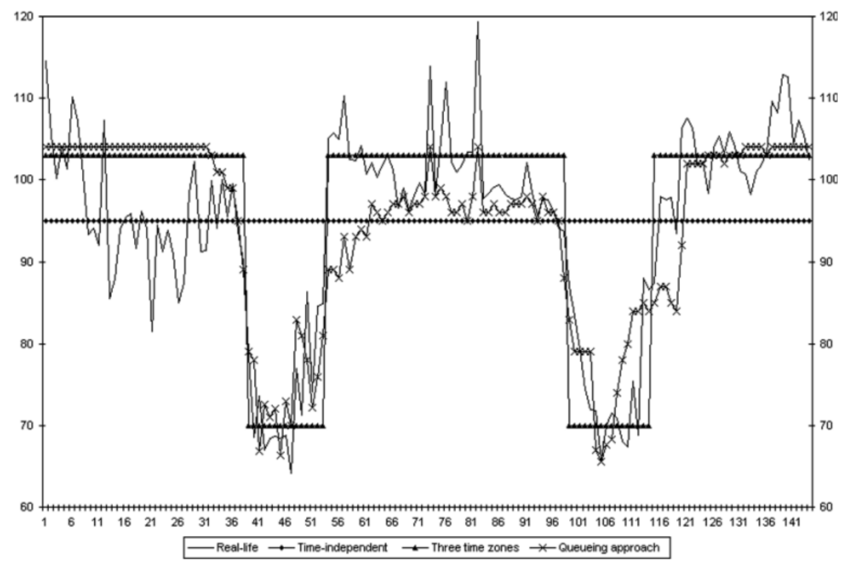


Figure 3.5: Different Urban Traffic Speed Patterns throughout the Day

In this graph, you see four different representations of average travel speed on the city roads. The first one shows real life observations. The second one shows the time-independent representation of travel speed. In this representation, travel speed is averaged through all hours of the day and it is reduced to a single travel speed to be used in the model for all hours. The third type is about three time zones. It is very similar to our rush-hour representation, since it decreases the travel speed significantly during the rush-hours. The last one is the queueing approach, which tries to achieve a representation as close to the real life observations as possible. Since using a queueing approach would cause further difficulties in our models, we chose to consider a three time zone representation when constructing scenarios.

Time-dependency can be useful in a CL setting, but only if the delivery windows are implemented on a city scale (if time windows are the same for all customers). In such a case, if the delivery windows are set earlier, the travel time values in the model would change significantly and the decisions would not be optimal anymore. We do not explicitly consider time-dependency. The scenarios may still be interpreted in a time-dependency context.

3.3 Single-Stage Stochastic Location-Allocation Problem

Now, we use these scenarios in a stochastic programming model. The single stage stochastic model optimizes the weighted average of costs that is incurred under different scenarios, where weights represent the probabilities of scenarios. We add a new parameter p_s to our basic location-allocation model and we change the objective function (1) only slightly.

- p_s: probability of scenario s

We use the abbreviation SLAP to denote the Single-Stage Stochastic Location-Allocation Problem.

SLAP Model:

minimize

$$\sum_i F_i * z_i + \sum_i \sum_j \sum_s T_{ijs} * x_{ij} * p_s \qquad (9)$$

subject to

$$\sum_i x_{ij} = D_j \quad \forall j \qquad (2)$$

$$\sum_j D_j * x_{ij} \leq C_i * z_i \quad \forall i \qquad (3)$$

$$x_{ij} \geq 0 \ \ \forall i, j \qquad (4)$$

$$z_i \in \{0,1\} \ \ \forall i \qquad (5)$$

The objective function (9) minimizes the total of fixed CDC opening costs and the weighted average of transportation costs across scenarios. T_{ijs} is the modified version of T_{ij} and indicates the cost of transportation between CDC i and customer j in scenario s.

To penalize the longest distance in the system, we further modify the objective function as seen in (10). We use the abbreviation SLAP-P to denote Single-Stage Stochastic Location-Allocation Problem with Longest Distance Penalty.

SLAP-P Model:

minimize

$$\sum_i F_i * z_i + \sum_i \sum_j \sum_s T_{ijs} * x_{ij} * p_s + pt * LD \qquad (10)$$

subject to

$$\sum_i x_{ij} = D_j \quad \forall\, j \qquad (2)$$

$$\sum_j D_j * x_{ij} \leq C_i * z_i \quad \forall\, i \qquad (3)$$

$$LD \geq T_{ijs} * x_{ij} \quad \forall\, i, j, s \qquad (6)$$

$$x_{ij} \geq 0 \quad \forall\, i, j \qquad (4)$$

$$z_i \in \{0,1\} \quad \forall\, i \qquad (5)$$

The objective function (9) and (10) can actually be simplified easily by replacing the second cost term with:

$$\sum_i \sum_j E_s\left[T_{ijs}\right] * x_{ij} \qquad (11)$$

where $E_s\left[T_{ijs}\right]$ represents the expected distance calculated over all scenarios. This is because the decisions on locations and allocations are made only once, for all scenarios. As the objective function does not distinguish between scenarios, the second term is equivalent to the sum-product of expected distances and assignment decisions.

In Table 3.1, we illustrate the decisions in a Single-Stage Stochastic Models, also known as Expected Value Models. For a location-allocation problem with 10 scenarios, both decisions are made only once, effective for all scenarios. In other words, both the location and allocation decisions are the same across scenarios. This will not be the case in other formulations of the problem, such as Two-Stage Stochastic Models and Wait-and-See Models, both of which will be explained in the coming sections.

Table 3.1: Decisions of the Expected Value Model

Scenarios	Location	Allocation
1	Z	X
2	Z	X
3	Z	X
4	Z	X
5	Z	X
6	Z	X
7	Z	X
8	Z	X
9	Z	X
10	Z	X

When the constraints are not affected by the changes in transportation cost, the stochastic character of the model disappears. In such a case, we just need the expected values of transportation cost and no scenarios. Therefore, the model becomes an "expected value problem" where the parameters under uncertainty are simply replaced with the expected values of those parameters.

On the other hand, when the constraints are affected by the differences in transportation cost, the scenarios should still be used. For instance, the longest distance is calculated considering travel times in each scenario. Thus, using only the expected values of transportation cost would not give the worst-case occurrence of transportation cost. Therefore, scenarios should be kept in use when the longest distance is constrained or penalized.

Figure 3.6 shows the selected CDC locations and customer assignments when the "expected value problem" is considered. As can be seen, there are unusual assignments near the middle of the city, which are circled in green. These assignments are due the way we represented the rush hour traffic congestion in the scenarios in order to hedge against uncertainty.

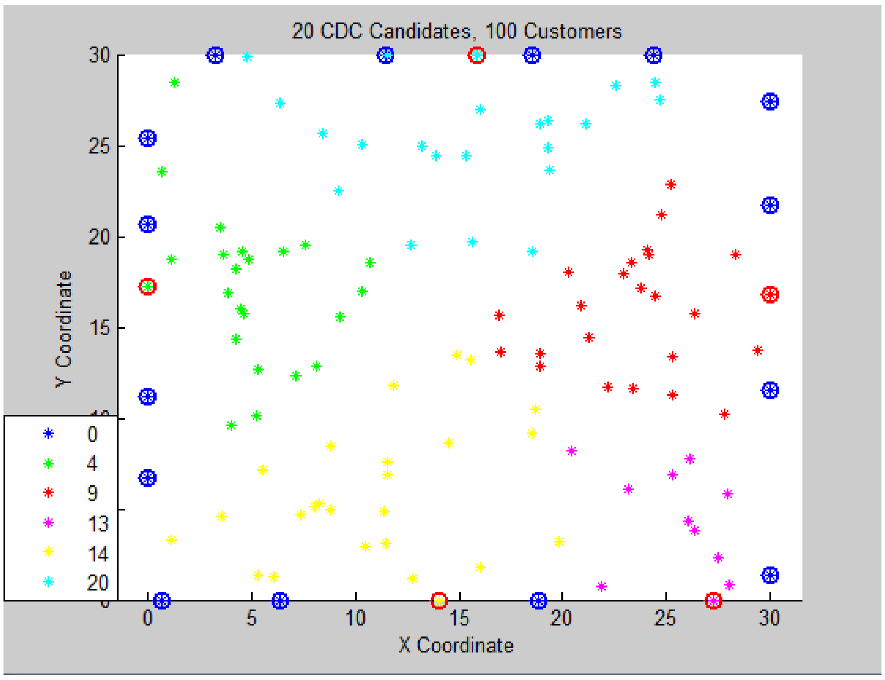


Figure 3.6: Expected Value Solution of a Typical Instance

When the longest distance is penalized, the results change significantly:

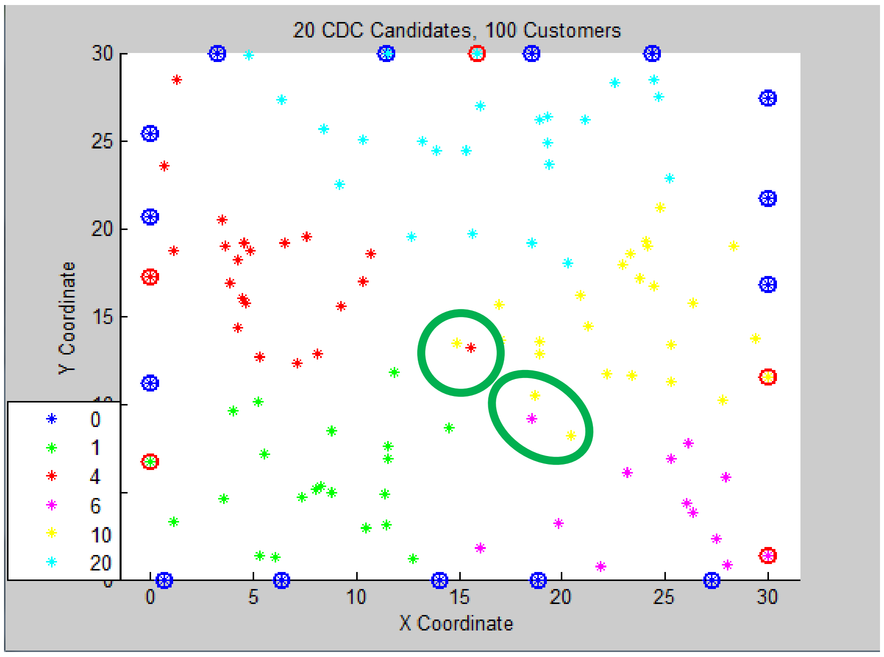


Figure 3.7: Expected Value Solution of a Typical Instance when LD is Penalized

As can be seen in Figures 3.6 and 3.7, penalizing LD has significant effect on the optimal solution. This effect can be seen especially in the selection of CDCs.

3.4 Two-Stage Stochastic Location-Allocation Problem

We can also formulate the location-allocation problem in a two-stage stochastic approach. When such a setting is used, decisions in the first stage are made under uncertainty and decisions in the second stage are made after uncertain parameters are observed. In other words, the first stage decisions are made only once, to be effective for all scenarios. However, the second stage decisions are made specifically for each scenario to be effective only for that particular scenario.

To formulate the location-allocation problem in a two-stage stochastic model, we add the index s to the allocation decision variable x_{ij}. Since location decisions are to be made only once, their representation does not change.

- x_{ijs} determines the amount of demand served from CDC i to customer j in scenario s (continuous variable)

We use the abbreviation TLAP to denote the Two-Stage Stochastic Location-Allocation Problem.

TLAP Model:

minimize

$$\sum_i F_i * z_i + \sum_i \sum_j \sum_s T_{ijs} * x_{ijs} * p_s \qquad (12)$$

subject to

$$\sum_i x_{ijs} = D_j \quad \forall\, j, s \qquad (13)$$

$$\sum_j D_j * x_{ijs} \leq C_i * z_i \quad \forall\, i, s \qquad (14)$$

$$x_{ijs} \geq 0 \;\; \forall\, i, j, s \qquad (15)$$

$$z_i \in \{0,1\} \;\; \forall\, i \qquad (5)$$

The objective function (12) minimizes the total of fixed CDC opening costs and the weighted average of transportation costs across scenarios. x_{ijs} is the modified version of x_{ij} and indicates whether the arc between CDC i and customer j in scenario s used for transportation. Constraints (13)-(15) are modified versions of constraints (2)-(4).

We use the abbreviation TLAP-P to denote the Two-Stage Stochastic Location-Allocation Problem with Longest Distance Penalty.

TLAP-P Model:

minimize

$$\sum_i F_i * z_i + \sum_i \sum_j \sum_s T_{ijs} * x_{ijs} * p_s + pt * LD \qquad (16)$$

subject to

$$\sum_i x_{ijs} = D_j \quad \forall\, j, s \qquad (13)$$

$$\sum_j D_j * x_{ijs} \leq C_i * z_i \quad \forall\, i, s \qquad (14)$$

$$LD \geq T_{ijs} * x_{ijs} \quad \forall\, i, j, s \qquad (17)$$

$$x_{ijs} \geq 0 \;\; \forall\, i, j, s \qquad (15)$$

$$z_i \in \{0,1\} \;\; \forall\, i \qquad (5)$$

The objective function (16) minimizes the total fixed and variable cost items along with the longest distance penalty. Constraints (13)-(15) are modified from (2)-(4) and constraints (17) is modified from (6).

Table 3.2 illustrates the decisions of the Two-Stage Stochastic Model. Parallel to Table 3.1, we observe the location and allocation decisions across 10 scenarios. Unlike the result of the Expected Value Model where a single decision set was effective for all scenarios, the Two-Stage Stochastic Model produces different allocation decisions for each scenario, while keeping location decisions the same.

Table 3.2: Decisions of the Two-Stage Stochastic Model

Scenarios	Location	Allocation
1	Z	X_1
2	Z	X_2
3	Z	X_3
4	Z	X_4
5	Z	X_5
6	Z	X_6
7	Z	X_7
8	Z	X_8
9	Z	X_9
10	Z	X_{10}

Note that there is only a small distinction between the two formulations; however, the logic behind is different. In the single stage stochastic model, we optimize allocation decisions against the expected value of transportation cost, while in the two-stage stochastic model, we can find the optimum allocation decisions for each scenario. Thus, the Two-Stage Stochastic model implies that it is possible to set the second stage decisions according to realized conditions. When there is no recourse capability, single stage stochastic models make more sense, because they work on the assumption that the decisions are made only once. However, in a typical CL system, updating customer assignment is certainly possible. In fact, we would expect such a capability to bring significant benefit in terms of transportation cost.

Making different allocation decisions at each scenario reveals another measure about the performance and applicability of our decisions. In a real-life application, both minimizing transportation cost and minimizing changes in allocation decisions would be desirable, so it is necessary to strike a balance between those conflicting objectives according to the preferences of the decision maker.

In Figure 3.8, we illustrate the change in customer assignments through scenarios. Here, the x-axis represents different customers (from 1 to 100) and the y-axis represents different scenarios (from 1 to 20). The bars are colored to show the number of scenarios in which a customer is assigned to a CDC, different colors representing different CDCs. For instance, customer 90 is assigned to CDC 9 in four scenarios, to CDC 10 in seven scenarios and to CDC 11 in nine scenarios. This graph does not show the assignments made in each scenario. Instead, we are interested in the number of changes in allocation decisions for each customer.

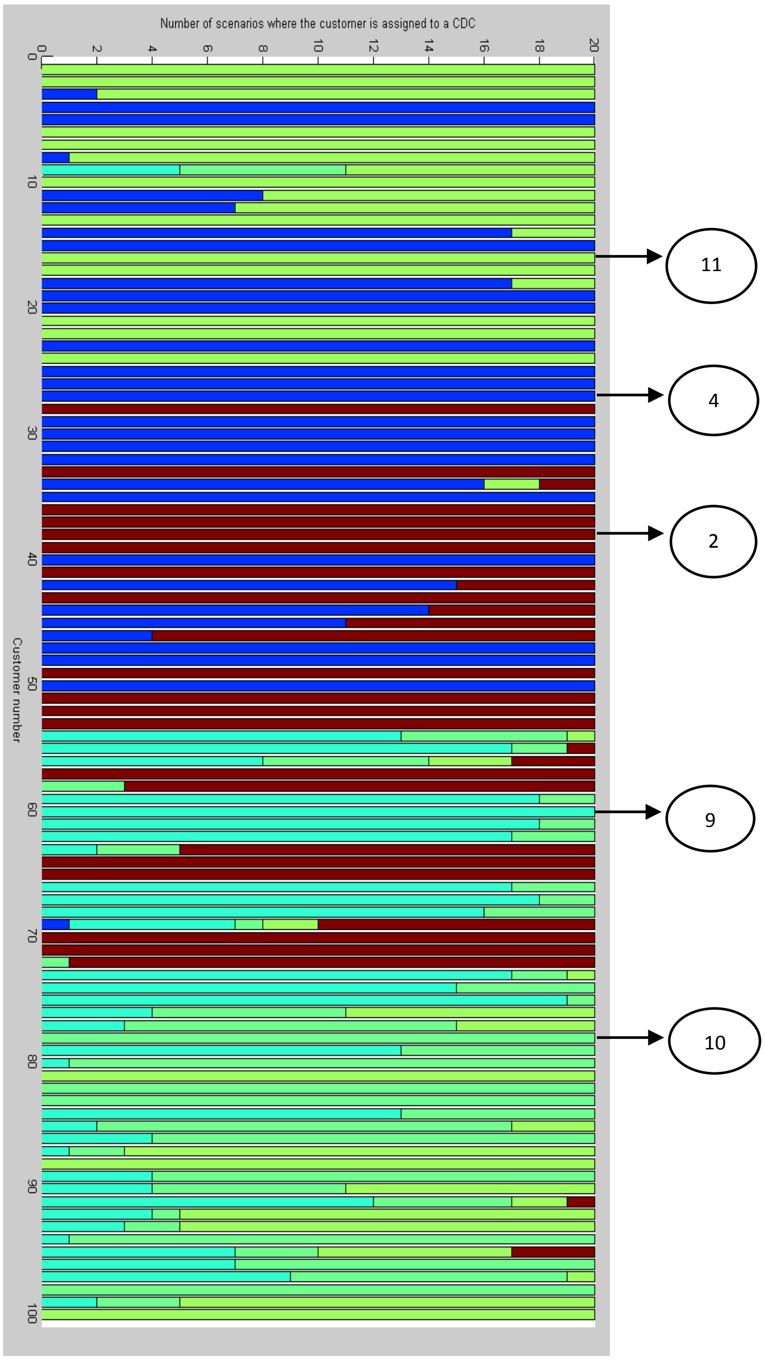


Figure 3.8: Customers with Changing Assignments

We count the changes in the following way. We create a new binary variable that keeps track of whether two allocation variables corresponding to a particular CDC-customer pair are different in two consecutive scenarios. If there is at least one change, this indicator variable stores the information by taking the value one. The sum of these indicator variables equal the number of customers with changing assignments. We only consider the allocation changes across scenarios. We do not consider it a coordination necessity if the customers are serviced by multiple CDCs in a particular scenario.

If changes in the allocation decisions are costly or they are generating some issues that are not considered in the models but could be important in practice, then it is possible to modify the model to restrict or penalize the number of customers with changing CDC assignments. For this purpose, we define indicator variables to distinguish how many times a customer is assigned to a particular CDC through scenarios:

- δ_j identifies the customers assigned to different CDCs

And we add the following set of constraints:

$$\sum_{i_1} \sum_s x_{i_1 js} * i_1 \leq \sum_{i_2} \sum_{s+1} x_{i_2 js+1} * i_2 + M * \delta_j \quad \forall\, j, s \in \{1 \ldots |S| - 1\} \quad (18)$$

$$\sum_{i_1} \sum_s x_{i_1 js} * i_1 \geq \sum_{i_2} \sum_{s+1} x_{i_2 js+1} * i_2 - M * \delta_j \quad \forall\, j, s \in \{1 \ldots |S| - 1\} \quad (19)$$

These constraints force the indicator variable to take value 1 if a customer j is assigned to different CDCs i_1 and i_2 in different scenarios s_1 and s_2.

If a constraint will be added to limit the number of customers assigned to different CDCs, we add the following:

$$\sum_j \delta_j \leq RHS \quad (20)$$

If the number of customers with changing assignments is to be penalized, we change the objective function as follows:

minimize

$$\sum_i F_i * z_i + \sum_i \sum_j \sum_s T_{ijs} * x_{ijs} * p_s + \sum_j \delta_j * penalty \quad (21)$$

Both approaches are valid and one of them can be preferred according to the characteristics of the system of interest. If there is an estimated cost equivalent of the coordination requirement, penalization makes sense. If not, the relationship between coordination requirement and objective function value can be identified by obtaining the non-dominated solution set of the problem.

3.5 Measuring the Value of Using Different Formulations

There are several ways of measuring the value of information when there is uncertainty in one of the parameters.

Expected Value of Perfect Information (EVPI) is the price that the decision maker is willing to pay so that decisions can be made with perfect information. To calculate EVPI, we need to consider two solution approaches and their results.

The first approach requires solving a set of problems called the Wait-and-See problems. In this approach, each Wait-and-See problem corresponds to a particular scenario in which the parameters under uncertainty are replaced with the values given in that particular scenario. In each scenario, both location and allocation decisions are made optimally. Then, the Wait-and-See solution (WS) becomes the expected cost of these solutions (See Table 3.3):

$$WS = E_s[\sum_i z_{is} * F_i + \sum_{i,j} x_{ijs} * D_{ijs}] = \sum_s p_s * \sum_{i,j}(z_{is} * F_i + x_{ijs} * T_{ijs}) \quad (22)$$

Table 3.3: Decisions of the Wait-and-See Model

Scenarios	Location	Allocation
1	Z_1	X_1
2	Z_2	X_2
3	Z_3	X_3
4	Z_4	X_4
5	Z_5	X_5
6	Z_6	X_6
7	Z_7	X_7
8	Z_8	X_8
9	Z_9	X_9
10	Z_{10}	X_{10}

Table 3.3 illustrates the approach. Z_s and X_s refer respectively to the optimal location and allocation decisions in the scenario s.

A different approach is using two-stage stochastic models. In such a setting, first stage decisions are made for all scenarios, but second stage decisions are made specifically for each scenario. Thus, while first stage decisions are fixed, second stage decisions are made optimally for each scenario. Such problems are called Recourse Problems or Here-and-Now Problems. The objective function value obtained in Recourse Problems is the sum of two cost items: RP is the cost of first stage decisions and the expected cost of second stage decisions:

$$RP = E_s[\sum_i z_i * F_i + \sum_{i,j} T_{ijs} * x_{ijs}] = \sum_i z_i * F_i + \sum_s [p_s * \sum_{i,j} T_{ijs} * x_{ijs}] \quad (23)$$

The second measure, Value of Stochastic Solution (VSS) coined by Birge (1982), indicates the benefit of making updates on the second stage decisions. To calculate VSS we need to consider a third kind of problem: the Expected Value Problem (EV), where the parameters under uncertainty are simply replaced by their expected values over all scenarios. Expected cost of using the Expected Value Problem (EEV) is then:

$$EEV = E_s[\sum_i z_i * F_i + \sum_{i,j} E_s(T_{ijs}) * x_{ij}] = \sum_i z_i * F_i + \sum_{i,j} x_{ij} * \sum_s p_s * T_{ijs} \quad (24)$$

Since we are dealing with a minimization problem, the following relation can be easily observed (Madansky, 1960):

$$EEV \geq RP \geq WS \quad (25)$$

Then, for minimization problems, EVPI is found as:

$$EVPI\% = \frac{RP - WS}{WS} * 100 \quad (26)$$

Also, for minimization problems, VSS is found as:

$$VSS\% = \frac{EEV - RP}{RP} * 100 \quad (27)$$

EVPI is a useful measure when the parameters under uncertainty will not keep changing once they are observed. For example, when planning for disaster relief, decisions are made for a single rare event. The parameter values we are interested in are the ones that are observed once the event occurs. Such models are actually planning for a single scenario, but they cannot perfectly predict it. If EVPI is high, more resources should be spent on better predicting that scenario. For a discussion of Value of Information in Humanitarian Logistics, see Noyan (2012) and Doyen et al. (2012).

On the other hand, when city logistics systems are considered, plans are made for the long-term while the transportation costs keep changing every day. Therefore, our decisions must be valid for a number of different scenarios, rather than a single one. In such a case, VSS becomes more useful. This is because VSS measures the cost we avoid if we can change our second stage decisions according to the realization of uncertainty. If VSS is high, more investments can be made for the capability of adapting second stage decisions to the environment. If it is near zero, expected value of parameters can be successfully used to make near optimal decisions. For further discussion on expected value solutions, see Maggioni and Wallace (2012).

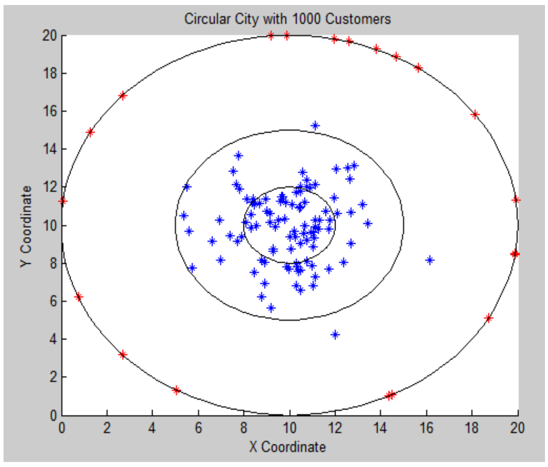
Now we present our calculations on VSS as a percentage of the RP value. Since we have to come up with fixed location decisions, we cannot obtain meaningful WS results. Thus, RP is the best result available to us. Also, as mentioned earlier, the single-stage stochastic model reduces to the Expected Value Problem, when we do not consider any constraint or penalty on the longest distance to be traveled. We can calculate EEV using the solutions of this model. VSS is simply the difference between those values.

Generated Instances

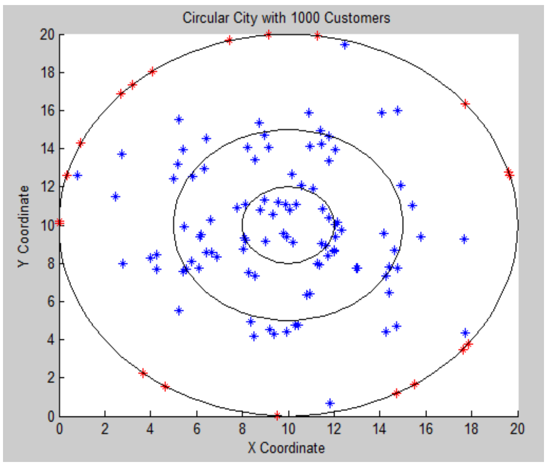
We generated a set of instances each of them including 100 customers and 100 scenarios. These instances are constructed in two main groups according to their shapes: circular and square. Customers in circular instances are generated with normal distribution and these distributions are used for observing the change in VSS with respect to change of standard deviation. We make use of the fact that as the standard deviation increases, normal distribution within a restricted circular area approaches uniform distribution.

We use the following standard deviation values for circular shaped instances: 2, 4, 6 and 8. Figure 3.9 represents each instance respectively. Here the blue stars correspond to the customer locations and red stars indicate candidate CDC sites. In all instances

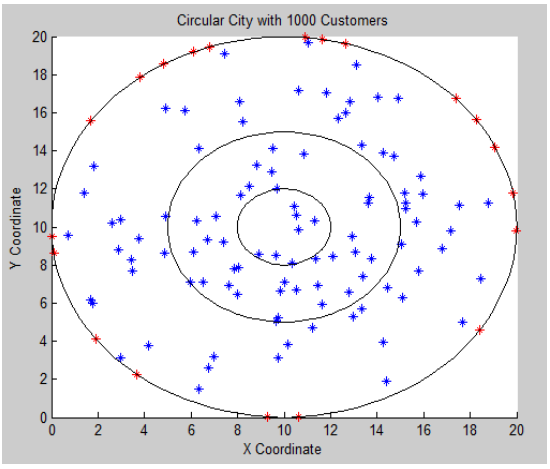
100 customers are distributed normally around the city center. Candidate CDC sites are uniformly distributed in all directions on the city boundary.



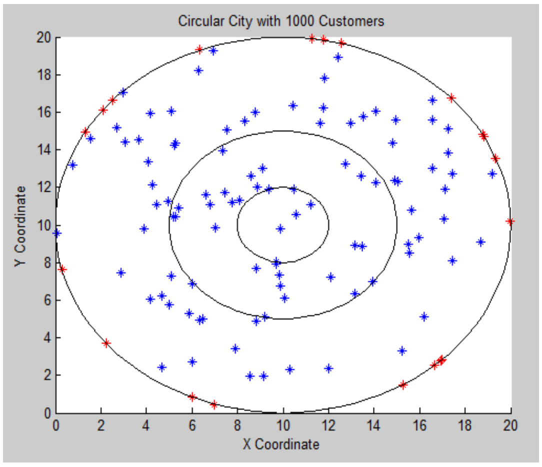
(a) Circular Pattern with σ=2



(b) Circular Pattern with σ=4



(c) Circular Pattern with σ=6



(d) Circular Pattern with σ=8

Figure 3.9: Circular Patterns

Square shaped instances are generated in three patterns (See Figure 3.10): random, clustered and mixed (in line with the Solomon Instances: R, C and RC). Each group has two subgroups with different spatial distributions. Cities in all instances are 30x30 km in size.

- Random Pattern
    - *Uniformly distributed customers (R_u_100)*

      Both customers and CDCs are uniformly distributed along both axes. At each side of the city (north, south, etc.) there are 5 CDCs, which are distributed uniformly along that side.
    - *Normally distributed customers (R_n_100)*

      Customers are normally distributed around a single center (x,y) which is created inside a central square with the corners on (5,5), (5,25), (25,5), and (25,25). Center point is chosen by a normal distribution with parameters $N_X(15,5)$ and $N_Y(15,5)$, so that the instances represent a wider variety of customer patterns. Around the center point, normally distributed customers are created with parameters $N_X(x,7)$ and $N_Y(y,7)$. Customers that are created outside the city boundaries are not represented in the instances. At each side of the city (north, south etc.) there are 5 CDCs, which are distributed uniformly along that side.
- Clustered Pattern
    - *Uniformly distributed customers on normally distributed cluster centers* (*) *(C_u_100)*

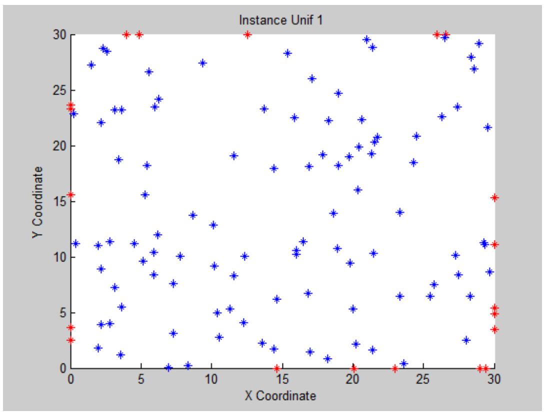
      Customers are uniformly distributed around 10 centers that are created inside a central square with the corners on (2,2), (2,28), (28,2), and (28,28). Center points are distributed normally within the square with parameters $N_X(15,7)$ and $N_Y(15,7)$. Around each center point, 10 uniformly distributed customers are created within the 4x4 km box around the center. At each side of the city (north, south, etc.) there are 5 CDCs, which are distributed uniformly along that side.
    - *Normally distributed customers on normally distributed cluster centers (C_n_100)*

      Customers are normally distributed around 10 centers that are created inside a central square with the corners on (2,2), (2,28), (28,2), and (28,28). Center points are distributed normally within the square with parameters $N_X(15,7)$ and $N_Y(15,7)$. Around each center point, normally distributed customers are created with parameters $N_X(x,2)$ and $N_Y(y,2)$. Customers that are created outside the city boundaries are replaced with new customers. At each side of the city (north, south etc.) there are 5 CDCs, which are distributed uniformly along that side.

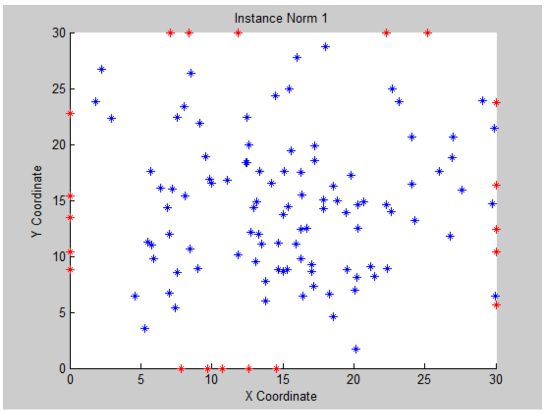
- Mixed Pattern
    - *Uniformly distributed customers along with * (RC_u_100)*

      In these hybrid instances, half of the customers are created with uniform distribution and the other half are created through the process explained in *. At each side of the city (north, south, etc.) there are 5 CDCs, which are distributed uniformly along that side.
    - *Normally distributed customers along with * (RC_n_100)*

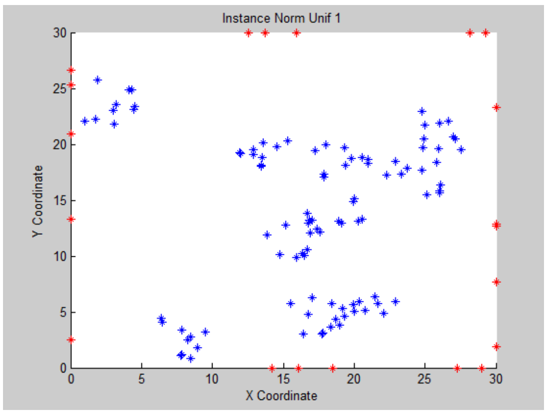
      In these hybrid instances, half of the customers are created with normal distribution around a single center (x,y) that is created inside a central square with the corners on (5,5), (5,25), (25,5), and (25,25). The center point is chosen by a normal distribution with parameters $N_X(15,5)$ and $N_Y(15,5)$, so that the instances represent a wider variety of customer patterns. Around the center point, normally distributed customers are created with parameters $N_X(x,7)$ and $N_Y(y,7)$. The other half are created through the process explained in *. At each side of the city (north, south, etc.) there are 5 CDCs, which are distributed uniformly along that side.



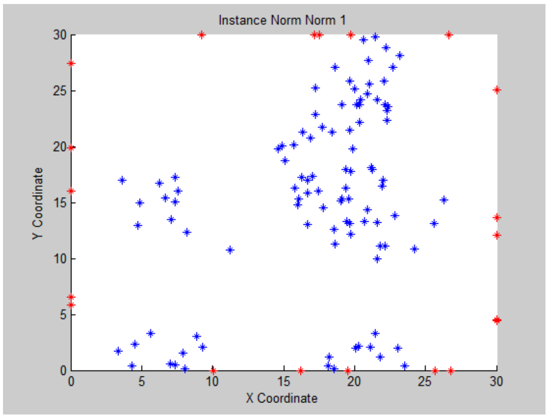
(a) Square Pattern with Unif. Distr.



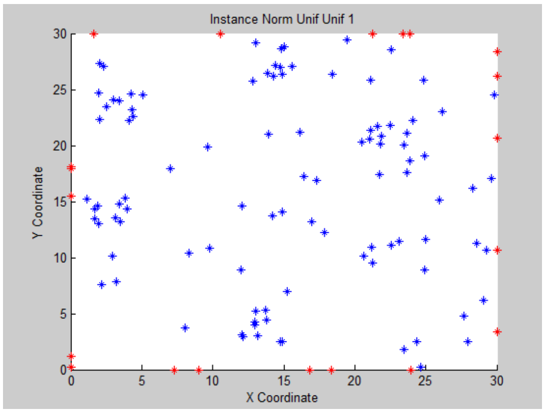
(b) Square Pattern with Normal Distr.

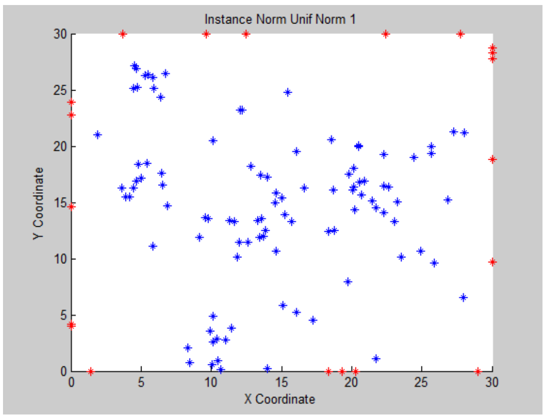


(c) Square Pattern with Unif. Clust.



(d) Square Pattern with Normal Clust.





(e) Square Pattern with Mixed Unif. D. (f) Square Pattern with Mixed Normal D.

Figure 3.10: Square Patterns

From each recipe, we created 20 instances. In order for VIO results to exclusively show the effect of the transportation cost uncertainty, we use the same (unit) demand and capacity values in all instances. Capacity of each CDC is 24 times the unit demand. Longest distance and changes in allocation are not penalized. Now, we present the VSS and EVPI values obtained from the analyses made on these instances.

VSS

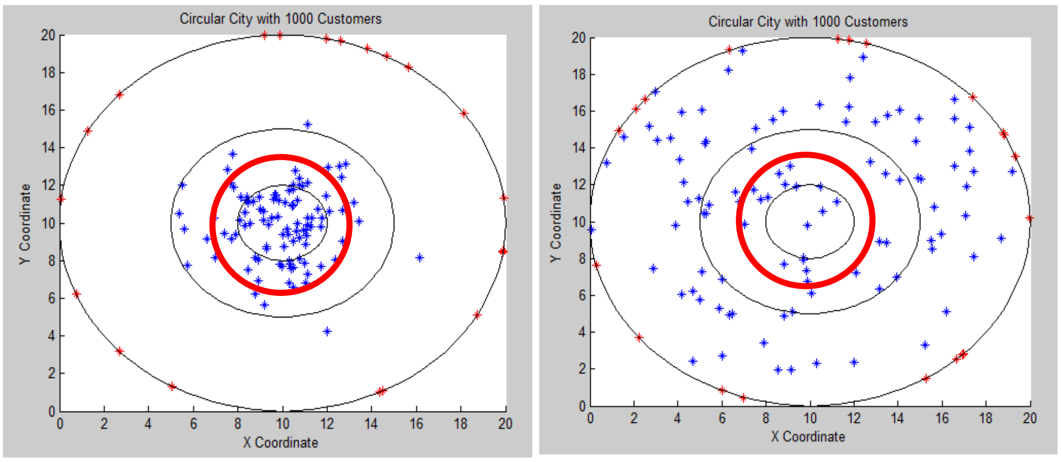
Now we present the statistics on VSS and EVPI values found in our computations for each of the instance groups that contain 20 instances. In Tables 3.4-3.7, we present the average, minimum, maximum and range (minimum-maximum) values for VSS and EVPI of instances.

Table 3.4: VSS Results for Circular Patterns

VSS-Circular	Average(%)	Min(%)	Max(%)	Range(%)
σ=2	18.8	17.8	19.5	1.8
σ=4	12.6	11.7	13.6	1.9
σ=6	9.5	8.3	11.0	2.7
σ=8	9.2	8.7	9.5	0.8

Observe that VSS level with σ=2 is more than twice the VSS level with σ=8 in Table 3.4. Based on VSS values calculated for the circular instances, we can see that the distribution of customer locations significantly changes the level benefit of using a two-stage stochastic approach instead of an expected value approach. In other words, when the standard deviation is lower, it is more important for the CL system to be able to change the allocation decisions when encountered with different scenarios.

The explanation for this observation is not solely based on the difference between normal and uniform distribution. Because a more important difference than the distribution is the customer density in the city center. We make the comparison between instances lowest and highest standard deviation with the help of Figure 3.11:



(a) Circular Pattern with σ=2 (b) Circular Pattern with σ=8

Figure 3.11: Comparison of City Center Customer Density in Circular Patterns

Figure 3.11 represents the instances with the lowest (σ=2) and highest (σ=8) standard deviation. The red circles roughly indicate the region we consider the city center. The significance of customer density in the city center is due to CDCs locations being on the city boundaries. Since CDCs are located far from the city center, there is a higher possibility for customers in the city center to have a different closest CDC in different scenarios.

For instance, if we consider a customer near the city boundary, we would expect it to have a fixed assignment to the closest CDC. On the other hand, if a customer in the city center is considered, since it is farther away from all CDCs, its assignment would probably change depending on the scenarios. As a result, when there are more customers in the city center, there is higher value that can be obtained from allowing CDC assignments to change over time, which is reflected by the higher VSS.

Table 3.5: VSS Results for Square Patterns

VSS-Square	Average(%)	Min(%)	Max(%)	Range(%)
R_u_100	8.9	7.0	10.8	3.9
R_n_100	13.1	11.0	15.8	4.8
C_u_100	10.6	5.8	17.3	11.5
C_n_100	13.1	10.3	16.9	6.6
RC_u_100	10.1	6.2	13.1	6.9
RC_n_100	11.1	7.3	13.7	6.5

Now, we analyze the instances with square patterns. While EVPI values do not differ greatly between instances, VSS values show large variation. We can consider the instances R_u_100, C_u_100 and C_n_100 to be the extremes of the pool of instances, since the former provides the most uniform spatial distribution of the customers and the latter does the opposite. Therefore, it is not surprising to observe the large difference between their VSS values as their average values are the highest and lowest. It is also expected that the mixed instances' values lie between the two extremes. The instances called R_n_100 produce high VSS values since the distribution of customers is far from being uniform through the city, rather the number of customers is much higher near the city center than in the city outskirts. Therefore, a small change in transportation cost may change the optimal allocation decision for more customers.

The R_u_100 instances show the lowest range. However, the results from clustered instances differ greatly according to the locations of cluster centers. We observe the largest range in C_u_100 and not in C_n_100 because the customers are located more closely to the cluster centers in the former, while they are allowed to deviate more from the cluster centers (an effect of normal distribution) in the latter. Therefore, in C_u_100 the VSS values depend largely on the selection of cluster centers, but the effect of cluster centers is smaller in C_n_100. VSS ranges for mixed instances are again between the extremes.

An interesting remark is that when we compare the circular instance with highest standard deviation (which can be considered closest to uniform) and the square instance with random uniform distribution, we observe similar VSS results, which may indicate that the effect of city's shape is negligible compared to the effect of customer distribution.

EVPI

When we look at the EVPI results given in Tables 3.6-3.7, the most important observation is that the average EVPI values are always significantly smaller than the average VSS values. Table 3.6 summarizes EVPI results for the circular instances.

Table 3.6: EVPI Results for Circular Patterns

EVPI-Circular	Average(%)	Min(%)	Max(%)	Range(%)
σ=2	3.4	3.0	3.7	0.7
σ=4	3.1	2.7	3.5	0.9
σ=6	2.8	2.5	3.0	0.5
σ=8	2.9	2.3	3.2	0.8

Similar to the VSS results, we again observe a decrease in EVPI results as the standard deviation increases for the instances with circular patterns. This decrease, however, is not as large as it was in the VSS case, but it is still significant. The reason for the decrease is the same as before; the transportation costs of customers in the city center are less stable. Thus, when the customer density in the city center is higher, we expect the collective change in the transportation cost matrix to have more impact on the selection of candidate CDCs.

Table 3.7: EVPI Results for Square Patterns

EVPI-Square	Average(%)	Min(%)	Max(%)	Range(%)
R_u_100	2.9	1.5	4.6	3.1
R_n_100	2.4	1.1	3.5	2.3
C_u_100	2.2	0.9	3.4	2.5
C_n_100	2.6	1.5	4.0	2.5
RC_u_100	2.3	1.2	3.2	2.0
RC_n_100	2.5	1.1	4.0	2.9

Similar comments can be made about the EVPI values for square instances. Again, the instances called R_u_100 and C_u_100 seem to be the extreme cases, while others' EVPI values are in-between.

When the VSS and EVPI values are compared, we can easily see that the difference between RP and EEV approaches are much more significant than the difference between WS and RP approaches. This means that most of the benefit from having perfect information can be obtained just by updating allocation decisions for each scenario. We must also note that periodically changing customer allocations according to system conditions would be possible in a real-life system, whereas it is surely impossible to change the locations of CDCs in the short term; thus, the best realistic solution is obtained by the two-stage stochastic solution. Based on these observations, we will use the two-stage approach solving the CL network design problem under cost uncertainty.

3.6 Significance of Value of Information in City Logistics

It is clear that the ability to make allocation decisions specifically for each scenario brings great benefit in a City Logistics system. However, we also wanted to analyze the impact of locating CDCs on the city boundaries, which is a defining feature of City Logistics systems. Therefore, we compared the VSS and EVPI values obtained from instances with CDCs on the city boundaries and with CDCs located inside the city. Only changing the candidate CDC locations are changed. New locations are selected using uniform distribution on both x and y axes. The Tables 3.8-3.9 show the average values over 20 instances from each group:

Table 3.8: Comparison of VSS in CL vs. Classical Location Setting

VSS-Square	City Logistics(%)	Classical LP(%)
R_u_100	8.9	5.7
R_n_100	13.1	8.3
C_u_100	10.6	6.9
C_n_100	13.1	4.4
RC_u_100	10.1	5.4
RC_n_100	11.1	6.8

The values in the “City Logistics” column are the same as the ones explained in the previous section. The values in the “Classical LP” column, however, are obtained after we modified the instances by locating the candidate CDCs uniformly inside the city.

There is a clear difference between VSS values in the CL setting and the classical facility location setting for every instance group. The reason for this observation can be explained in the following way: When the CDCs are located on the city boundaries, their distance to the customers near the city center do not vary greatly; thus, small changes in transportation cost may change the optimal allocation decision for a customer. Therefore, the two-stage stochastic approach produces significantly better results, providing optimal allocations for such customers, while the expected value approach cannot. On the other hand, when some of the CDCs are located close to the city center, the customers near the city center are automatically allocated to those CDCs. Thus, there are fewer customers that would benefit from allocation changes, and the benefit of using the two-stage stochastic approach is reduced.

Table 3.9: Comparison of EVPI in CL vs. Classical Location Setting

EVPI-Square	City Logistics(%)	Classical LP(%)
R_u_100	2.9	2.6
R_n_100	2.4	1.6
C_u_100	2.2	1.2
C_n_100	2.6	0.5
RC_u_100	2.3	2.3
RC_n_100	2.5	0.7

We can see significant differences also for the EVPI values for most instance groups. The benefit of having perfect information on the transportation cost parameter is reduced when CDCs are located inside the city, because in such a case, not selecting the best CDC for a particular scenario does not significantly increase the transportation cost.

Another argument could be made on the relative distance of customers and facilities. In the classical location literature, the customers are closer to facilities in all instances, since facilities are located within the region. As we have seen in Figure 3.11, higher standard deviation values for customer location distribution create smaller distances between customers and CDCs and vice versa. Thus, the instances with high standard deviation are closer in behavior to the classical instances. We observe that such instances produce smaller VSS, which is parallel to our conclusion that VSS is larger in the CL context.

From these observations, we conclude that using two-stage stochastic formulation provides larger benefit in a City Logistics location problem than in a classical facility location problem.

3.7 The Tradeoff Between Transportation Cost and Coordination Cost

As explained in the previous subsections, the ability to change customer assignments according to changing conditions brings large benefits. While changing assignments would be possible in a real-life system, it would come with a coordination cost due to the expenses of maintaining the organization despite the changes. If the assignment decisions are to be changed periodically, the corresponding stakeholders of the system need to update their shipments arrangements accordingly.

It is difficult to estimate the magnitude of coordination costs in a real-life system and it would not be meaningful to compare the coordination costs to transportation costs as different decision makers would have different priorities. However, we can compare the necessity of coordination in terms of the number of customers with changing assignments. We may assume the number of customers with changing assignments to be proportional to the coordination cost in a real-life system. Table 3.10 summarizes our findings on the coordination necessity. We report the average proportion of customers with changing assignments within the whole customer population, along with minimum, maximum and range values.

Table 3.10: Number of Customers with Assignment Change

Customers with Assignment Change	Average(%)	Min(%)	Max(%)	Range(%)
R_u_100	82.9	79	89	10
R_n_100	94.7	88	99	11
C_u_100	92.0	80	100	20
C_n_100	92.3	84	99	15
RC_u_100	87.9	77	96	19
RC_n_100	90.5	85	97	12

Here we see that the coordination requirement is significantly affected by the spatial distribution and pattern of customers. We expect this requirement to be lower when a larger proportion of the customers are located close to the city boundaries, since only large changes in travel times would change the allocation decisions for these customers. Among our instance groups, the ones called R_u_100, RC_u_100 and RC_u_100 are the ones that consistently place more customers far from the city center. Thus it is expected that these instances result in a higher average number of stable assignments. Conversely, the instances called R_n_100 put most of the customers near city center. The instances called C_u_100 and C_n_100 vary according to the selection of cluster centers; therefore, their values are between the two extremes.

Table 3.11: Customers with Assignment Change without Congested Days

Customers with Assignment Change	Average(%)	Average w/o Congested Days(%)
R_u_100	82.9	63.7
R_n_100	94.7	87.1
C_u_100	92.0	84.0
C_n_100	92.3	84.9
RC_u_100	87.9	77.5
RC_n_100	90.5	82.7

When we exclude the congested days from our scenarios (See Table 3.11), we expectedly find a significantly larger number of stable assignments. Since the travel times are not changed as much as in the previous case, we obtain roughly two times the number of stable assignments.

Table 3.12: Customers with Assignment Change under Classical Location Setting

Customers with Assignment Change	City Logistics(%)	Classical LP(%)
R_u_100	82.9	81.9
R_n_100	94.7	84.7
C_u_100	92.0	77.2
C_n_100	92.3	74.0
RC_u_100	87.9	85.8
RC_n_100	90.5	81.7

If the CDCs were located within city, the results are again different (See Table 3.12). This time clustered instances have significantly lower coordination requirements, because clusters are simply assigned to CDCs with all their members; larger changes in travel times are needed for changing assignments, thus changes occur less often. We also observe an increase in the number of stable assignments when an instance includes more customers close to the city center. Due to the same reason, the closely packed customers are assigned as a whole. On the other hand, uniformly distributed instances are not significantly affected from the change.

In conclusion, the results and observations in this chapter made it clear that a City Logistics system must be adaptable to the ever-changing city environment. Cities are dynamic organizations that evolve over time according to the needs and actions of their inhabitants. The quality, capacity and coverage of the road network are among the most important characteristics of a city that participate in the evolution process. These characteristics are the subject of our interest in our analyses on the travel time uncertainty. Our findings show that an effective City Logistics system can only be implemented if the tactical and operational level decisions are updated in reaction to the changes on the traffic conditions.

CHAPTER 4

EXACT SOLUTION METHODS AND PRELIMINARY COMPUTATIONAL RESULTS

In the previous section, we have seen that significant gains can be achieved when we allow the allocation decisions to change over time. Since such an approach would be realistic, we apply the two-stage stochastic formulation to the city logistics system design problem.

Recall that the location-allocation problem is NP-hard (Snyder, 2006) and a small increase in the number of customers results in a large increase in solution time. Since we consider uncertainty through scenarios and need to make assignment decisions specifically for each scenario, the computational complexity of the problem is higher than its deterministic counterpart. For example, the instances considered in Chapter 3 include 20 binary and 100 continuous variables when the expected value formulation is used and 20 binary and 10000 continuous variables when the two-stage stochastic formulation is used (with 100 scenarios). To be able to deal with instances of realistic size, we need to employ suitable solution methods. The largest instances that can be solved under 2 hours include around 100 customers, which is clearly not a sufficiently large customer base for a CL system.

Benders (1962) introduced Benders Decomposition for mixed integer problems. In order to avoid the large number of variables, Benders suggests decomposing the variables into two sets: continuous variables and complicating (binary/integer) variables. The method requires construction of a Master Problem (MP) including the complicating variables and a Sub-Problem (SP) including the continuous variables. At each iteration, SP creates the cuts that constrain the solution space for MP and MP makes guesses on the optimal levels of complicating variables and sends them to SP.

The algorithm converges in finite number of iterations and MP reaches a solution that is equal/very close to the upper bound found by SP.

In this chapter, we construct and implement several Benders Decomposition algorithm variants for computational enhancements.

The following is the optimization model of the city logistic system design problem under cost uncertainty:

Variables

- z_i determines if the candidate CDC location i is selected or not (binary variable)
- x_{ijs} determines the amount of demand served from CDC i to customer j in scenario s (continuous variable)

Parameters

- D_j, demand from customer j
- C_i, capacity of CDC i
- T_{ijs}, travel time between CDC i and customer j in scenario s
- F_i, fixed cost of opening and operating CDC i
- P_s, probability of scenario s

Model – SLP2S

min

$$\sum_i F_i * z_i + \sum_i \sum_j \sum_s T_{ijs} * x_{ijs} * P_s \qquad (1.1)$$

subject to

$$\sum_j x_{ijs} \leq C_i * z_i \quad \forall\, is \qquad (1.2)$$

$$\sum_i x_{ijs} = D_j \quad \forall\, js \qquad (1.3)$$

$$z_i \in \{0,1\} \quad \forall\, i \qquad (1.4)$$

$$x_{ijs} \geq 0 \quad \forall\, ijs \qquad (1.5)$$

The objective function minimizes the sum of fixed cost of opening and operating CDCs and variable cost of making deliveries to customers. Constraints (1.2) ensure that the CDCs in service are open, constraints (1.3) guarantee the demand of each customer to be satisfied and constraints (1.4)-(1.5) describe the domain of variables.

4.1 Benders Decomposition

Although Benders Decomposition is an effective approach when dealing with large-scale mixed integer programming (MIP) problems and it has been used for decades on many kinds of optimization problems, it also presents some difficulties. Benders Decomposition algorithms often require very high solution times for convergence, due to the fact that the algorithm may need to run for thousands of iterations (You and Grossman, 2013). When the algorithm requires many iterations for convergence, both Master Problem (MP) and Sub-problem (SP) need to be solved many times, plus the solution time increases as the complexity of these problems increase.

We first observe the advantages and drawbacks of the Benders Decomposition algorithm before proceeding with enhancements on it. The following figure illustrates how we decompose the variables of our problem:

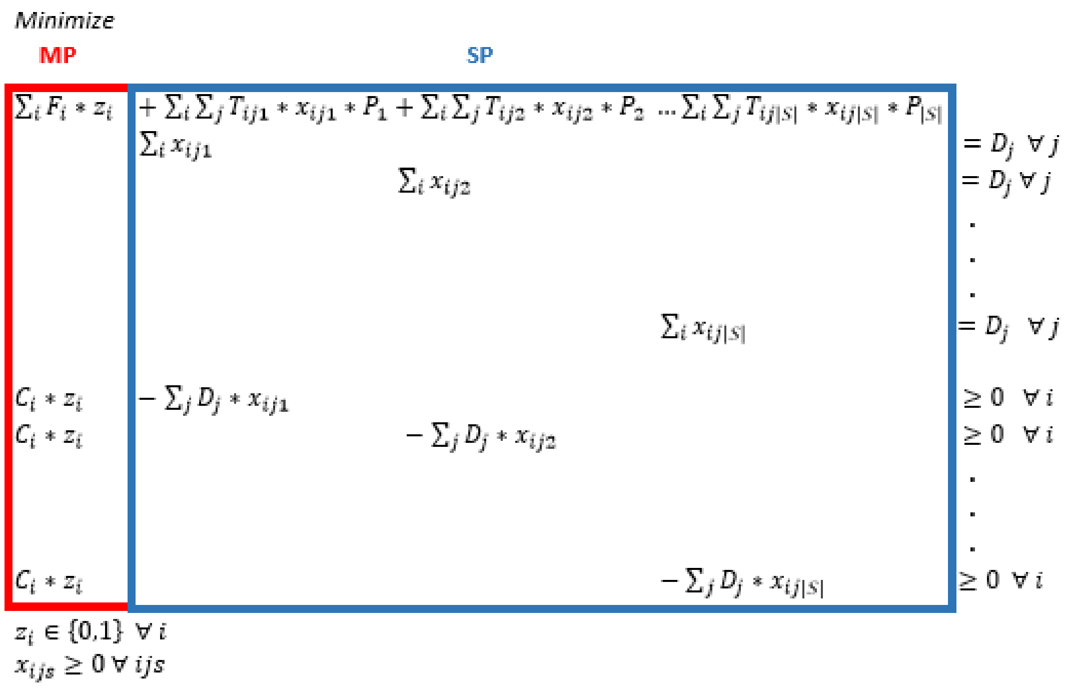


Figure 4.1: Repr. of the Model SLP2S and the Benders Decomposition Setting

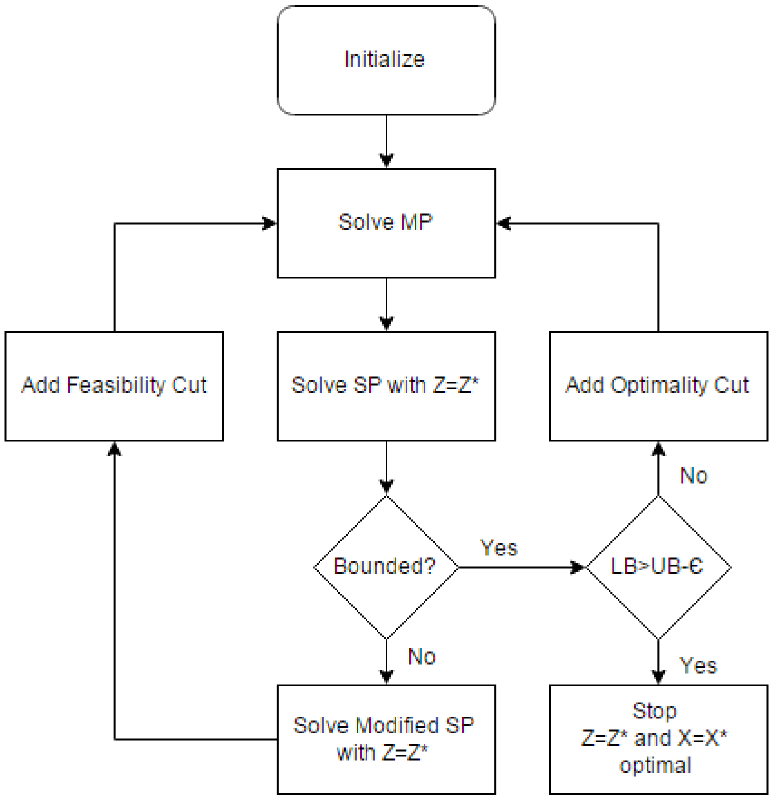


Figure 4.2: Benders Decomposition Algorithm (BD)

Below we give the necessary definitions to be used in this chapter.

BD Subproblem

- u_{is}, positive dual variable corresponding to (1.2)
- v_{js}, free dual variable corresponding to (1.3)
- z_i', constant that keeps the location decisions made in MP
- z_{SP}, objective function value of subproblem

max

$$z_{SP} = \sum_i \sum_s -C_i * u_{is} * z_i' + \sum_j \sum_s D_j * v_{js} \quad (2.1)$$

subject to

$$-u_{is} + v_{js} = T_{ijs} * P_s \quad \forall\, ijs \quad (2.2)$$

$$u_{is} \geq 0 \quad \forall\, is \quad (2.3)$$

$$v_{js}\ URS \quad \forall\, js \quad (2.4)$$

$$z_{SP}\ URS \quad (2.5)$$

BD Modified Subproblem

max

$$0 \quad (2.6)$$

subject to

$$\sum_i \sum_s -C_i * u_{is} * z_i' + \sum_j \sum_s D_j * v_{js} = 1 \quad (2.7)$$

$$-u_{is} + v_{js} \leq 0 \quad \forall\, ijs \quad (2.8)$$

$$u_{is} \geq 0 \quad \forall\, is \quad (2.9)$$

$$v_{js}\ URS \quad \forall\, js \quad (2.10)$$

BD Cut Generation

- *oc*, set of optimality cuts
- *fc*, set of feasibility cuts
- $c1_{oc}$ and $c1_{fc}$, constant value in a particular cut
- $c2_{oc,i}$ and $c2_{fc,i}$, variable coefficient in a particular cut
- u'_{is} , constant that keeps the corresponding decisions made in SP or MSP
- v'_{js} , constant that keeps the corresponding decisions made in SP or MSP

Optimality cuts and feasibility cuts are generated in the same way. For optimality cuts:

$$c1_{oc} = \sum_j \sum_s D_j * v'_{js} \tag{2.11}$$

$$c2_{oc,i} = \sum_s -C_i * u'_{is} \tag{2.12}$$

For feasibility cuts:

$$c1_{fc} = \sum_j \sum_s D_j * v'_{js} \tag{2.13}$$

$$c2_{fc,i} = \sum_s -C_i * u'_{is} \tag{2.14}$$

BD Master Problem

- z_i, binary variable that indicates whether the CDC location i is selected or not
- z_{MP}, objective function value of master problem

min

$$z_{MP} \tag{2.15}$$

subject to

$$z_{MP} \geq \sum_i F_i * z_i + c1_{oc} + \sum_i c2_{oc,i} * z_i \quad \forall\, oc \tag{2.16}$$

$$c1_{fc} + \sum_i c2_{fc,i} * z_i \leq 0 \quad \forall\, fc \tag{2.17}$$

$$z_i \in \{0,1\} \quad \forall\, i \tag{2.18}$$

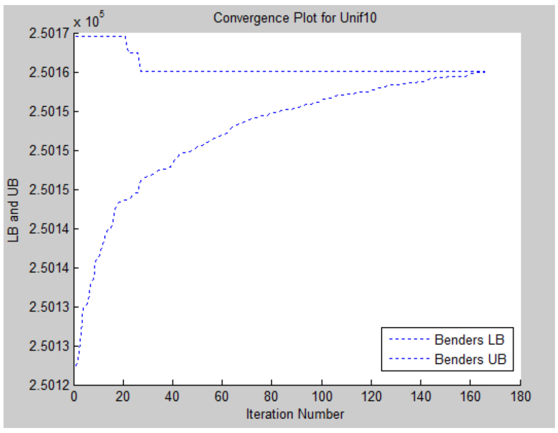
$$z_{MP}\ URS \tag{2.19}$$

Algorithm 4.1 Benders Decomposition Algorithm

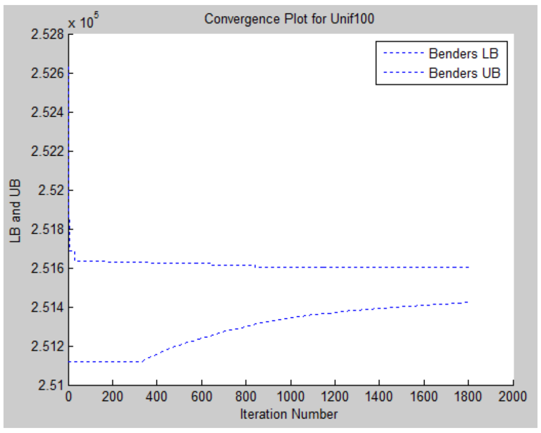
```
Input: $T_{ijs}, D_j, C_i, F_i, p_s, \epsilon$, Time limit
Output: Gap, best solution generated
 1: $iter \leftarrow 0, UB \leftarrow +INF, LB \leftarrow -INF$
 2: while $UB - LB > \epsilon$ do
 3:     $iter \leftarrow iter + 1$
 4:     solve (2.1) - (2.5)
 5:     if $z_{SP} < INF$ then
 6:         if $\sum_i z_i' * F_i + z_{SP} < UB$ then
 7:             $UB \leftarrow \sum_i z_i' * F_i + z_{SP}$
 8:             $Best \leftarrow Z$
 9:         end if
10:         generate cut from (2.11) - (2.12)
11:     else
12:         solve (2.6) - (2.10)
13:         generate cut from (2.13) - (2.14)
14:     end if
15:     solve (2.15) - (2.19)
16:     $LB \leftarrow z_{MP}$
17: end while
```

To observe convergence of the algorithm, we experiment with three groups of instances: small (10 customers), medium (100 customers) and large (1000 customers). All instances include 20 candidate CDC locations and 100 scenarios.

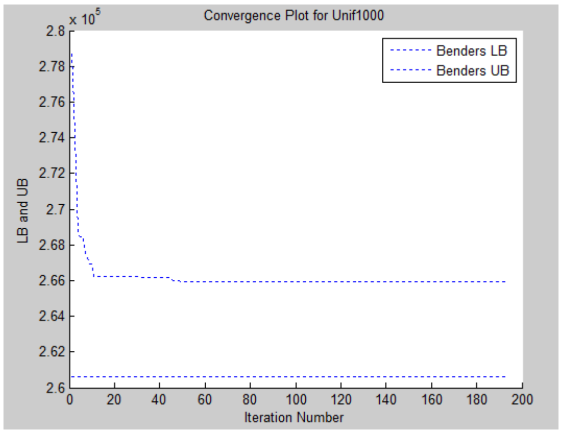
We examine the convergence behavior of the algorithm using some numerical examples. The following figures are the convergence plots of such instances. The runs were limited to 4 hours. We observed that only two feasibility cuts are added in the second and third iterations of the algorithm. Addition of these cuts could be avoided if a simple constraint ensuring the feasibility by forcing total capacity to be more than total demand. However, we did not add such a cut, since the strength of feasibility cuts are stronger than this constraint.



(a) Convergence of BD with small inst.



(b) Convergence of BD with medium inst.



(c) Convergence of BD with large inst.

Figure 4.3: Convergence of BD

As can be seen in Figure 4.3, the algorithm requires about 180 iterations even for the small instance and CPU time requirement is 92 seconds. For the middle size instance, we see that the LB does not increase quickly and the algorithm does not converge in 4 hours. Same is true for the large instance where the LB increases only a little through all the iterations. We also observe an important detail in the plot of the large instance, since only around 120 iterations could be completed in 4 hours, while over 1800 iterations could be completed for the middle instance. Size of SP must be the reason for this observation, since we keep the number of binary variables constant for all instances, and size of MP does not change. To ensure that the size of SP is not affected this much from the changes in the instance size, we apply the L-Shaped method.

4.2 L-Shaped Method

Slyke and Wets (1969) proposed the L-shaped method to further decompose SP into smaller problems, each corresponding to a scenario of the problem. To show how this is possible, we present the L-Shaped method representation in Figures 4.4-4.5:

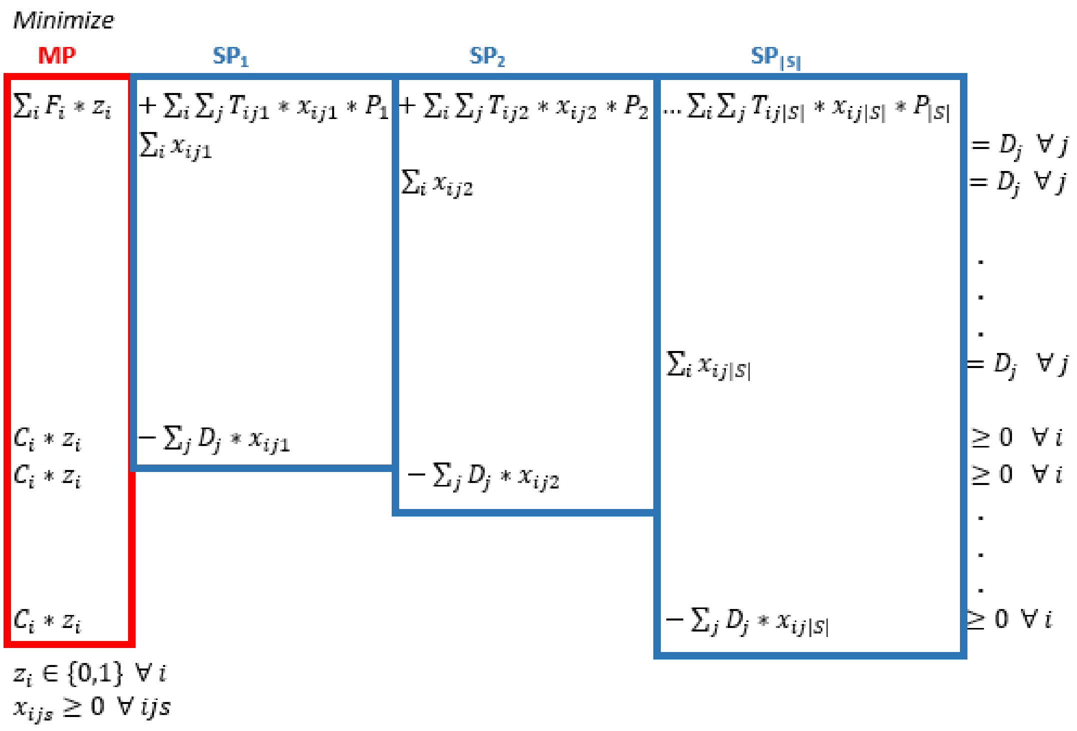


Figure 4.4: Repr. of the Model SLP2S and the Lshaped Decomposition Setting

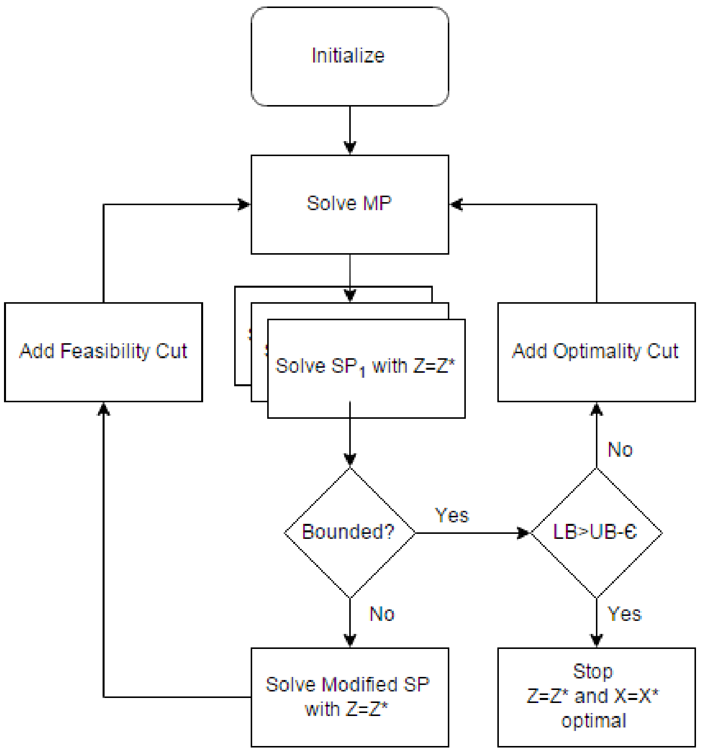


Figure 4.5: L-Shaped Algorithm (LS)

Note that the SP is decomposed into many smaller problems. The following changes are made on the algorithm to decompose SP with respect to scenarios:

LS Algorithm Subproblem (s)

- z_{SP_s}, objective function value of subproblem s

max

$$z_{SP_s} = \sum_i -C_i * u_{is} * z_i' + \sum_j D_j * v_{js} \qquad (3.1)$$

subject to

$$-u_{is} + v_{js} = T_{ijs} * P_s \quad \forall\, ij \qquad (3.2)$$

$$u_{is} \geq 0 \quad \forall\, i \qquad (3.3)$$

$$v_{js}\ URS \quad \forall\, j \qquad (3.4)$$

$$z_{SP_s}\ URS \qquad (3.5)$$

LS Algorithm Modified Subproblem (s)

max

$$0 \tag{3.6}$$

subject to

$$\sum_i -C_i * u_{is} * z_i' + \sum_j D_j * v_{js} = 1 \tag{3.7}$$

$$-u_{is} + v_{js} \leq 0 \quad \forall\, ij \tag{3.8}$$

$$u_{is} \geq 0 \quad \forall\, i \tag{3.9}$$

$$v_{js}\ URS \quad \forall\, j \tag{3.10}$$

LS Cut Generation

Same as BD Cut Generation.

LS Master Problem

Same as BD Master Problem.

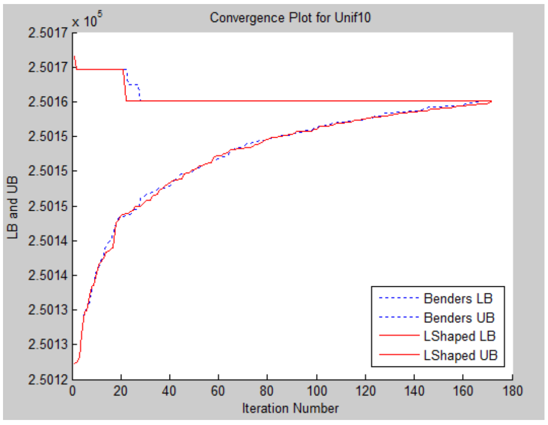
Algorithm 4.2 L-Shaped Algorithm

Input: $T_{ijs}, D_j, C_i, F_i, p_s, \epsilon$, Time limit

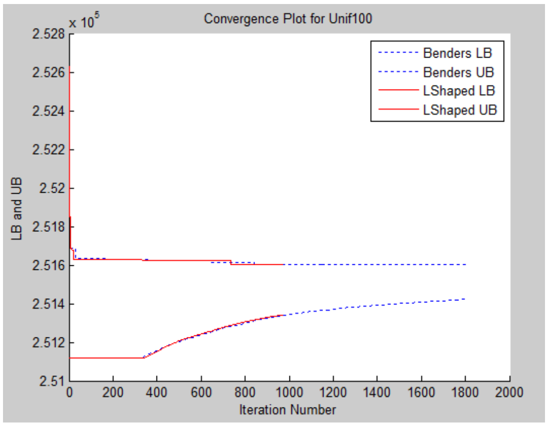
Output: Gap, best solution generated

```
 1: $iter \leftarrow 0, UB \leftarrow +INF, LB \leftarrow -INF$
 2: while $UB - LB > \epsilon$ do
 3:     for $s = 1\ to\ |S|$ do
 4:         $iter \leftarrow iter + 1$
 5:         solve (3.1) - (3.5)
 6:     end for
 7:     if $\sum_s z_{SP_s} < INF$ then
 8:         if $\sum_i z_i{}' * F_i + \sum_s z_{SP_s} < UB$ then
 9:             $UB \leftarrow \sum_i z_i{}' * F_i + \sum_s z_{SP_s}$
10:             $Best \leftarrow Z$
11:         end if
12:         generate cut from (2.11) - (2.12)
13:     else
14:         for $s = 1\ to\ |S|$ do
15:             solve (3.6) - (3.10)
16:         end for
17:         generate cut from (2.13) - (2.14)
18:     end if
19:     solve (2.15) - (2.19)
20:     $LB \leftarrow z_{MP}$
21: end while
```

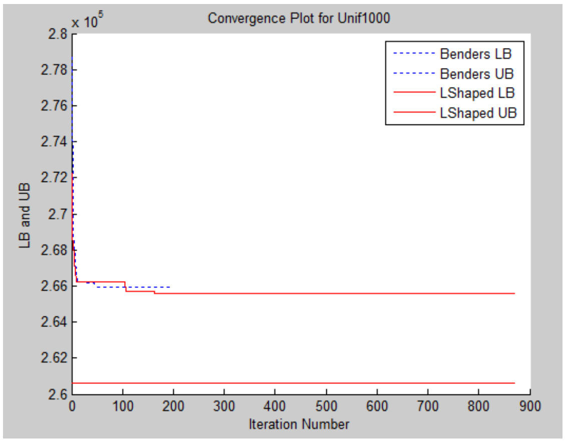
Using this method, we significantly reduce the size of SP and obtain |S| small SPs. When the instance size increases, sizes of these small SPs increase much slower than with the SP of Benders Decomposition. The following are the convergence plots of the L-Shaped method for the same three instances. To be able to compare the Benders algorithm with the L-Shaped method, we plot their results together:



(a) Convergence of LS with small inst.



(b) Convergence of LS with medium inst.



(c) Convergence of LS with large inst.

Figure 4.6: Convergence of LS

As expected, the number of iterations did not change when we used L-Shaped method. However, the solution time is much higher this time; because the SP was already small and defining 100 SPs, which are sequentially solved at each iteration, was not efficient. Thus, the solution time increased from 92 seconds to 1432 seconds. It seems that the Benders algorithm gives better results, for small instances.

For the medium instance, we observe that the L-Shaped method does not do well since the instance is not large enough to complete iterations more quickly with |S| smaller SPs. Since the convergence does not occur until time limit, we can only observe the number of iterations, and how much the LB could be increased. Again both bounds of the two methods follow the same trajectories as expected.

However, a significant improvement could be observed when L-Shaped method is used for the large instances. Since SP of the large instance is very large, replacing it with small SPs provide great benefit. The number of iterations that we could observe increased from 126 to 877 in this way, with the same time limit.

4.2.1 Multi-Cut

The reason for slow convergence of these algorithms is mainly due to the weakness of the cuts generated by SPs. At each iteration we add one cut to MP, an optimality cut or a feasibility cut, depending on SP being bounded/unbounded. A possible solution to this problem could be adding multiple cuts at each iteration. Although these cuts would be weak individually, their combined effect would hopefully achieve a faster convergence.

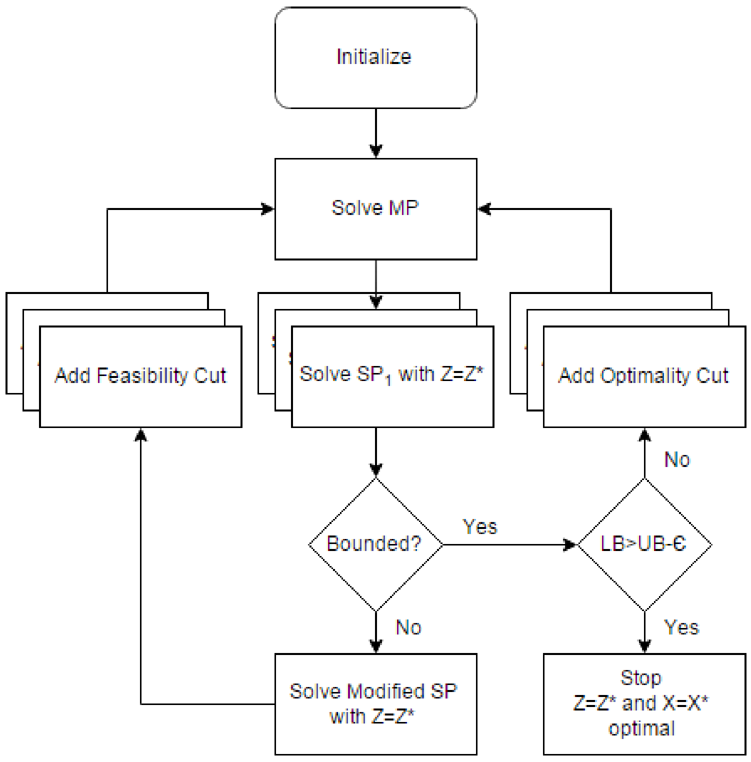


Figure 4.7: L-Shaped Multi-Cut Algorithm (MC)

Birge and Louveaux (1988) and Birge and Louveaux (2011) propose the generation multiple cuts, based on the L-Shaped method. Normally, we aggregate the dual prices found by the |S| many SPs in the L-Shaped method, and create a single cut at each iteration. The proposal is creating cuts out of each SP separately instead of aggregating the cut data. Oliveira et al. (2014) and You and Grossman (2013) also use this method when solving a two-stage stochastic model of a stochastic supply chain problem.

The following changes are made on the algorithm to allow for the creation of multiple cuts (See Figure 4.7):

MC Subproblem (s)

Same as LS Subproblem(s).

MC Modified Subproblem (s)

Same as LS Modified Subproblem(s).

MC Cut Generation

For optimality cuts:

$$c1_{oc,s} = \sum_j D_j * v'_{js} \tag{4.1}$$

$$c2_{oc,i,s} = -C_i * u'_{is} \tag{4.2}$$

For feasibility cuts:

$$c1_{fc} = \sum_j \sum_s D_j * v'_{js} \tag{4.3}$$

$$c2_{fc,i} = \sum_s -C_i * u'_{is} \tag{4.4}$$

MC Master Problem

- $z1_s$, free variables that enable specific cuts for each scenario

min

$$z_{MP} = \sum_i F_i * z_i + \sum_s z1_s \tag{4.5}$$

subject to

$$z1_s \geq c1_{oc,s} + \sum_i c2_{oc,i,s} * z_i \quad \forall\, oc, s \tag{4.6}$$

$$c1_{fc,s} + \sum_i c2_{fc,i,s} * z_i \leq 0 \quad \forall\, fc, s \tag{4.7}$$

$$z_i \in \{0,1\} \quad \forall\, i \tag{4.8}$$

$$z1_s\ URS \quad \forall\, s \tag{4.9}$$

$$z_{MP}\ URS \tag{4.10}$$

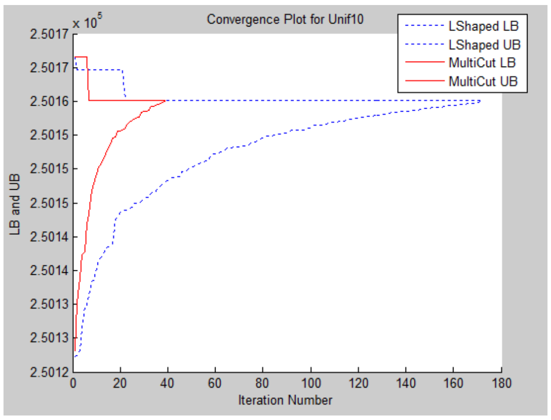
Algorithm 4.3 Multi-Cut Algorithm

Input: $T_{ijs}, D_j, C_i, F_i, p_s, \epsilon$, Time limit

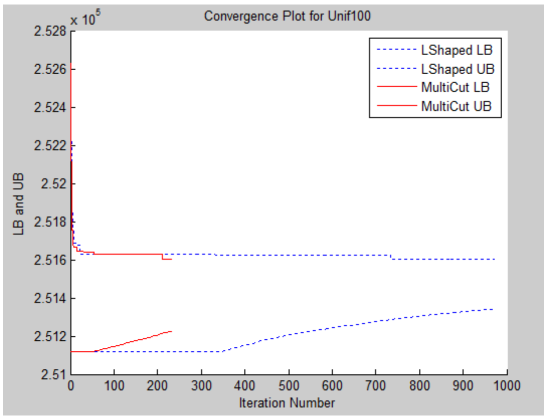
Output: Gap, best solution generated

```
 1: $iter \leftarrow 0, UB \leftarrow +INF, LB \leftarrow -INF$
 2: while $UB - LB > \epsilon$ do
 3:     for $s = 1\ to\ |S|$ do
 4:         $iter \leftarrow iter + 1$
 5:         solve (3.1) - (3.5)
 6:     end for
 7:     if $\sum_s z_{SP_S} < INF$ then
 8:         if $\sum_i z_i' * F_i + \sum_s z_{SP_S} < UB$ then
 9:             $UB \leftarrow \sum_i z_i' * F_i + \sum_s z_{SP_S}$
10:             $Best \leftarrow Z$
11:         end if
12:         generate cut from (4.1) - (4.2)
13:     else
14:         for $s = 1\ to\ |S|$ do
15:             solve (3.6) - (3.10)
16:         end for
17:         generate cut from (4.3) - (4.4)
18:     end if
19:     solve (4.5) - (4.10)
20:     $LB \leftarrow z_{MP}$
21: end while
```

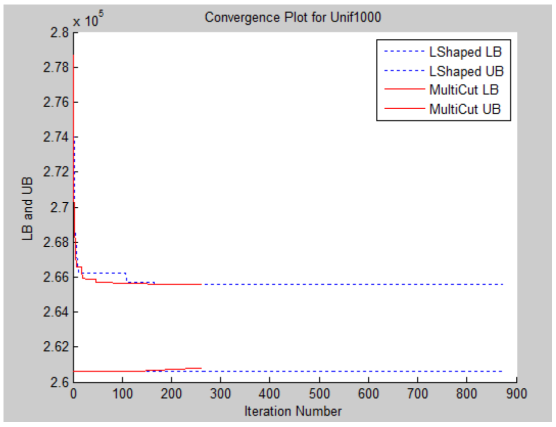
By adding multiple cuts, we expect MP to be much more tightly constrained, and convergence to occur in a smaller number of iterations. Figure 4.8 illustrates the convergence plots of the Multi-cut algorithm for the same three instances. To be able to compare the Multi-cut algorithm with the L-Shaped method, we plot their results together:



(a) Convergence of MC with small ins.



(b) Convergence of MC with medium ins.



(c) Convergence of MC with large ins.

Figure 4.8: Convergence of MC

For the small instance, the number of iterations was reduced significantly compared to the L-Shaped method. This is the result of constraining MP much stronger by adding |S| cuts at each iteration. Thanks to this reduction, the CPU time is also reduced to 416 seconds (although still higher than the solution time of the Benders algorithm). For small instances, it seems the Benders algorithm gives better results.

For medium and large instances, adding many cuts at each iteration results in MP becoming very hard to solve after a number of iterations. Since these instances are expected to require many iterations for convergence, we can predict that the size of MP will increase with time, thus iterations will take more and more time. Although we can observe a smaller number of iterations for this reason, the final LB found by the Multi-Cut algorithm is higher than the one found by the L-Shaped algorithm.

4.2.2 Scenario-Group-Cut

A possible solution for creating too many cuts may be finding a balance between the number of cuts (complexity of MP) and aggregate strength of cuts. Adulyasak et al. (2015a) propose such a method. Instead of generating cuts for each scenario, they suggest creating groups of scenarios and generating cuts for each scenario group. This way, the number of cuts can be significantly reduced, thus making it possible to observe more iterations with large instances. Naturally, the speed of convergence will be reduced, but this is expected to be compensated by the higher number of iterations. With more iterations, we can generate cuts based on a wider variety of levels for complicating variables and creating cuts on different regions of the solution space.

This method is especially suitable in our problem, since we already generated the scenarios in groups. We also expect that the real life travel time data can be easily grouped in a similar way to how we constructed our scenarios. There would be three main groups of high, medium and low traffic congestion levels. In addition, disruptions to traffic flow would have deteriorating effect on the local level, especially for some of the high and medium congestion days. The resulting five groups could be further divided into two, by taking into account the similarities or dissimilarities between days.

In this subsection, we perform an experiment with the Scenario Group Cut (SGC) Algorithm, so that its effect on the solution performance can be observed.

We can readily define scenario groups due to the way scenarios are constructed, as explained in Chapter 3. The following are the natural groups that can be used:

- Average days
  - Group 1 (No local traffic)
  - Group 2 (With local traffic)
- Congested days
  - Group 3 (No local traffic)
  - Group 4 (With local traffic)
- Relaxed days
  - Group 5

There are two ways for constructing scenario groups. One way is to create representative scenario groups by sampling from the original groups we constructed.

To do the sampling, we simply take one scenario from each original group. This approach actually maximizes dissimilarity within groups. Since in each group, we have a representative sample of all the scenarios, the scenario groups resemble each other.

The second way is to create natural scenario groups, simply taking the original groups as a whole. With this approach, we maximize similarity within each scenario group. Therefore, we obtain dissimilar groups, in contrast to the first grouping approach.

It is clear that the scenario grouping approach that generates stronger cuts would lead to better solution performance. To be able to compare them, we conducted some preliminary experiments. The experiments showed that the second approach is significantly more successful than the first one.

There is a simple reason for this result. When the scenario groups are similar, as is the case with the first approach, the cuts generated from these groups are also similar. Therefore, the combination of the cuts generated in one iteration is weaker. Since cuts are similar, their effects overlap and the result is close to the single cut case.

On the other hand, when the scenario groups are dissimilar, the cuts they generate are also dissimilar. Therefore, the effect of cuts do not overlap and the combined strength of all cuts is higher.

We use the natural scenario groups. For this purpose, we construct 10 scenarios for every instance, as this number is compatible with the number of scenarios in the original groups. Preliminary experiments show that decreasing or increasing the number of groups deteriorates solution performance.

Below we explain the modifications on the algorithm in order to generate scenario group cuts:

SGC Subproblem (s)

Same as LS Subproblem(s).

SGC Modified Subproblem (s)

Same as LS Modified Subproblem(s).

SGC Cut Generation

Same as MC Cut Generation.

SGC Master Problem

- G_{sg}, set of scenarios included in scenario group sg

min

$$z_{MP} = \sum_i F_i * z_i + \sum_{sg} z1_{sg} \tag{5.1}$$

subject to

$$z1_{sg} \geq \sum_{s\in G_{sg}} c1_{oc,s} + \sum_{s\in G_{sg}} \sum_i c2_{fc,i,s} * z_i \quad \forall\, oc, sg \tag{5.2}$$

$$\sum_{s\in G_{sg}} c1_{fc,s} + \sum_{s\in G_{sg}} \sum_i c2_{fc,i,s} * z_i \leq 0 \quad \forall\, fc, sg \tag{5.3}$$

$$z_i \in \{0,1\} \quad \forall\, i \tag{5.4}$$

$$z1_{sg}\ URS \quad \forall\, s \tag{5.5}$$

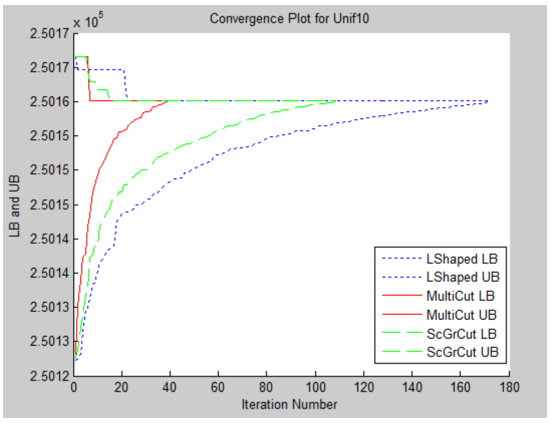
Algorithm 4.4 Scenario-Group Cut Algorithm

Input: $T_{ijs}, D_j, C_i, F_i, p_s, \epsilon$, Time limit

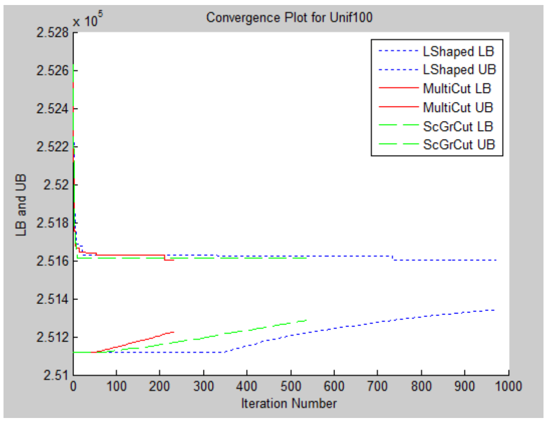
Output: Gap, best solution generated

```
 1: $iter \leftarrow 0, UB \leftarrow +INF, LB \leftarrow -INF$
 2: while $UB - LB > \epsilon$ do
 3:     for $s = 1\ to\ |S|$ do
 4:         $iter \leftarrow iter + 1$
 5:         solve (3.1) - (3.5)
 6:     end for
 7:     if $\sum_s z_{SP_s} < INF$ then
 8:         if $\sum_i z_i' * F_i + \sum_s z_{SP_s} < UB$ then
 9:             $UB \leftarrow \sum_i z_i' * F_i + \sum_s z_{SP_s}$
10:             $Best \leftarrow Z$
11:         end if
12:         generate cut from (4.1) - (4.2)
13:     else
14:         for $s = 1\ to\ |S|$ do
15:             solve (3.6) - (3.10)
16:         end for
17:         generate cut from (4.3) - (4.4)
18:     end if
19:     solve (5.1) - (5.5)
20:     $LB \leftarrow z_{MP}$
21: end while
```

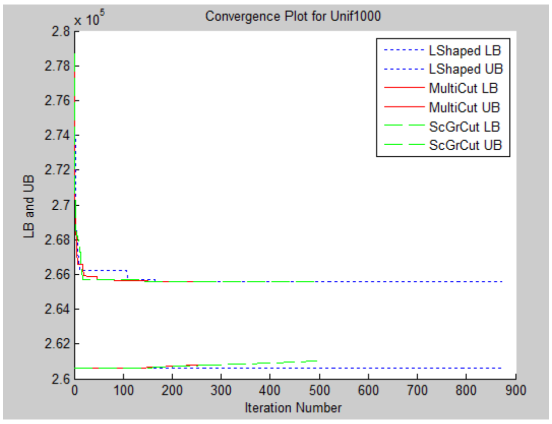
With this method, we reduce the number of cuts, at the expense of faster convergence. Figure 4.9 shows the convergence plots of the Scenario Group Cut algorithm for the same instances. To be able to observe the performance of the algorithm, we plot the results together with the Benders Algorithm and Multi-Cut Algorithm:



(a) Convergence of SGC with small ins.



(b) Convergence of SGC with medium ins.



(c) Convergence of MC with large ins.

Figure 4.9: Convergence of SGC

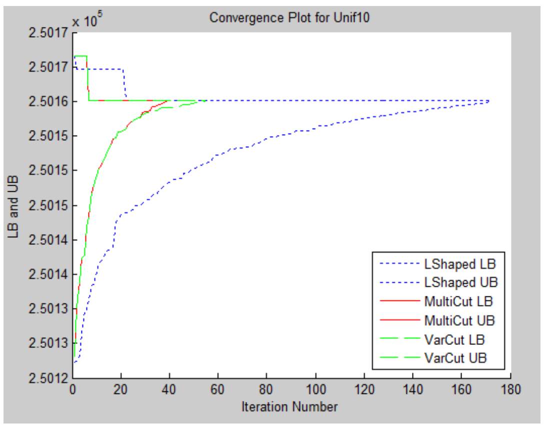
The results are in parallel with our expectations. In the small instance, we can easily observe that the new algorithm has slower convergence than Multi-Cut Algorithm, but faster convergence than L-Shaped Algorithm, which creates a single cut at each iteration. The solution time of 1100 seconds is also between these algorithms.

For the medium and large instances, the situation is similar. But especially for the large instance, we observe a significant improvement both in terms of the number of iterations and final LB value. When a good balance can be found between the number of cuts (and the resulting increase in MP size) and the strength of added constraint set, we achieve better bounds in the same amount of time.

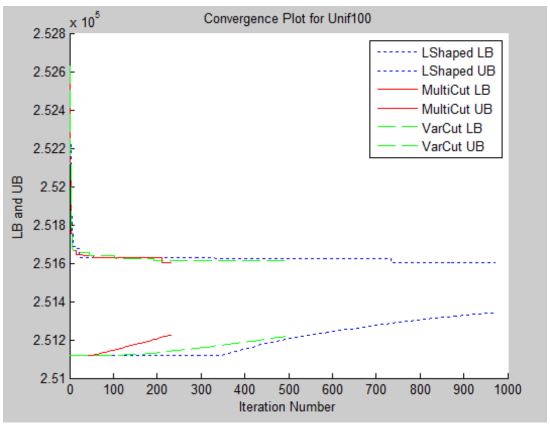
4.2.3 Variable Number of Cuts

Another way to hybridize the algorithms is to start by following the multi-cut procedure for some number of iterations and then continue by adding single/scenario-group cuts. This method avoids adding too many cuts to MP, while retaining the strength of the multi-cut algorithm for some iterations.

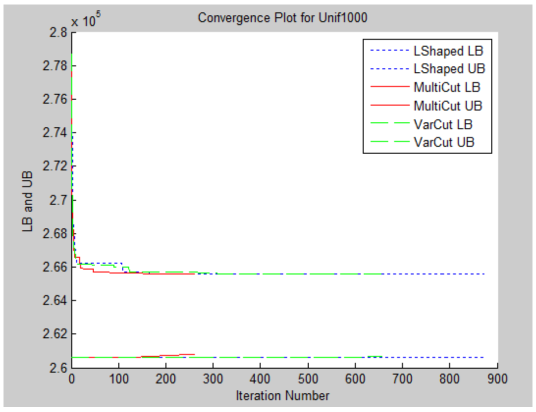
Below, we make the comparison on the same instances. For simplicity, we provide the convergence plot of the Variable Cut (VC) algorithm along with the plots of L-Shaped and Multi-Cut algorithms in Figure 4.10. In these experiments, we generate multiple cuts for the first 30 iterations and continue with single cuts afterwards.



(a) Convergence of VC with small ins.



(b) Convergence of VC with medium ins.



(c) Convergence of VC with large ins.

Figure 4.10: Convergence of VC

For the small instance, plots look like what we expected. LB of the Variable Cut algorithm increases as quickly as LB of the Multi-Cut algorithm for the first 30 iterations. Then, only a single cut is added at each iteration and the plots proceed almost parallel to LB of the L-Shaped algorithm. Time until convergence is 846 seconds for the Variable Cut algorithm.

For the medium and large instances, we see that the variable cut idea did not work. Since the number of iterations for achieving a meaningful increase in LB is very high, generating multiple cuts for 30 iterations does not make much difference from the ordinary L-Shaped algorithm. Still it produces slightly better results than the L-Shaped algorithm and worse results than Multi-Cut algorithm.

If we create multiple cuts periodically, every 10 iterations, instead of for the first 30 iterations (VC-2), the resulting convergence plot for the small instances greatly resembles to the one of L-Shaped algorithm (See Figure 4.11):

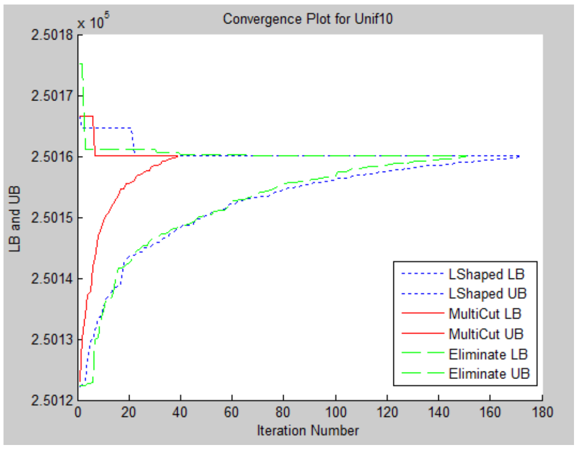


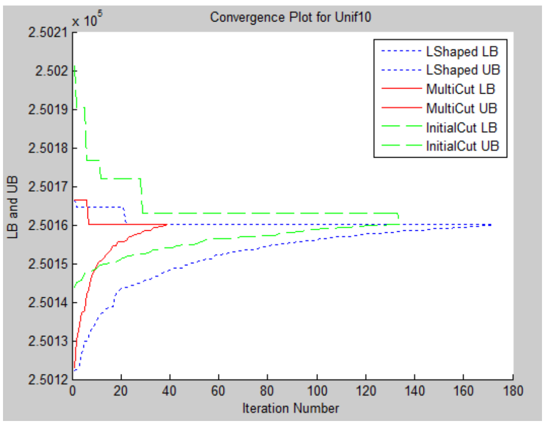
Figure 4.11: Convergence of VC-2 with small ins.

Note that there are small jumps every 10 iterations, but this is not sufficient to make a significant difference on the speed of convergence. For larger instances the results are similar.

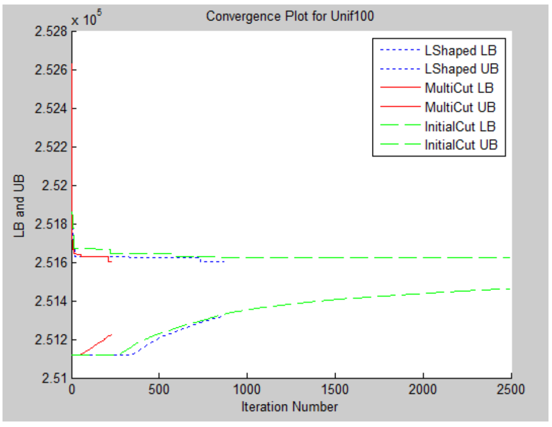
4.2.4 Adding Initial Cuts

A similar approach can be used to generate many cuts at the beginning of the algorithm but proceed by adding single/scenario-group cuts afterwards. We know that MP does not select optimal locations (at least for the first iterations) and the cuts are generated based on bad location decisions. Therefore, we may generate cuts based on arbitrarily selected locations, too.

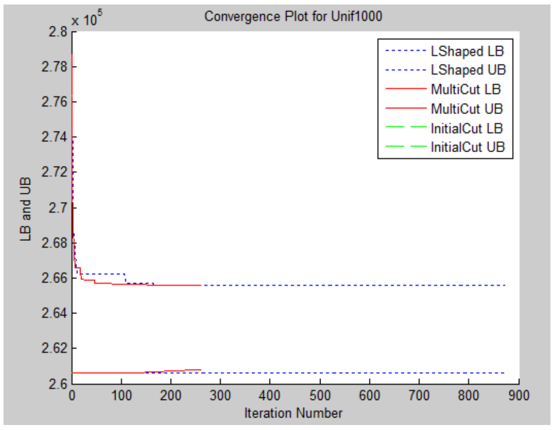
To select the locations, we solve each scenario separately in a Wait-and-See setting and generate cuts based on these results. In this way, we avoid solving MP for |S| many times, while generating better cuts than the ones produced by the algorithm on its own. In Figure 4.12, we give the convergence plots of the algorithm that adds initial cuts (IC).



(a) Convergence of IC with small ins.



(b) Convergence of IC with medium ins.



(c) Convergence of IC with large ins.

Figure 4.12: Convergence of IC

For the small instance, the effect of adding initial cuts is clear from the figure. The lower bound started the iterations from a significantly larger value. However, the collective strength of the cuts we add through the iterations seems to be lower, since the slope of the LB curve is lower than the LShaped method. Adding multiple cuts produces better results.

For the medium and large instances, we observe that the effect of adding initial cuts is much smaller. With the large instance, adding the initial cuts takes so much time that there is little time left for making the actual iterations.

4.2.5 Partial Decomposition

Despite having implemented the Multi-Cut and Scenario-Group Cut algorithms, we still cannot achieve fast convergence. One of the reasons for slow convergence of our Benders Decomposition variants is that MP does not maintain any information on the effect of its selection of complicating variables. It only takes into account the lower bound values generated by the SP for some combinations of the complicating variables. Therefore, MP has the tendency to select a combination that is not yet covered by the cuts, whether this selection improves the objective function of the real problem or not.

To maintain some of the information from the real problem in MP, Crainic et al. (2014) suggest the partial decomposition (PD) method. In this method, the main idea of fully separating the continuous and complicating variables into SP and MP is abandoned. Instead the authors propose keeping some of the continuous variables in MP, so that the problem is forced by the corresponding constraints to produce feasible objective function values while also to improve the objective function value more quickly.

Since the continuous variables of our problem is suitable for decomposition with respect to scenarios (as in the L-Shaped method), we can easily select some of the scenarios to represent the continuous part of the problem and insert them into MP. Naturally, dealing with large instances, it would be a good idea to only insert a small number of constraints into MP, in order to avoid increasing the size of it too much. Also, since we use this procedure to help MP make a more informed selection on the complicating variables, we select the scenarios in a way that they collectively carry as much information as possible. Therefore, the dissimilarity of scenarios is desirable. We take one scenario from each scenario-group explained in the Section 6.2, to construct a representative group of scenarios.

We add scenario-group cuts at each iteration. Since MP is already large, we try not to further increase its size through adding a lot of cuts. When we use scenario-group cuts, we maintain a good amount of information generated from subproblems without suffering the drawbacks of the multi-cut approach.

PD Subproblem (s)

Same as LS Subproblem(s). Some subproblems are not used.

PD Modified Subproblem (s)

Same as LS Modified Subproblem(s). Some modified subproblems are not used.

PD Cut Generation

Same as MC Cut Generation.

PD Master Problem

- G_{sg}, set of scenarios included in a certain scenario group
- UG, union of all scenario groups

min

$$z_{MP} = \sum_i F_i * z_i + \sum_{sg} z1_{sg} + \sum_{s \notin \mathrm{UG}} \sum_i \sum_j x_{ijs} * T_{ijs} * P_{ijs} \quad (6.1)$$

subject to

$$z1_{sg} \geq \sum_{s \in G_{sg}} c1_{oc,s} + \sum_{s \in G_{sg}} \sum_i c2_{oc,i,s} * z_i \quad \forall\, oc, sg \quad (6.2)$$

$$\sum_{s \in G_{sg}} c1_{fc,s} + \sum_{s \in G_{sg}} \sum_i c2_{fc,i,s} * z_i \leq 0 \quad \forall\, fc, sg \quad (6.3)$$

$$\sum_j x_{ijs} \leq z_i * C_i \quad \forall\, i, s \notin \mathrm{UG} \quad (6.4)$$

$$\sum_i x_{ijs} \leq D_j \quad \forall\, j, s \notin \mathrm{UG} \quad (6.5)$$

$$x_{ijs} \in \{0,1\} \quad \forall\, ijs \quad (6.6)$$

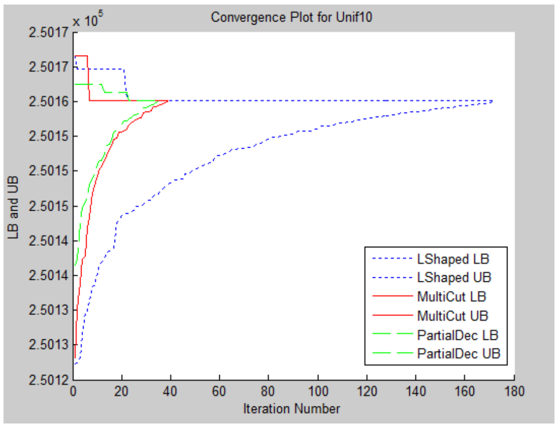
$$z_i \in \{0,1\} \quad \forall\, i \quad (6.7)$$

$$z1_{sg}\ URS \quad \forall\, s \quad (6.8)$$

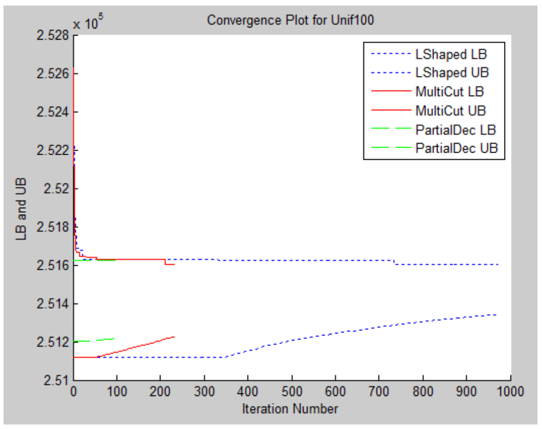
Algorithm 4.5 Partial Decomposition Algorithm

```
Input: $T_{ijs}, D_j, C_i, F_i, p_s, \epsilon$, Time limit
Output: Gap, best solution generated
 1: $iter \leftarrow 0, UB \leftarrow +INF, LB \leftarrow -INF$
 2: while $UB - LB > \epsilon$ do
 3:     for $s = 1\ to\ |S|$ do
 4:         $iter \leftarrow iter + 1$
 5:         solve (3.1) - (3.5)
 6:     end for
 7:     if $\sum_s z_{SP_S} < INF$ then
 8:         if $\sum_i z_i' * F_i + \sum_s z_{SP_S} < UB$ then
 9:             $UB \leftarrow \sum_i z_i' * F_i + \sum_s z_{SP_S}$
10:             $Best \leftarrow Z$
11:         end if
12:         generate cut from (4.1) - (4.2)
13:     else
14:         for $s = 1\ to\ |S|$ do
15:             solve (3.6) - (3.10)
16:         end for
17:         generate cut from (4.3) - (4.4)
18:     end if
19:     solve (6.1) - (6.8)
20:     $LB \leftarrow z_{MP}$
21: end while
```

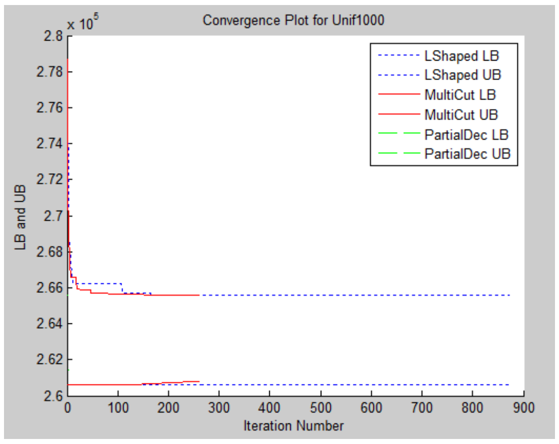
The convergence plot of the Partial Decomposition algorithm along with the L-Shaped and Multi-Cut algorithms are given in Figure 4.13.



(a) Convergence of PD with small ins.



(b) Convergence of PD with medium ins.



(c) Convergence of PD with large ins.

Figure 4.13: Convergence of PD

There are clear differences between the algorithms' performances. The most striking difference is that both for small and medium instances, UB very quickly reaches optimum (we already know the optimal solution). In addition, LB also increases sharply and reaches optimum more rapidly than any other algorithm we implemented for the small instance. However, this is not the case for the medium instance. In fact, only 80 iterations could be completed in a time limit of 4 hours. This is due to the increased size of MP, which takes a lot more time to be solved, due to the additional continuous variables. An important observation is that the partial decomposition idea does not work for our problem, if we attempt to deal with large instances.

4.2.6 Eliminating Dominated Cuts

A way to avoid increasing the size of MP due to the addition of multiple cuts may be to eliminate the cuts that are dominated by others. We implemented such an elimination procedure into our algorithm, In order to see how many cuts can be eliminated and what effect the procedure would have on the results. To explain cut dominance, we give the following generic example, given in Magnanti and Wong (1981):

When the problem can be formulated as:

$$Minimize_{y \in Y, z \in R} \{z: z \geq f(u) + yg(u) \; \forall \; u \in U\}$$

where R is the set of real numbers, and we have two cuts:

$$z \geq f(u^1) \; + \; yg(u^1) \qquad \text{(a)}$$

$$z \geq f(u) \; + \; yg(u) \qquad \text{(b)}$$

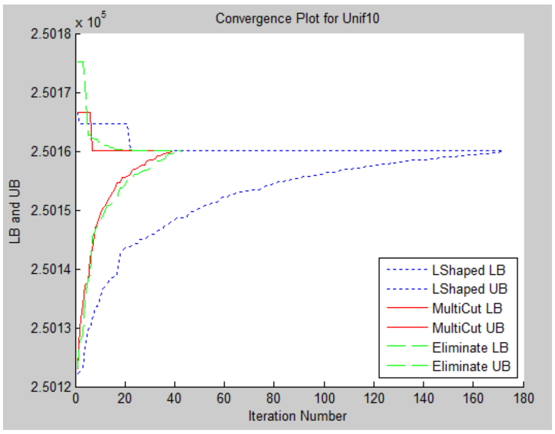
We say that (a) dominates (b) if the following condition is satisfied with strict inequality for at least one point $y \in Y$:

$$f(u^1) + \; yg(u^1) \geq f(u) + \; yg(u) \quad \forall \; y \in Y$$

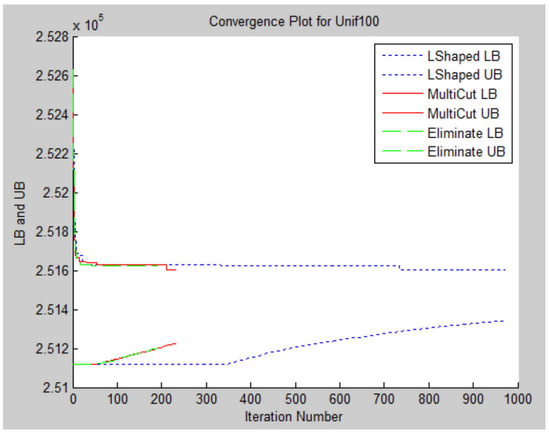
If no cut dominates (a), we call this a Pareto-optimal cut.

To determine if a cut dominates another, we have to know whether the inequality is valid for all elements of the set Y, which has tens of thousands of elements in our case. At this point, we need to take into account the tradeoff between the computational burden of determining dominance relations and the benefit of reducing the number of constraints. Based on our preliminary experiments, we choose twenty location decision combinations on which to decide the validity of the inequality. We risk losing some non-dominated cuts since we do not consider all the combinations.

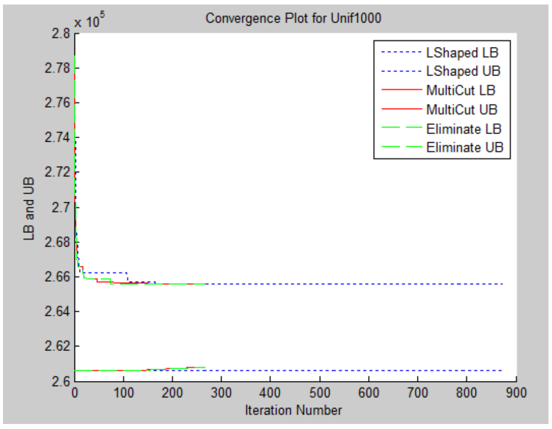
When we implemented the cut elimination (CE) procedure, we obtained the following results. We plot the results together with the L-Shaped and Multi-Cut algorithms for convenience in Figure 4.14:



(a) Convergence of IC with small ins.



(b) Convergence of IC with medium ins.



(c) Convergence of CE with large ins.

Figure 4.14: Convergence of CE

For both the small and large instances, we observe that the cut-elimination procedure did not improve or deteriorate the algorithm performance. The statistics show that 414 cuts were eliminated out of 4600 cuts generated for the small instance, and 2025 cuts were eliminated out of 27100 cuts generated for the large instance. On the other hand, the number of iterations for the small instance increased from 43 to 46 and from 266 to 271 respectively. Solution time slightly increased for the small instance, and did not change for the larger instance. Overall, we can conclude that the elimination procedure does not produce significant improvement on these problem instances.

4.3 Other Methods for Accelerating Benders Decomposition

Other than the ones we considered, there are many modifications proposed in the literature attempting to accelerate the Benders Decomposition algorithm. Some of these methods achieve great improvement over the original algorithm. However, these modifications are usually problem-specific and fail to make significant difference for other problems. In this section, we list these modifications and discuss the reasons why we did not consider them in our study.

Pareto-optimal Cuts

A frequently used method in the Benders Decomposition literature is the addition of Pareto-Optimal Cuts to MP. This method was proposed by Magnanti and Wong (1981) and has been used due to its ease of implementation. The method exploits the idea that several different cuts can be generated based on a single location decision, if the resulting SP is degenerate. Then, the problem becomes finding the tightest (non-dominated) cut for addition to MP. This can be done using another SP for detecting the tightest possible cut with the help of a core point in the feasible region of the complicating variables. Usually, application of this method obtains slightly better results than the original algorithm.

Wentges (1996), Adulyasak et al. (2015a), Sherali and Lunday (2013) and Papadakos (2008) use this method and provide insights on how the algorithm can be further developed. However, generating pareto-optimal cuts do not guarantee an overall improvement on the solution performance. As pointed out by Oliveira et al. (2014), there are several implementation issues. For instance, a core point must be selected according to the current convex hull of the complicating variables and an additional SP is required for the algorithm to find a pareto-optimal cut. These two requirements are major problems for the algorithmic efficiency, because finding a core point may be difficult and may require an additional separation problem to be solved at each iteration, plus solving the SP dedicated to pareto-optimal cuts may be difficult in itself. Thus, the solution time for each iteration increases due to the extensions and adding slightly better cuts does not provide significant benefit.

Є-optimal solution

Geoffrion and Graves (1974) suggest not solving MP to optimality, but solving it until a specified gap is obtained (when UB-LB<Є). This method does not prevent the proof of optimality, because the decisions of MP are only required for the generation of cuts. This approach accelerates the completion of iterations when the master problem is hard to solve. Fischetti et al. (2010) also uses this method.

For our case, this enhancement does not make much difference. This is because our Master Problem is small. It includes only 20 binary variables. The complexity of MP increases only when a large number of cuts are added to the problem or when the original problem is decomposed partially, leaving some continuous variables in MP. Our experiments showed that even in those cases, the effect of this enhancement is insignificant. For this reason, we do not implement it in our algorithms.

Local branching

Fischetti and Lodi (2003) propose the local branching method. In this method, the feasible region of a large optimization problem is divided into smaller regions and the optimal solutions in each region are found separately by using significantly smaller models, thus in a shorter time. Then, these results are used with a branching strategy to solve the original problem. Rei et al. (2009) uses this method.

The main advantage of this enhancement is to obtain faster solution of MP especially when there is a meaningful way of dividing the feasible region. We do not consider its implementation, because we can divide the feasible region only arbitrarily. Also, the time-consuming part of our algorithm is solving SPs rather than solving MP.

Cross-decomposition

Lastly, we have the cross-decomposition method proposed by Roy (1986) and used by Uster and Agrahari (2011). This method aims to accelerate the Benders Decomposition algorithm by achieving a faster solution of MP. Since MP usually includes only binary variables, there are several decomposition methods that may provide a quicker solution than the classical Branch and Bound; for instance, Lagrangean Relaxation is a good example to such methods.

Advantages of this enhancement and reasons for us not implementing it are the same as local branching.

Other methods we did not consider are as follows: maximum feasible subsystem cut generation strategy (Saharidis and Ierapetritou, 2010a), covering cut bundle generation (Saharidis et al., 2010b), valid inequalities in initialization (Saharidis et al., 2011), maximum density cut generation (Saharidis and Ierapetritou, 2013), minimal infeasible subsystems (Fischetti et al., 2008).

4.4 Branch and Cut Based on Benders Decomposition

The cut generation procedure used in Benders Decomposition is also frequently used in combination with the Branch and Bound method. One of the first studies to use this framework is McDaniel and Devine (1977). A contemporary example can be seen in Fortz and Poss (2009). The main idea is to make use of the Benders cuts in order to more efficiently search the classical Branch and Bound tree. Benders cuts reduce the size of the feasible region at each node, resulting in larger lower bounds, which increases the number of fathomed branches. Being able to fathom more branches, algorithm is expected to search a larger portion of the tree compared to the classical Branch and Bound algorithm.

We observed that most of the studies considering the Branch and Benders Cut (B&BC) procedure consider problems requiring more feasibility cuts than optimality cuts for convergence. Such behavior increases the benefit of using B&BC, since a lot of nodes can be fathomed due to infeasibility, which allows for a significantly faster search. Although a large majority of our cuts are of optimality type, we still consider this solution method as a possibility.

There are many alternative strategies that can be implemented in a B&BC framework. For instance, using depth-first, breadth-first, and best-first strategies produce quite different outcomes. The number of cuts to be added at each iteration is also an important choice. These alternatives will be explored in this section.

Before explaining the distinctions between different B&BC settings, we give the general B&BC algorithm. Figure 4.15 illustrates the main steps of the algorithm.

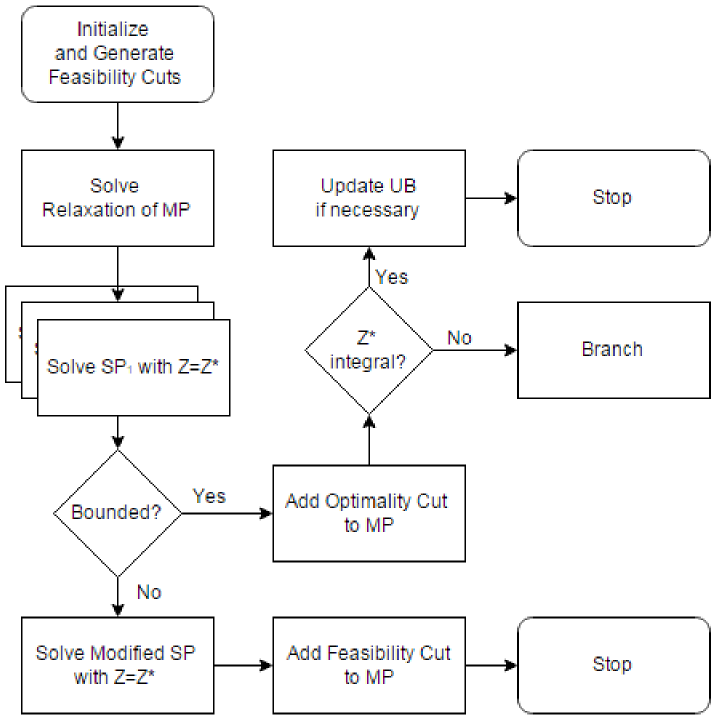


Figure 4.15: Branch and Cut with Benders Decomposition Algorithm (B&C-BD)

4.4.1 No Decomposition

Not making any decomposition in the original problem results in a similar method to the classical Branch and Bound. With this approach, the whole problem, including decision variables from both stages, is solved at each node of the tree. For our case, the only advantage of this approach over the Branch and Bound method may be the use of optimality cuts with the hope of fathoming more nodes which would accelerate the search.

Our preliminary experiments showed that this acceleration is not large enough to rationalize using this method. Especially in medium and large instances, solving the whole problem at each node takes too much time. Even when the binary constraints are relaxed, the sheer number of continuous variables require a long solution time at each node. We abandoned this method, since we have obtained more promising results with the L-Shaped method.

4.4.2 Full Decomposition

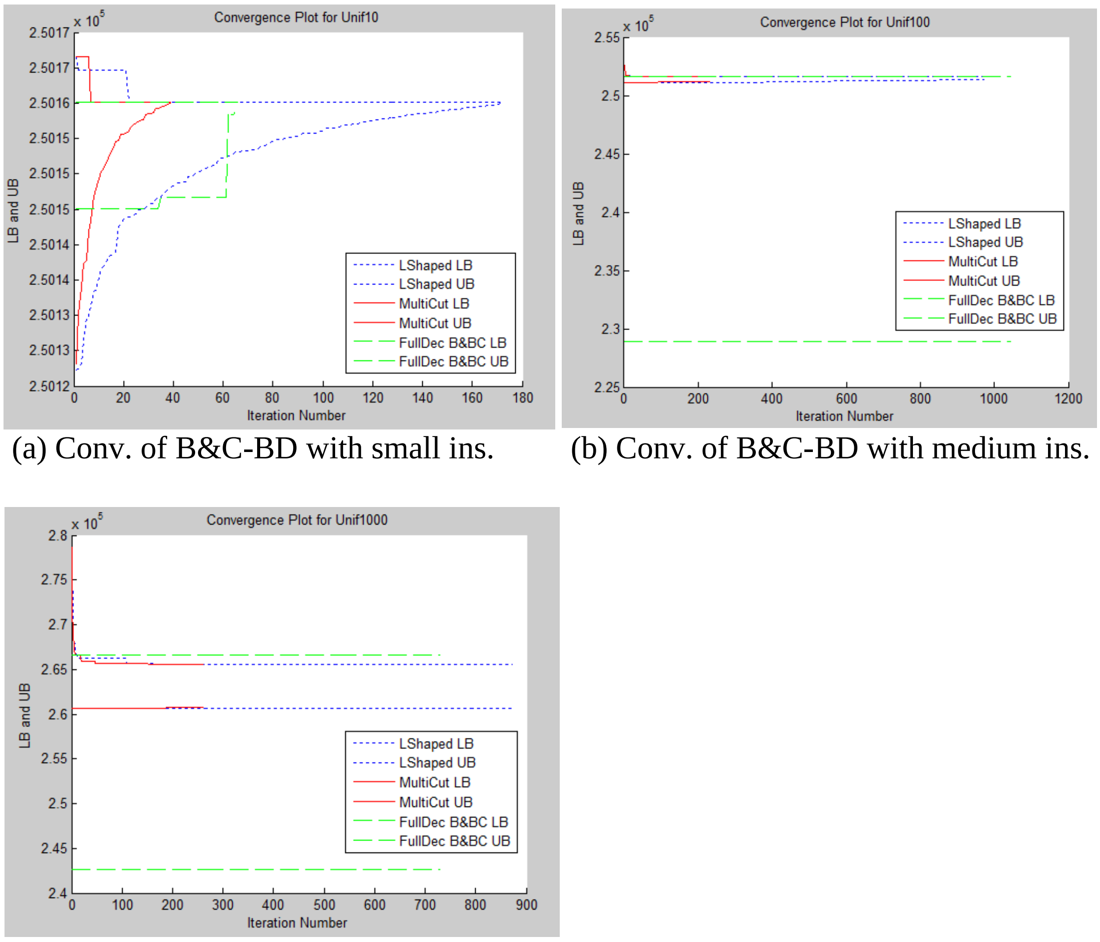
In this setting, we fully decompose the first and second stage decision variables. We solve MP and SPs at each node. Similarly to the L-Shaped method, we combine the results of MP and SPs to make the necessary computations. For instance, as the objective function value of MP does not fully include the transportation cost associated with the given location decisions (it only includes the cost as much as forced by the optimality cuts, which may not be strong enough), we obtain the transportation cost term from the SPs. The objective function value obtained in this way becomes lower bound for the node. If the location decisions made in MP are integral, then the solution of the current node is a candidate solution and its objective function value becomes the upper bound if it is the best solution found so far.

We know from the preliminary experiments that a few feasibility cuts are required when full decomposition is employed. To simplify the tree, we generate these feasibility cuts with an initial cut generation procedure. We apply the L-Shaped method for only 10 iterations and generate the necessary cuts. We still maintain the modified SPs within the B&BC algorithm. After this initial procedure no additional feasibility cuts are needed. The algorithm only generates optimality cuts in the B&BC algorithm.

Parallel to the L-Shaped method, we have three main alternative approaches for cut generation: single-cut, multiple-cuts and scenario-group cuts. We do not consider the other enhancements to the L-Shaped method, since the preliminary experiments showed that they are inferior to the selected approaches.

The advantages and disadvantages of using these approaches are similar to the ones explained earlier in section 6.2. However, there is an important difference between B&BC and L-Shaped method. Master problems of B&BC include continuous location decisions. This leads to much lower computational complexity for the master problems which neutralizes the main drawback of adding multiple cuts. When MP was a binary integer problem in L-Shaped method, adding cuts increased the complexity of MP so much that the L-Shaped algorithm would slow down after too many cuts were added. This is not expected to be the case with B&BC, since this time MP is a continuous problem.

Based on these results we expect that adding multiple cuts will perform significantly better than the others and preliminary experiments showed this is the case. Figure 4.16 shows the convergence behavior of the algorithm for this particular implementation. We plot the results of B&BC along with L-Shaped and MultiCut algorithms. Note that the iteration number for B&BC is equal to the number of nodes search in the tree.



(a) Conv. of B&C-BD with small ins. (b) Conv. of B&C-BD with medium ins.

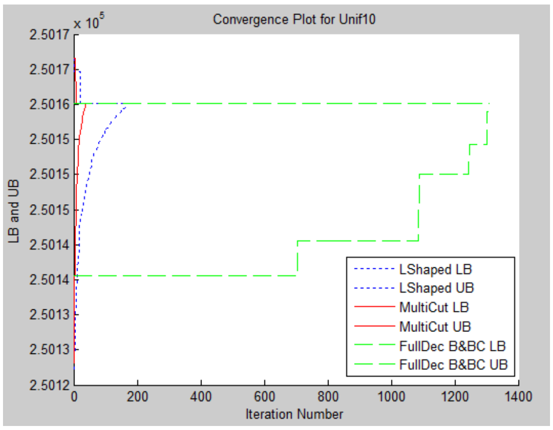
(c) Conv. of B&C-BD with large ins.

Figure 4.16: Convergence of B&C-BD

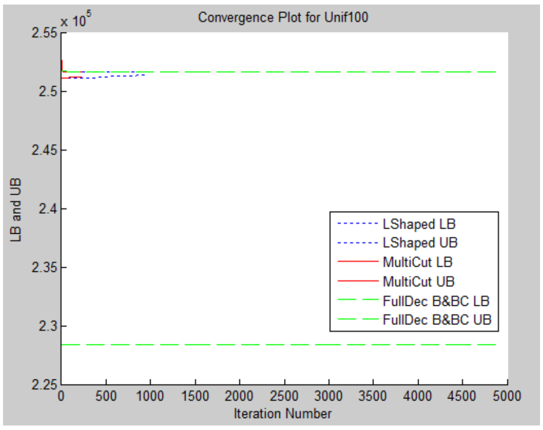
The most important difference in the results obtained by the B&BC algorithm is the large gaps left after the time limit is reached. This is due to the unexplored nodes in the branch and bound tree. Since the algorithm cannot search the tree fast enough to fathom these nodes, there are always unexplored nodes in the tree with very small lower bound values. Thus, the gap we obtain at the time limit is not as small as the ones obtained by the previous algorithms.

The algorithm spends a lot of time for solving SPs. To accelerate the search, we considered solving SPs at only integer nodes. Solving SPs allow us to observe the real

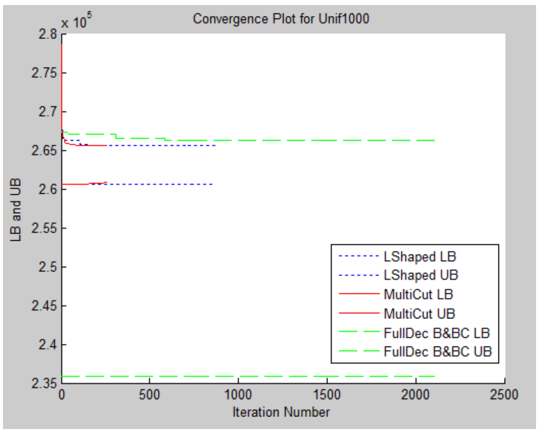
LB values at each node and we are able to fathom some nodes by comparing them with the UB. If we sacrifice quick fathoming, it is possible to avoid solving SPs at non-integer nodes, since the objective function value found by MP is a valid LB, however, it is not tight. We experimented on this idea and the results are illustrated in Figure 4.15.



(a) Conv. of B&C-BD-2 with small ins.



(b) Conv. of B&C-BD-2 with medium ins.



(c) Conv. of B&C-BD-2 with large ins.

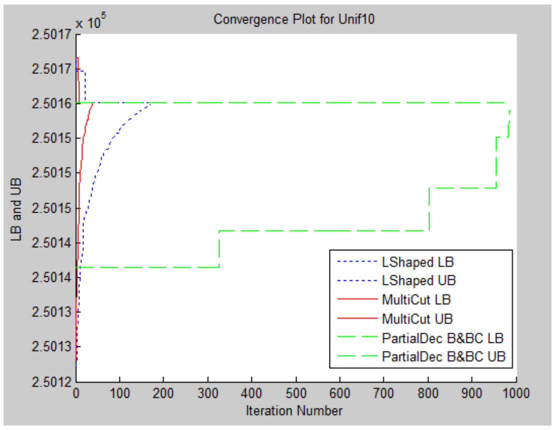
Figure 4.17: Convergence of B&C-BD-2

With this approach we are able to search a much larger part of the tree and we find better UBs most of the time. However, we still cannot decrease the gap at the time limit. In fact, the gap is larger this time, because LBs are equal to the objective function values of MP. In other words, we do not find the actual LBs for the nodes of the tree, but we use the bounds that correspond to the L-Shaped methods LBs for each node.

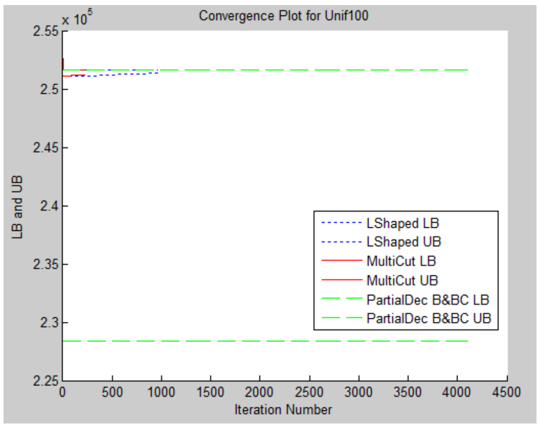
4.4.3 Partial Decomposition

Partial decomposition is the hybrid implementation between no decomposition and full decomposition. We consider partial decomposition due to its success in finding good solutions faster than the full decomposition method. The allocation decisions left in MP have more significant power than optimality cuts in directing MP to good solutions. On the other hand, it has the same drawback as no decomposition method. With large instances, the extra assignment decisions increase the complexity of MP significantly. Therefore, MP takes longer to be solved and the algorithm slows down. With the lower speed, only a smaller part of the tree can be searched.

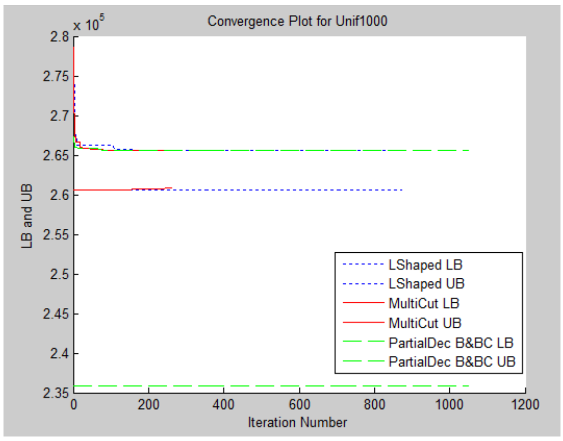
We provide the preliminary results for partial decomposition (B&C-PD), in Figure 4.16. This time we only consider the results obtained by solving SPs in fewer nodes.



(a) Conv. of B&C-PD with small ins.



(b) Conv. of B&C-PD with medium ins.



(c) Conv. of B&C-PD with large ins.

Figure 4.18: Convergence of B&C-PD

The results are very similar to the previous algorithm. We observe two differences. First, very good UBs are found in the early stages of the algorithm. These UBs are often equal to the best known integer solutions in our experiments. Also, for the small instance, the algorithm converges in a moderate time. Second difference is the significant decrease in the number of nodes that can be searched within time limit. For the medium and large instances, this number decreased because MP takes significantly longer due to the continuous variables it includes.

4.5 Comparison of Exact Solution Methods

In this subsection, we test and compare performances of the algorithms implemented. We mainly consider two performance measures, some other details are also given for assessing the differences between the algorithms:

- Percent gap between the best integer solution found and lower bound at the time of completion
- CPU time in seconds

We make the comparison on two levels. On the first level, we experiment with all methods on 3 instances (R_u_L_10, R_u_L_100 and R_u_L_1000). We use this level of computational results to screen out the methods that are obviously dominated by others in terms of solution quality. On the second level, we experiment with selected methods on 36 instances. Obviously the larger number of instances provide a more reliable base for comparing the selected methods.

4.5.1 First Level Comparison

Tables 4.1-4.3 summarize our findings for the small, medium and large instances respectively. We report the best integer/UB, lower bound, number of iterations (number of nodes checked), CPU time and Gap results. Gap is always calculated as the percent difference between the last UB and LBs found by the method.

$$Gap\ \% = \frac{UB-LB}{LB}*100$$

Table 4.1: Comparison of Exact Solution Methods with Small Instance

Results for R_u_L_10_1	**Best Integer**	**Lower Bound**	**Iter. (Nodes Checked)**	**CPU Time (sec)**	**GAP%**
Two-Stage St.	250160.1	250160.1	N/A	2.6	0.0
Benders Dec. Alg.	250160.1	250160.1	170	75.3	0.0
L-Shaped Alg.	250160.1	250160.1	175	1203	0.0
Multi-Cut Alg.	250160.1	250160.1	42	402.1	0.0
Sc. Gr. Cut Alg.	250160.1	250160.0	120	618.5	0.0
Variable Cut Alg.	250160.1	250160.1	58	640.4	0.0
Initial Cut Alg.	250160.1	250160.1	100+137	103.7	0.0
Partial Dec.	250160.1	250160.1	38	375.2	0.0
B&C-BD-2	250160.1	250160.1	46(1313)	1494.9	0.0
B&C-PD	250160.1	250160.1	38(993)	1295.4	0.0

For the small instance, we can see that all algorithms are able to find the optimal solution within time limit. As explained in the corresponding sections, Benders Algorithm appear to converge in considerably shorter time than others, since SP is small, and further decomposition does more harm than good. We see that the multi-cut algorithm sharply reduces the CPU time and number of iterations, and the scenario-group cut algorithm performs between the two, as expected. Variable cut algorithm performs badly despite the small number of iterations, but partial decomposition is able to further reduce the number of iterations.

Table 4.2: Comparison of Exact Solution Methods with Medium Instance

Results for R_u_L_100_1	**Best Integer**	**Lower Bound**	**Iter. (Nodes Checked)**	**CPU Time (sec)**	**GAP%**
Two-Stage St.	251602.1	251602.1	N/A	9706.6	0.0
Benders Dec. Alg.	251602.1	251378.1	1818	*	0.1
L-Shaped Alg.	251602.1	251409.4	977	*	0.1
Multi-Cut Alg.	251602.1	251220.2	225	*	0.2
Sc. Gr. Cut Alg.	251610.9	251305.6	599	*	0.1
Variable Cut Alg.	251610.9	251217.9	498	*	0.2
Initial Cut Alg.	251622.1	251462.4	100+2501	*	0.1
Partial Dec.	251622.3	251222.2	111	*	0.2
B&C-BD-2	251602.1	228391.8	817(4904)	*	10.2
B&C-PD	251610.9	228394.3	653(3125)	*	10.2

* Time Limit Exceeded

For the medium instance, none of the Benders algorithm variants reach convergence within the time limit. Although they produce small gaps, they do not perform in any way better than CPLEX. Although the algorithms do not converge within time limit, an important observation is that most of their UB values are equal to the optimal objective function value. This shows that all the algorithms find the optimal location decisions at some point and they update their UB to the optimal value with the corresponding dual prices. Actually, this often happens quite early through the iterations, for example at the 4th iteration of the partial decomposition procedure.

Table 4.3: Comparison of Exact Solution Methods with Large Instance

Results for R_u_L_1000_1	**Best Integer**	**Lower Bound**	**Iter. (Nodes Checked)**	**CPU Time (sec)**	**GAP%**
Two-Stage St.	1010631.3	239004.7	N/A	*	322.8
Benders Dec. Alg.	265938.4	260631.3	199	*	2.0
L-Shaped Alg.	265603.0	260631.3	877	*	1.9
Multi-Cut Alg.	265595.0	260861.2	292	*	1.8
Sc. Gr. Cut Alg.	265595.0	261182.8	612	*	1.7
Variable Cut Alg.	265605.7	260679.6	661	*	1.9
Initial Cut Alg.	276335.5	260631.3	100+6	*	6.0
Partial Dec.	265593.4	261407.5	6	*	1.6
B&C-BD-2	266185.4	235856.5	620(2116)	*	12.9
B&C-PD	265593.4	235869.9	491(1188)	*	12.6

* Time Limit Exceeded

The real power of the algorithms can be seen when the large instance is considered. While CPLEX cannot find any meaningful results within the time limit (it also failed to produce a good integer solution in 48 hours), algorithms are able to conduct many iterations, gradually finding better bounds. This time, the differences between algorithms appear to be more important.

Since the problem size is large, dealing with a large SP reduces the performance of Benders Decomposition algorithm considerably. L-Shaped algorithm does not suffer from such a large SP, but it lacks the cuts to produce good bounds, despite completing 877 iterations. On the other hand, multi-cut and scenario-group cut algorithms obtain the best solutions among all due to the addition of multiple cuts at each iteration. Multi-cut algorithm reaches the best UB luckily at the 265th iteration and scenario-group cut algorithm later at 341st iteration due to the lower number of cuts added each time.

Since the actual number of iterations needed for convergence is very high, we do not expect the variable-cut algorithm to be a lot different than the l-shaped algorithm; in fact, it gives slightly worse results, since the addition of multiple cuts at the beginning increases MP size and wastes time during the rest of the algorithm.

In the light of these observations, we select MC, SGC, PD and B&C-PD for making further experiments on the second level. The first three are able to consistently produce low gap values and their UBs are always optimal or equal to the best solution known. B&C-PD will be included in the experiments because it is also good at finding good UBs, despite producing very bad gap values.

4.5.2 Second Level Comparison

In addition to the methods selected in the previous section, we provide the results of TLAP as well.

Tables 4.4-4.6 summarize our findings for the small, medium and large instances respectively. Best solutions for each instance are given in bold.

Table 4.4: Comparison of Selected Exact Solution Methods with Small Instances

Results for Small Instances	TLAP			MC			SGC			PD			B&C-PD		
	Best Integer	CPU Time (sec)	GAP%	Best Integer	CPU Time (sec)	GAP%	Best Integer	CPU Time (sec)	GAP%	Best Integer	CPU Time (sec)	GAP%	Best Integer	CPU Time (sec)	GAP%
R_u_L_10_1	**250160.1**	2.6	0.0	**250160.1**	402.1	0.0	**250160.1**	618.5	0.0	**250160.1**	375.2	0.0	**250160.1**	1295.4	0.0
R_n_L_10_1	**250179.6**	7.2	0.0	**250179.6**	1973.3	0.0	**250179.6**	4543.0	0.0	**250179.6**	770.6	0.0	**250179.6**	3781.9	0.0
C_u_L_10_1	**250180.5**	2.9	0.0	**250180.5**	483.1	0.0	**250180.5**	789.2	0.0	**250180.5**	241.9	0.0	**250180.5**	998.6	0.0
C_n_L_10_1	**250167.2**	2.4	0.0	**250167.2**	160.2	0.0	**250167.2**	224.9	0.0	**250167.2**	105.2	0.0	**250167.2**	456.7	0.0
RC_u_L_10_1	**250180.7**	4.2	0.0	**250180.7**	492.8	0.0	**250180.7**	1248.2	0.0	**250180.7**	271.3	0.0	**250180.7**	1237.0	0.0
RC_n_L_10_1	**250155.7**	3.2	0.0	**250155.7**	267.4	0.0	**250155.7**	281.5	0.0	**250155.7**	174.5	0.0	**250155.7**	721.2	0.0
R_u_S_10_1	**730244.1**	4.4	0.0	**730244.1**	698.2	0.0	**730244.1**	1911.0	0.0	**730244.1**	494.5	0.0	**730244.1**	1927.9	0.0
R_n_S_10_1	**788669.3**	7.1	0.0	**788669.3**	3023.7	0.0	**788669.3**	9435.6	0.0	**788669.3**	1089.4	0.0	**788669.3**	6591.5	0.0
C_u_S_10_1	**791586.6**	3.0	0.0	**791586.5**	737.1	0.0	**791586.5**	1594.1	0.0	**791586.5**	386.4	0.0	**791586.5**	1313.1	0.0
C_n_S_10_1	**751561.4**	2.1	0.0	**751561.4**	320.2	0.0	**751561.4**	595.7	0.0	**751561.4**	150.5	0.0	**751561.4**	496.7	0.0
RC_u_S_10_1	**792066.5**	4.0	0.0	**792066.5**	714.0	0.0	**792066.5**	2297.8	0.0	**792066.5**	418.6	0.0	**792066.5**	1686.4	0.0
RC_n_S_10_1	**708776.3**	3.3	0.0	**708776.3**	385.0	0.0	**708776.3**	829.3	0.0	**708776.3**	268.8	0.0	**708776.3**	1023.6	0.0

Table 4.5: Comparison of Selected Exact Solution Methods with Medium Instances

Results for Medium Instances	TLAP			MC			SGC			PD			B&C-PD		
	Best Integer	CPU Time (sec)	GAP%	Best Integer	CPU Time (sec)	GAP%	Best Integer	CPU Time (sec)	GAP%	Best Integer	CPU Time (sec)	GAP%	Best Integer	CPU Time (sec)	GAP%
R_u_L_100_1	**251602.1**	9706.6	0.0	**251602.1**	*	0.2	251610.9	*	0.1	251622.3	*	0.2	251610.9	*	10.2
R_n_L_100_1	**251744.3**	7538.3	0.0	**251744.3**	*	0.1	251747.3	*	0.1	251747.7	*	0.1	**251744.3**	*	10.1
C_u_L_100_1	**251551.2**	6305.9	0.0	**251551.2**	*	0.0	**251551.2**	*	0.0	**251551.2**	*	0.0	**251551.2**	*	10.1
C_n_L_100_1	**251560.9**	5139.4	0.0	**251560.9**	*	0.1	**251560.9**	*	0.0	**251560.9**	*	0.1	**251560.9**	*	10.1
RC_u_L_100_1	**251455.0**	6361.9	0.0	251472.8	*	0.1	**251455.0**	*	0.1	251462.0	*	0.1	**251455.0**	*	10.1
RC_n_L_100_1	**251677.3**	5553.7	0.0	**251677.3**	*	0.1	**251677.3**	*	0.0	**251677.3**	*	0.0	**251677.3**	*	10.0
R_u_S_100_1	**730624.5**	1754.9	0.0	733263.9	*	19.3	738260.1	*	16.1	736679.9	*	19.2	733263.9	*	30.1
R_n_S_100_1	**773292.2**	1994.9	0.0	774182.3	*	14.5	**773292.2**	*	11.7	774318.1	*	12.3	**773292.2**	*	21.3
C_u_S_100_1	**715369.3**	436.7	0.0	**715369.3**	*	5.0	**715369.3**	*	3.5	**715369.3**	*	3.4	**715369.3**	*	15.8
C_n_S_100_1	**718274.1**	769.0	0.0	**718274.1**	*	8.0	**718274.1**	*	6.1	**718274.1**	*	6.6	**718274.1**	*	19.1
RC_u_S_100_1	**686507.6**	769.0	0.0	**686507.6**	*	12.4	**686507.6**	*	9.8	688607.4	*	12.5	**686507.6**	*	24.6
RC_n_S_100_1	**753200.8**	857.8	0.0	**753200.8**	*	5.9	**753200.8**	*	4.1	**753200.8**	*	4.1	**753200.8**	*	14.5

* Time limit exceeded.

Table 4.6: Comparison of Selected Exact Solution Methods with Large Instances

Results for Large Instances	TLAP			MC			SGC			PD			B&C-PD		
	Best Integer	CPU Time (sec)	GAP%	Best Integer	CPU Time (sec)	GAP%	Best Integer	CPU Time (sec)	GAP%	Best Integer	CPU Time (sec)	GAP%	Best Integer	CPU Time (sec)	GAP%
R_u_L_1000_1	1010631.3	*	322.8	265595.0	*	1.8	265595.0	*	1.7	**265593.4**	*	1.6	**265593.4**	*	12.6
R_n_L_1000_1	1014487.2	*	319.6	267886.6	*	1.2	**267840.4**	*	1.2	267886.9	*	1.0	**267840.4**	*	11.7
C_u_L_1000_1	1012673.4	*	321.0	**265947.9**	*	1.1	**265947.9**	*	1.0	**265947.9**	*	0.9	266002.4	*	11.8
C_n_L_1000_1	1013515.5	*	320.0	**267120.0**	*	1.3	**267120.0**	*	1.2	**267120.0**	*	1.0	**267120.0**	*	11.9
RC_u_L_1000_1	1012550.9	*	321.2	**266456.9**	*	1.4	**266456.9**	*	1.3	266499.1	*	1.2	**266456.9**	*	12.1
RC_n_L_1000_1	1011994.5	*	322.0	265975.3	*	1.5	265940.7	*	1.4	265940.7	*	1.2	**265920.5**	*	12.1
R_u_S_1000_1	1318937.5	*	107.6	718170.6	*	24.9	**717802.7**	*	22.8	**717802.7**	*	21.2	726146.0	*	33.3
R_n_S_1000_1	1434617.2	*	98.6	**785211.0**	*	14.0	**785211.0**	*	13.8	786605.8	*	10.6	**785211.0**	*	18.9
C_u_S_1000_1	1380202.2	*	102.1	**728436.7**	*	13.5	**728436.7**	*	11.8	**728436.7**	*	11.1	732680.1	*	20.9
C_n_S_1000_1	1405464.6	*	99.1	**763599.7**	*	15.7	**763599.7**	*	14.8	**763599.7**	*	12.1	**763599.7**	*	21.0
RC_u_S_1000_1	1376527.1	*	102.5	**743705.8**	*	17.9	**743705.8**	*	17.1	744972.8	*	14.3	750672.6	*	24.6
RC_n_S_1000_1	1359835.6	*	105.4	728554.3	*	18.6	**727614.1**	*	17.4	728220.1	*	14.9	**727614.1**	*	24.3

***** Time limit exceeded.

On Table 4.4, the most successful method is TLAP. Since the instances in this groups are small, the low computational complexity of the model does not necessitate a decomposition approach. MC, SGC, PD and B&C-PD are all able to find the optimal, but they require much longer solution time to converge. The average solution time is only 3.9 seconds for TLAP, while for other methods, this figure is 804.8, 2030.7, 395.6 and 1794.2 respectively. In terms of solution time, PD is the most efficient exact solution method with small instances. This result is due to the lower number of iterations needed until convergence. On average, only 36 iterations are enough for PD to converge. For MC and SGC, 49 and 171 are the average number of iterations, respectively. B&C-PD takes longer time until convergence, since it explores hundreds of nodes on the branch-and-bound tree and needs to solve MP at each node.

Table 4.5 shows a different picture. This time, none of the decomposition methods converge within the time limit of four hours for any of the instances. Still the UB values are often equal to the optimal objective function values. The average gap obtained by MC, SGC, PD and B&C-PD are 5.5%, 4.3%, 4.9% and 15.5% respectively. The number of instances where the methods' UBs are equal to optimal are 9, 9, 6 and 10, respectively.

For large instances, TLAP is not able to find any optimal solution. Decomposition methods also fail to converge within time limit. Average gap obtained by MC, SGC, PD and B&C-PD are 9.4%, 8.8%, 7.6% and 17.9%. The number of instances where the methods' UBs are equal to the best known values are 7, 10, 6 and 8, respectively.

Overall, we observe that MC, SGC and PD produce comparable gap values, while B&C-PD is unsuccessful with respect to this performance measure. SGC and PD are the most successful methods with medium and large instances, respectively. For medium and large instances, where solution time is always limited to the time limit, we cannot make a comparison with respect to this performance measure. For small instances, PD is by far the fastest solution method.

Among the exact solution methods we considered in this study, we believe SGC and PD are the two most successful ones. In Chapter 6, we give the results from further experiments with these methods.

CHAPTER 5

APPROXIMATE SOLUTION METHODS AND PRELIMINARY COMPUTATIONAL RESULTS

Methods based on Benders Decomposition are more successful with large problem instances than the standard solvers, as expected. Although they do not converge within the time limit, they are able to produce good upper bounds, which are often found in early steps and stay as the best solution until the end. Most of the time it is hard to find any other solutions to which these solutions can be compared for large instances. This makes it difficult for a benchmarking study to be performed.

We usually observe that the methods based on Benders Decomposition find optimal solutions, but they may not converge within time limit in order to certify optimality for medium instances.

In this chapter, we introduce an evolutionary algorithm to deal with the benchmarking issues, assessment of results of computational experiments, and finding a good upper bound in a reasonable time.

We briefly summarize the relevant studies in the literature. Kung-Jeng et al. (2011) propose a genetic algorithm for the location-allocation problem in a two-echelon supply chain with stochastic demand. Altınel et al. (2009) develop a location-allocation heuristic for the capacitated multi-facility Weber problem with probabilistic customer locations. Stanimirovic and Kratica (2007) present two heuristics based on genetic algorithms and fast interchange heuristic in order to solve the discrete ordered median problem. Bischoff and Dachert (2009) compare several heuristic algorithms for the solution of generalized location-allocation problems. Their primary focus is on the local search algorithms for the allocation decisions. Lastly, Rajagopalan et al. (2007) develop four metaheuristic algorithms for a probabilistic location model. We analyzed the operators of the algorithms in these studies, and the conventional ones in the literature, before developing our algorithm.

In this section, we first explain the details of the algorithm. Then, we conduct a factorial design analysis to find the best parameter values. The major steps of an evolutionary algorithm can be illustrated as given in Figure 5.1:

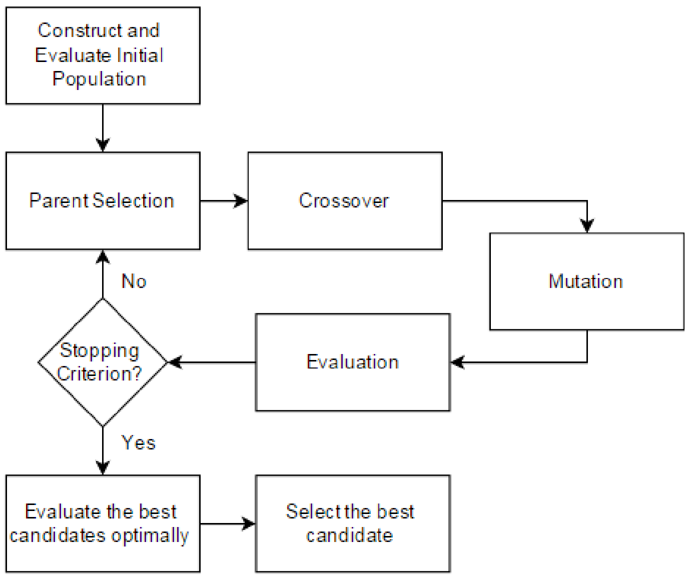

Construct and Evaluate Initial Population
Parent Selection
Crossover
Mutation
No
Stopping Criterion?
Evaluation
Yes
Evaluate the best candidates optimally
Select the best candidate

Figure 5.1: Evolutionary Algorithm

Now, we go into the detail of the algorithm steps one by one.

5.1 Representation Scheme

We use a simple binary representation scheme, only storing the first-stage decisions in the chromosomes. This is a valid approach, because there is only a single optimum set of second-stage decisions for each set of first-stage decisions. Thus, no information is lost by storing only the first-stage decisions.

Also, we do not need to include the second-stage decisions in the chromosomes, because it would not be meaningful to apply algorithmic operators (crossover and mutation) to them. For example, we cannot perform a crossover operation between the second-stage decisions of two solutions if their first-stage decisions are different.

A typical chromosome looks like the one given in Example 5.1:

Example 5.1.

CDC Candidate	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Open/Closed	1	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0

Each CDC candidate is assigned a binary variable determining whether it is open or not.

5.2 Decoding and Fitness Evaluation

In our case, decoding is a very important part of the algorithm, since the whole set of second-stage decisions are to be modeled in this part. There are several alternative methods that can be used to obtain these decisions: making optimal decisions, using (nested) metaheuristics, using heuristics.

Making optimal decisions for the second-stage is possible through solving a simple mathematical model for each chromosome. Note that we fix the first-stage decision variables to the values on the chromosome. Since the second-stage decision variables are all continuous, this process is expected to be relatively short. However, the process may take too much time in application. For instance, our preliminary experiments showed that for large instances, evaluation of each chromosome in this way takes more than 90 seconds. Considering that the algorithm needs to evaluate possibly thousands of distinct chromosomes, we believe that making optimal decisions would not serve our purpose.

Instead of dealing with longer computational times, it is possible to use metaheuristics embedded within an algorithm focusing on the first-stage decisions. Second-stage decisions can be made using another evolutionary or tabu search algorithm. If we consider the large instances, let us say each of which include 1000 customers and 100 scenarios, the estimated time required for convergence would still be too large.

For such an evolutionary algorithm, using crossover and mutation operators to change customer allocations, it would not be efficient in finding good solutions compared to a simple heuristic. Due to the random nature of evolutionary algorithms, these operators are preserving the solution variety, as the algorithm relies on evaluating a large number of alternatives in finding a good solution. Such a process may waste the precious solution time by disregarding the allocation decisions that can be easily made. To direct the search, we can make local search within the algorithm, using an improvement heuristic. We believe that such an improvement heuristic can be successful without an evolutionary algorithm and find the solutions of similar quality in much shorter time.

If implemented efficiently, it also provides as good results as a tabu search algorithm. The main advantage of tabu search algorithms over simple heuristics is their permitting moves that worsen the solution performance. In this way, tabu search algorithms escape local optima. In our case, in which the customer demand is usually much smaller than the CDC capacities, local optima are very close to the global optimum. We can expect a simple swapping move to find good results. With these points in mind, we develop a heuristic for making second-stage decisions.

5.2.1 Hybrid Algorithm

We first consider developing a greedy algorithm for making the second-stage decisions. At each iteration, the algorithm simply searches the transportation costs between non-assigned customers and open CDCs and then finds customer-CDC pair that incurs the smallest cost and assigns the customer to that CDC, provided the CDC still has enough capacity.

Unfortunately, it did not work very well. We have seen that the solutions produced are not very close to the optimum. On average, we observe a large gap for the solution performance of greedy algorithm from the optimal solution.

The reason for the failure of greedy algorithm is that the customers close to the city center in our problem setting. All these customers who are far away from all CDCs are always assigned in the later iterations of the algorithm. Since some CDCs are already working at full capacity in these stages, such customers cannot be assigned to their closest CDCs. Instead, they are assigned to the closest CDCs with some available capacity.

To prevent such an outcome, we need to improve the solution found by the greedy algorithm. When CDCs are at full capacity, an improving move can be in the form of a swap. Thus, an improvement heuristic such as 2-opt or 3-opt may suit very well to the purpose.

Applying 2-opt or 3-opt to the whole set of customers is very time-consuming for large instances. As expected, preliminary experiments showed that the time requirement of 2-opt (3-opt) is close to (much larger) than the time needed for optimizing the second stage decisions.

An obvious remedy to this problem is to reduce the number of customers considered in the improvement heuristic. Unfortunately, it is not possible without reducing the solution performance. However we can benefit from problem specific knowledge to keep solution performance as high as possible, while reducing the time requirement.

Locations of CDC candidates being on the city boundaries is a very important feature of our problem. A typical instance can be seen in Figure 5.2. As mentioned earlier, customers close to the city boundaries can easily be assigned to their closest CDCs by the greedy algorithm, but this is not the case for customers close to the city center. Clearly, it does not make sense for us to consider these customers in the improvement stage. Instead, we determine a range value, and assignments incurring transportation cost below that value are not considered in the improvement stage. Reducing this value is expected to improve solution performance, while increasing it deteriorates solution performance.

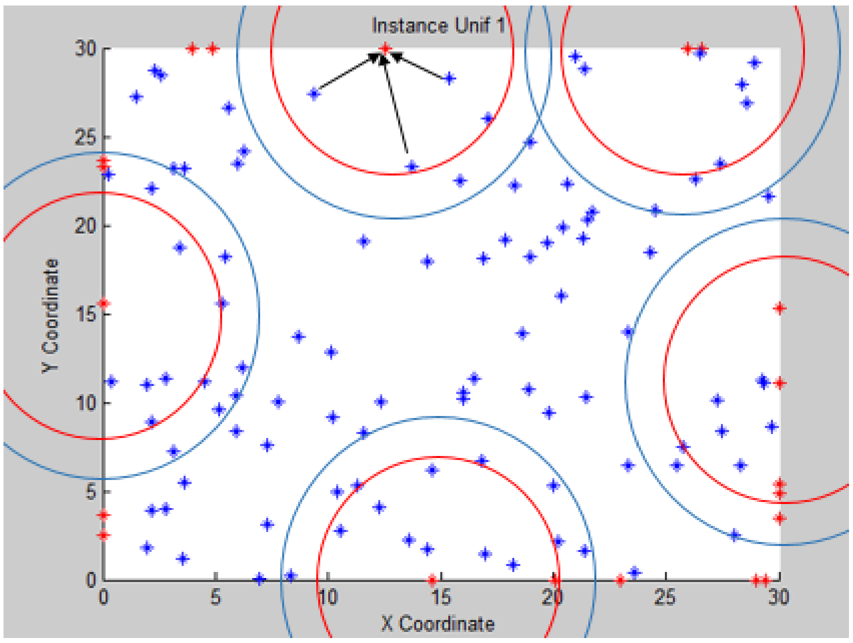


Figure 5.2: Range Values of the Hybrid Algorithm

We made experiments combining the greedy algorithm to construct a feasible solution and the swapping algorithm to improve this solution. The results are summarized in the following table. 20 experiments are made with different sets of location decisions for each combination of parameters. Note that average gap to optimal values only take into account the transportation cost terms of the objective function.

Table 5.1: Comparison of Evaluation Methods

Method	Range Value	Average # of Nodes Included	Average Gap to Optimal (Tr. Cost) (%)	Average Time (sec)
Greedy+2opt	25	204.3	1.4	19.4
	35	120.7	2.1	7.9
	40	81.0	2.6	6.1
	50	24.3	3.2	5.2
	70	2.1	3.4	4.6
	100	0.3	3.5	4.6
Greedy+3opt	25	204.3	-	+90.0
	35	120.7	-	+90.0
	40	81.0	-	+90.0
	50	24.3	3.2	60.1
	70	2.1	3.4	7.2
	100	0.3	3.5	5.0
LP solution	N/A	N/A	0.0	90.0

Range values are determined so that the effect of small and large values can be observed. Also, values smaller than 25 are not considered, since the time requirement gets close to 90 seconds even for the first combination. Thus, the method loses its advantage over LP solution.

It can be easily seen from the table that the average gap to optimal does not change when 3-opt algorithm is used instead of 2-opt. This is because the demand from individual customers are very small in comparison to the CDC capacity. Therefore, we may expect the 2-way swaps to cover most or all of the improving moves. 3-way swaps only make sense when 3 customers that are assigned to 3 different CDCs must be swapped for improvement. However, due to the same reason mentioned above, such a swap is also possible with 2 consecutive 2-way swaps. When we compare the average times they take, we can say that the first combination, i.e. greedy algorithm and 2-opt, is much better.

Also when we compare the heuristic solutions and optimal solutions for a large number of chromosomes (See Figure 5.3). While some heuristic solutions are close to the optimum, others are very far from it. We are mostly interested in the ranking of chromosomes, because the algorithm is expected to converge to the best solution, no matter how well it has been evaluated. If the ranking of chromosomes is wrong, the algorithm continues the search in the wrong direction.

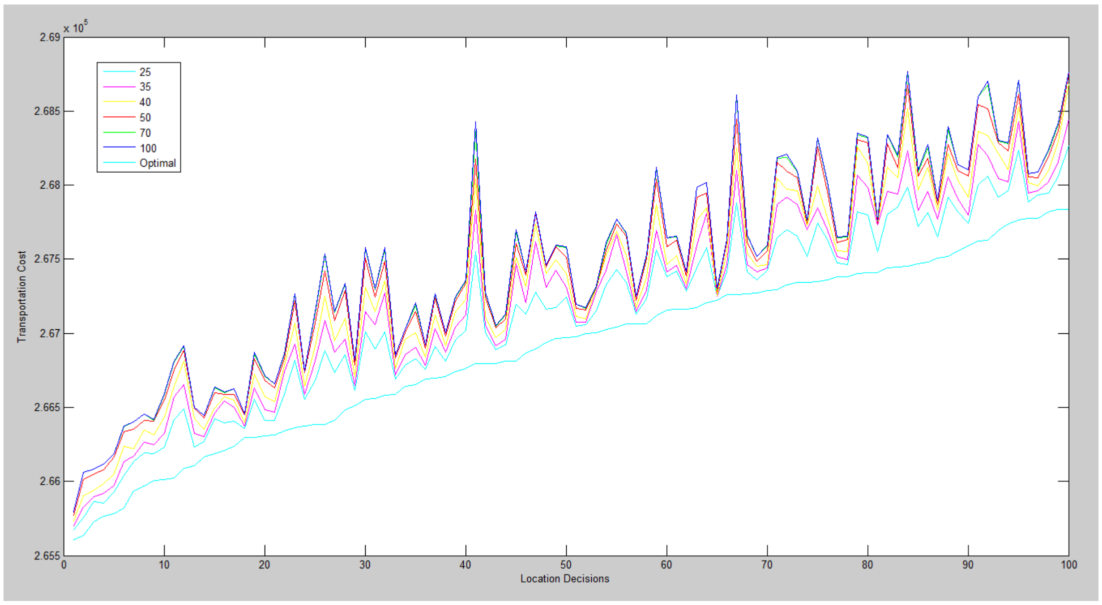


Figure 5.3: Performance of the Hybrid Algorithm with Constant Ranges

At this point an important observation was made. The range values determined above are constant for all scenarios. However, we construct our scenarios in groups to represent low congestion, high congestion and local congestion cases. This creates differences between the average transportation costs of different scenarios. Therefore, a constant distance limit affects each scenario differently. While hundreds of customers are considered for 2-opt in one scenario, only a few customers may be considered in another. This way, allocation may be made badly in one scenario, while it may be made near-optimally in another. Since we are interested in the average cost through scenarios, we need to make equally good decisions in all scenarios.

The solution we found to this problem is using variable range values. This time, they are determined specifically for each scenario, depending on the average and standard deviation of transportation cost. Table 5.2 summarizes the results. We do not consider 3-opt in this setting, since the previous results showed no significant difference between using 2-opt or 3-opt, despite the higher time requirement of 3-opt.

Table 5.2: Comparison of Evaluation Methods

Method	Range Value (Std. Dev.)	Average # of Nodes Included	Average Gap to Optimal (Tr. Cost)	Average Time (sec)
Greedy+2opt	0.0	423.5	0.2	36.3
	0.5	205.3	0.8	18.3
	1.0	112.5	2.0	9.7
	1.5	82.0	2.7	6.7
	2.0	55.4	2.9	5.7
	2.5	33.2	3.0	5.2
	3.0	18.9	3.1	4.9
LP solution	N/A	N/A	0.0	90.0

The improvement obtained from using variable distance limits can be most clearly seen in a comparison between first row of Table 5.1 and second row of Table 5.2. While the average time it takes for the algorithms to complete are nearly the same on these rows, the average gap to optimal is significantly lower with range values are variable.

We make the same comparison with these range values, we obtain Figure 5.4. This time we observe a significant improvement in the stability of the performance of heuristic solutions. Fitness ranking of chromosomes is quite similar to the actual ranking. This leads us to believe that determining the range values in this way makes the heuristic performance reliable.

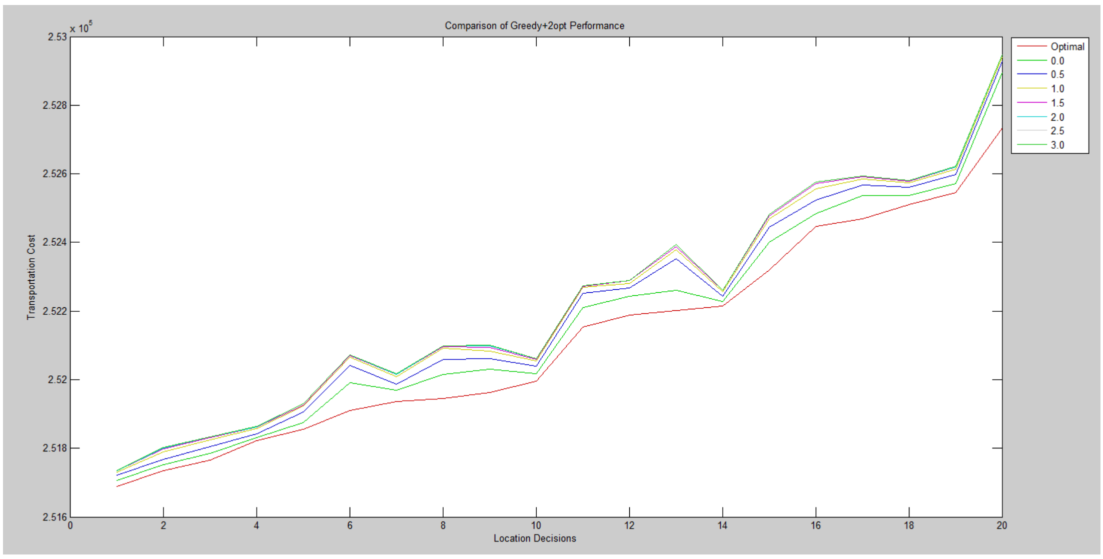


Figure 5.4: Performance of the Hybrid Algorithm with Changeable Ranges

We implement the hybrid algorithm in the evaluation step of the evolutionary algorithm. We use variable distance limits. The last decision regarding this step is determining how many standard deviations should be used for selecting the distance limits. σ=0 takes too much time, so it will not be considered. σ=0.5 again takes long, but it provides a significant benefit in the solution performance. σ=1.5, σ=2, σ=2.5 and σ=3 are close to each other with respect to solution performance, thus σ=3 can be considered due to its time requirement being shorter.

The average gap of 3% is still a little problematic, especially for small and large size instances. Although, the main algorithm is able to find the chromosome containing optimal decisions, it may converge towards a different solution with seemingly better performance due to the suboptimal allocation decisions. We could think of two different solutions to this problem.

As the main algorithm gets closer to satisfying the stopping condition, the distance limit may be reduced. This way, a finer evaluation of chromosome is possible without resorting to time-consuming methods like solving LPs. When this solution is implemented, we found that σ=0.5 performs sufficiently well. The timing we choose to switch to σ=0.5 will be explained later in the subsection 7.8.

After the main algorithm converges, we considered all the chromosomes evaluated so far, sorted them according to their fitness values (which are found with the hybrid heuristic) and selected the best chromosomes for exact evaluation. If the best fitness found so far is scaled to 1, we select the chromosomes until fitness value 1.05. If the optimal set of location decisions have already been found, the corresponding chromosome will most likely fall into the set selected for exact evaluation. Exact evaluation is made with the help of an LP model that takes fixed location decisions as input and produces optimum allocation decisions.

Algorithm 5.1 describes the steps of the hybrid algorithm. It takes the chromosome, CDC capacities, customer demands, transportation costs and range value as inputs and calculates the total cost of assignment decisions.

Algorithm 5.1 Hybrid Algorithm (HA)

```
Input: Chromosome, CDC cap.s, Cust. demands, Transp. costs, Range value
Output: Cost of assignment
 1: Initialize assignment variables
 2: for $s = 1\ to\ |S|$ do
 3:     Initialize CDC capacities
 4:     Initialize customer demands
 5:     while $\sum_j Dem_j > 0$ do
 6:         let $T_{ijs}$ be the minimum cost between an unassigned customer and a
            CDC with sufficient capacity
 7:         $Assign_{ijs} \leftarrow 1$
 8:         $Cap_i \leftarrow Cap_i - Dem_j$
 9:         $Dem_j \leftarrow 0$
10:     end while
11:     Initialize customer demands
12:     $Indicator \leftarrow Assign * T$
13:     for each customer pair with $Indicator > \mu(T) + limit * \sigma(T)$ do
14:         if $Dem_{j_1} + Cap_{i_1} > Dem_{j_2}$ and $Dem_{j_2} + Cap_{i_2} > Dem_{j_1}$ do
15:             $Impr_{i_1 j_1 i_2 j_2} \leftarrow T_{i_1 j_1 s} + T_{i_2 j_2 s} - T_{i_1 j_2 s} - T_{i_2 j_1 s}$
16:         else
17:             $Impr_{i_1 j_1 i_2 j_2} \leftarrow -1$
18:         end if
```

```
19:     end for
20:     let $Impr_{i_1 j_1 i_2 j_2}$ be the largest improvement
21:     while $Impr_{i_1 j_1 i_2 j_2} > 0$ do
22:         $Assign_{i_1 j_1 s} \leftarrow 0$
23:         $Assign_{i_2 j_2 s} \leftarrow 0$
24:         $Assign_{i_1 j_2 s} \leftarrow 1$
25:         $Assign_{i_2 j_1 s} \leftarrow 1$
26:         $Cap_{i_1} \leftarrow Cap_{i_1} + Dem_{j_1} - Dem_{j_2}$
27:         $Cap_{i_2} \leftarrow Cap_{i_2} + Dem_{j_2} - Dem_{j_1}$
28:         $Indicator \leftarrow Assign * T$
29:         for each customer pair with $Indicator > \mu(T) + limit * \sigma(T)$ do
30:             if $Dem_{j_1} + Cap_{i_1} > Dem_{j_2}$ and $Dem_{j_2} + Cap_{i_2} > Dem_{j_1}$ do
31:                 $Impr_{i_1 j_1 i_2 j_2} \leftarrow T_{i_1 j_1 s} + T_{i_2 j_2 s} - T_{i_1 j_2 s} - T_{i_2 j_1 s}$
32:             else
33:                 $Impr_{i_1 j_1 i_2 j_2} \leftarrow -1$
34:             end if
35:         end for
36:         let $Impr_{i_1 j_1 i_2 j_2}$ be the largest improvement
37:     end while
38: end for
39: $Cost \leftarrow \sum(Assign * T)$
```

5.2.2 Fitness Database

No matter which method we select, fitness evaluation of chromosomes is computationally expensive. Even when σ=3, it takes around 5 seconds for each chromosome. Due to the large number of chromosomes that requires evaluation, most of the solution time is spent on evaluating chromosomes.

We make use of a matrix called fitness database to prevent the evaluation of the same chromosomes in different generations. This matrix stores the chromosomes in its rows and the corresponding fitness values at the end of each row. Before evaluating each chromosome, we first check this database. If the chromosome is already evaluated, we simply take the fitness value from the database; if not, we evaluate it using the hybrid algorithm.

When the standard deviation value is changed, for instance from σ=3 to σ=0.5, all entries in the database are cleared, since the old fitness values are most probably not equal with the finer evaluation that can be done with the smaller distance limits.

5.3 Initial Population

We use two methods to generate the initial population. If the evolutionary algorithm is to be run after a Benders Decomposition based algorithm, it is possible to include the good solutions found by that method in the initial population of evolutionary algorithm. Knowing that these are all good solutions, we can expect EA to converge quickly. If the number of initial solutions obtained in this way is fewer than the population size, the remaining solutions can be generated randomly or by modifying the original solutions.

Preliminary experiments showed that the effect of using this approach is not satisfactory. While the convergence is quicker as expected, the good solutions found in the initial population rapidly direct the search to their neighborhood, and the algorithm often converges without finding a better solution than the best solution in the initial population. Changing the algorithm parameters such as crossover rate, mutation rate or tournament selection coefficient does not change this behavior.

On the other hand, when the initial population is constructed with randomly generated chromosomes, we observe slightly slower convergence. However, the algorithm is often able to find better results than the ones found in the previous setting. This change may be attributed to the diversity provided by the initial population. For a significantly diverse initial population, the algorithm is able to search a larger portion of the solution space. Due to these reasons, we choose to construct the initial population with randomly generated chromosomes. Since the chromosomes are simple constructs and all set of open CDCs are valid as long as they have sufficient capacity, we do not use a dedicated heuristic for selecting good locations.

Chromosomes are randomly generated in the following way. We start with a chromosome where all entries are zero. When we call the repair function, which will be explained later in subsection 7.6. The repair function selects random closed CDCs and opens them. This process is repeated until the total capacity is sufficient for total demand.

Algorithm 5.2 describes the steps of the process. The algorithm operates a single loop for generating “population size” many chromosomes. RO is the abbreviation of Repair Operator.

Algorithm 5.2 Initial Population Generator (IPG)

```
Input: $PSize$
Output: Initial population
 1: for $i = 1$ $to$ $PSize$ do
 2:     $Pop_i \leftarrow RO(zero\ chromosome)$
 3: end for
```

5.4 Parent Selection

We use the roulette wheel selection method to select the parents that will be included in the mating pool. We rank the chromosomes according to their fitness and assign them numbers that will determine selection probability. The best chromosome takes 0, the second one 1, the third one $1+p$, the fourth one $1+p+p^2$ etc. When a random number generated between 0 and the number assigned to the last chromosome, we select the chromosome with the largest number smaller than the random number. The selection is made with replacement, so that the chromosomes are allowed to be represented more than once in the mating pool. We repeat this process until the mating pool is full.

The *p* value has significant effect on the algorithm behavior. We will call *p* the “Elitism Rate”. When the elitism rate is higher, we apply elitism less strongly during the selection process. When *p* is small, the good solutions have a large selection probability, while bad solutions are rarely selected. This leads to faster convergence and less genetic variety, since the bad solutions cannot produce offspring and their genes are left out of the population. On the other hand, when *p* is large, the probability of selecting good or bad chromosomes are not very different. This leads to slower convergence and more genetic variety.

We work with p values around 0.95 and 0.97 to conserve genetic variety. Genetic variety is much more valuable than saving time with faster convergence, since the algorithm performance is largely affected by genetic variety.

Algorithm 5.3 describes the steps of parent selection process. The algorithm takes population size, number of matings, selection rate, population and fitness values of population members to create the mating pool.

Algorithm 5.3 Parent Selector (PS)

```
Input: $PSize, NMatings, SRate, Pop, Z_{Pop}$
Output: $MPool$
 1: $Pop \leftarrow$ sort $Pop$ in descending order of $Z_{Pop}$
 2: for $i = 1\ to\ PSize$ do
 3:     $S_{Pop_i} \leftarrow SRate^{i-1}$
 4: end for
 5: for $i = 1\ to\ NMatings * 2$ do
 6:     generate $rnd$, a standard random variable
 7:     for $j = 1\ to\ PSize$ do
 8:         if $S_{Pop_{j+1}} > rnd * S_{Pop_{PSize}}$ do
 9:             $MPool_i \leftarrow Pop_j$
10:             break
11:         end if
12:     end for
13: end for
```

5.5 Crossover

One-point and two-point crossover operators are the most frequently used operators in the facility location literature. However, we believe these operators are very inefficient, at least in our case, since they often produce infeasible solutions. Even when they manage to produce feasible solutions, the number of open facilities may be more than needed, leading to a bad solution performance. Often repair or local search operators are used in order to modify the resulting offspring into reasonable solutions. This process is often time-consuming and can be prevented with an effective crossover operator.

We use a simple operator. The parent chromosomes, taken as arrays, are multiplied with a constant smaller than or equal to 0.5. The result of their summation includes entries equal to 0, 1, or a fractional value. From this resulting array, both child chromosomes are constructed randomly. If the summation includes 1's, the corresponding CDCs are opened in both child chromosomes. Other CDCs with nonzero genes are opened randomly until the total capacity is sufficient to serve all customers. If the summation does not include any 1's, the CDCs with nonzero genes are opened randomly until the same condition is satisfied.

Selection of the constant may be left to the decision maker. If it is taken 0.5, the CDCs that are open in both parents will always be open in both children. If it is taken smaller than 0.5, this is not the case. In our case, we prefer the constant to be smaller than 0.5, to conserve or even increase the population diversity.

We believe we serve two conflicting objectives at the same time with this operator. First, we are able to transfer the genetic information from parents to their offspring; second, we conserve genetic variety by the random selection of candidate CDCs. The computational results also show that the operator is successful.

Example 5.2 illustrates the crossover process. Crossover constant is selected as 0.4. The selected parents are multiplied with the constant. We call their summation the mold, since two chromosomes are shaped by it. Then, the two chromosomes are generated in the way explained above.

Example 5.2.

Parent 1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Open/Closed	1	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0

Parent 2	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Open/Closed	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0

Mold	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Open/Closed	0.8	0.0	0.0	0.0	0.0	0.4	0.8	0.0	0.4	0.0	0.4	0.4	0.0	0.0	0.4	0.4	0.0	0.0	0.0	0.0

Offspring 1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Open/Closed	1	0	0	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0

Offspring 2	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Open/Closed	0	0	0	0	0	1	1	0	1	0	0	1	0	0	0	1	0	0	0	0

We prefer a large crossover rate to quickly search the solution space. Nevertheless, we will investigate the effect of alternative rates in the factorial design, which will be explained in the subsection 7.10.

Algorithm 5.4 describes the steps of the crossover operator. It takes number of matings, crossover rate and mating pool as inputs and creates the offspring.

Algorithm 5.4 Crossover Operator (CO)

```
Input: $NMatings, CRate, MPool$
Output: Offspring before mutation
 1: $i \leftarrow 0$
 2: while $i < NMatings * 2 - 1$ do
 3:     generate $rnd$, a standard random variable
 4:     if $rnd < CRate$ do
 5:         $Parent_1 \leftarrow MPool_{i+1} * 0.4$
 6:         $Parent_2 \leftarrow MPool_{i+2} * 0.4$
 7:         $Offs_{i+1} \leftarrow RO(Parent_1 + Parent_2)$
 8:         $Offs_{i+2} \leftarrow RO(Parent_1 + Parent_2)$
 9:     else
10:         $Offs_{i+1} \leftarrow MPool_{i+1}$
11:         $Offs_{i+2} \leftarrow MPool_{i+2}$
12:     end if
13:     $i \leftarrow i + 2$
14: end while
```

5.6 Mutation

Mutation is also carried out with a simple operator. One of the open CDCs is closed randomly and then the resulting chromosome is repaired. We always close only 1 open CDC in this way.

The following is an example of the mutation process. At first, the original chromosome indicates that the set of open CDCs is {1,6,7,11,12}. We randomly select CDC-7 to be closed. Then CDC-16 is opened again randomly. The repair process is complete since the capacity requirement is satisfied.

Example 5.3.

Original Chr	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Open/Closed	1	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0

Mutant Chr	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Open/Closed	1	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Repaired Chr	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Open/Closed	1	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	0	0	0	0

We again prefer a large mutation rate, so that the genetic variety of the population is maintained. A chromosome is subject to mutation with probability between 0.3 and 0.5 in our preliminary experiments. These probabilities translate roughly into the mutation probabilities 0.06 and 0.10 for open CDCs; 0.02 and 0.03 for closed CDCs.

Algorithm 5.5 describes the steps of the mutation process. It takes the number of matings, mutation rate and set of child chromosomes as inputs and updates the offspring.

Algorithm 5.5 Mutation Operator (MO)

```
Input: $NMatings, MRate, Offs$
Output: Offspring after mutation
1: for $i = 1\ to\ NMatings * 2$ do
2:     generate $rnd$, a standard random variable
3:     if $rnd < MRate$ do
4:         $Offs_i \leftarrow$ close a random CDC that is open in $Offs_i$
5:         $Offs_i \leftarrow RO(Offs_i)$
6:     end if
7: end while
```

5.6.1 Repair

Since the chromosomes represent only the location decisions, repair process only deals with the feasibility conditions related to those decisions. Feasibility of allocation decisions is handled by the decoding/fitness evaluation heuristic. Therefore, the repair process focuses on maintaining sufficient CDC capacity for serving all customer demand.

Repair algorithm ensures feasibility through opening additional CDCs randomly until the capacity condition is satisfied. This is a simple operation that we also carry out in the crossover and mutation operators. For instance, in the crossover operator, once the mold is constructed, all we do is to open random CDCs with nonzero representation.

Similarly to the mutation operator, we open a random CDC. To create a cleaner code, we make the CDC opening operations of other algorithms by calling the repair operator. This way, the only operations left to the crossover algorithm are creating the mold and calling the repair function. Similarly, the mutation algorithm only closes an open CDC and calls the repair function.

Algorithm 5.6 describes the steps of the repair algorithm. It takes a chromosome as input and repairs it.

Algorithm 5.6 Repair Operator (RO)

```
Input: Chromosome
Output: Repaired chromosome
  1: while $total\ capacity < total\ demand$ do
  2:     $Chr$ ← open a random CDC that is closed in $Chr$
  3:     update total capacity
  4: end while
```

5.7 Replacement

Once we obtain all the child chromosomes, we are ready for starting a new generation. There are two extremes in selecting the members of the new generation: total elitism and random selection. Total elitism leads to fast convergence while risking getting stuck in local optima. Random selection risks no convergence at all. There are several replacement methods used in the literature attempting to find a compromise between these two extremes.

Fitness scaling and fitness windowing do not work well for our problem, because the difference between fitness values of good solutions are very small in some intervals and very large in others. This leads to large fluctuations of selection probability, which works against genetic variety by eliminating slightly worse solutions.

To prevent fluctuating selection probabilities, we rank the chromosomes according to their fitness values and assign selection probabilities according to their ranks, instead of their fitness values.

The best chromosome takes 0, the second one 1, the third one 1+*p*, the fourth one 1+*p*+p^2 etc. When a random number is generated between 0 and the number assigned to the last chromosome, we select the chromosome with the largest number smaller than the random one. Selection is made without replacement. We repeat this process until the new population is full.

Algorithm 5.7 describes the steps of the replacement algorithm. It takes population size, number of matings, replacement rate, population, fitness values of population members, offspring and fitness values of offspring members. It creates the new population using these inputs.

Algorithm 5.7 Replacement Algorithm (RA)

```
Input: $PSize, NMatings, RRate, Pop, Z_{Pop}, Offs, Z_{Offs}$
Output: New population
 1: $Cand \leftarrow$ concatenate $Pop$ with $Offs$
 2: $Z_{Cand} \leftarrow$ concatenate $Z_{Pop}$ with $Z_{Offs}$
 3: $Cand \leftarrow$ sort $Cand$ in descending order of $Z_{Cand}$
 4: for $i = 1\ to\ Psize + NMatings * 2$ do
 5:     $S_{Cand_i} \leftarrow RRate^{i-1}$
 6: end for
 7: for $i = 1\ to\ PSize$ do
 8:     generate $rnd$, a standard random variable
 9:     for $j = 1\ to\ Psize + NMatings * 2$ do
10:         if $S_{Cand_{j+1}} > rnd * S_{Cand_{Psize+NMatings*2}}$ do
11:             $Pop_i \leftarrow Cand_j$
12:             $Z_{Pop_i} \leftarrow Z_{Cand_j}$
13:             break
14:         end if
15:     end for
16: end for
```

5.8 Stopping Condition

The fitness values, in other words the total costs, of solutions represented with each chromosome take continuous values. Therefore, a natural approach to monitor the convergence of the algorithm is observing the gap between the average fitness of the population and the fitness of the best individual.

At the end of each generation, we calculate the percent gap between the population's average fitness and best fitness. We assume the population converges when this gap is very small; the threshold value we use is 10^{-5}%. Since we observed that there appears to be several solutions with very close fitness to optimum, we select such a small convergence gap. This way, we wait until the algorithm completes its search within the region close to the optimal.

While checking the stopping condition, we also check if the convergence gap has been reduced below a second threshold value, which determines if the range value will be changed. As explained in the subsection 7.2, we may prefer to make a finer evaluation of the chromosomes in the later stages of the algorithm. To do so, we determine this threshold, typically taken 1%. Once the algorithm has achieved this level of convergence, we can see that the precision of the evaluation heuristic becomes insufficient with σ=3. Therefore, we update the range value o to σ=0.5 and make a more precise evaluation. Before the gap falls below 1%, making such a precise evaluation does not bring significant benefit. Moreover, it increases computational time substantially.

Updating the distance limit is actually very important with smaller instances. Since there are fewer customers in those instances, making a few suboptimal assignments has a larger impact on the solution performance, while the difference would be negligible for a large instance. Therefore, the solution generated by the heuristic may be very close to optimum for one chromosome, but far from optimum for another. For large instances, we expect the difference between heuristic and optimal solutions becomes more stable.

Updating the distance limit is not a must in this framework. The tradeoffs between the increased time requirement and more precise evaluation may be assessed in accordance with the instance of interest. In any case, the heuristic solutions cannot be precise enough for us to select the best solution with confidence. For this reason, we make exact evaluation at the last stage of the algorithm.

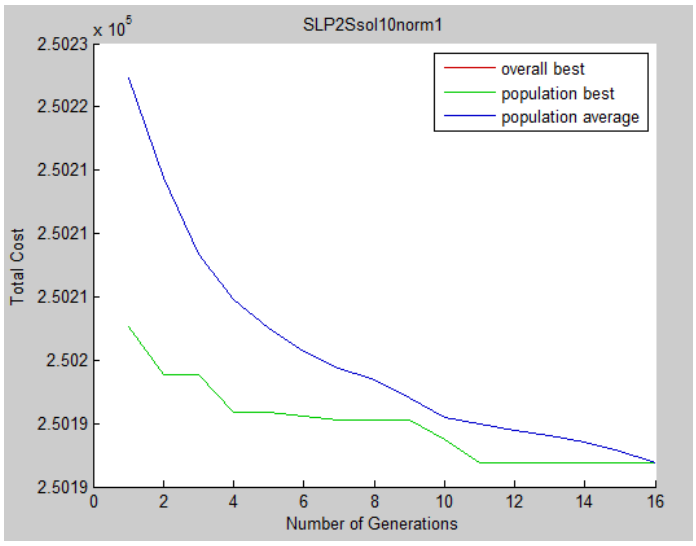


Figure 5.5: Convergence of the Evolutionary Algorithm

Number of generations required for convergence varies between 15 and 50 depending on the design parameters. With the parameters selected based on the factorial design, which will be explained in section 4.10, the algorithm takes between 15 to 25 generations to complete.

Algorithm 5.8 describes the steps followed for evaluating whether the stopping condition is satisfied. It takes population, threshold for updating range value and target gap as inputs. It determines the value of a binary number *stop* to indicate whether the algorithm converged.

Algorithm 5.8 Stopping Condition (SC)

```
Input: $Population, Threshold, CGap$
Output: Stopping condition
 1: $PopAvg \leftarrow \mu(Z_{Pop})$
 2: $PopBest \leftarrow min(Z_{Pop})$
 3: $Gap \leftarrow \frac{PopAvg-PopBest}{PopBest}$
 4: if $Gap < Threshold$ do
 5:     clear database
 6:     update range value
 7:     $Threshold \leftarrow 0$
 8: end if
 9: if $Gap < CGap$ do
10:     $stop \leftarrow 1$
11: end if
```

5.9 Exact Evaluation

As explained in the subsection 7.2, determining the distance limits according to the standard deviation greatly improves the performance of heuristic evaluation. Nevertheless, the evaluation algorithm still leaves average gaps from the optimum as large as 0.79% when σ=0.5 and 3.05% when σ=3. Since these values are averages, we can expect the gap to be larger for some chromosomes, possibly resulting in a wrong ranking order.

To be sure of selecting the best chromosome found by the algorithm, we make exact evaluation at the last stage, after the stopping condition is satisfied. From the fitness database, we select the best chromosomes to be evaluated optimally. The worst chromosome to be selected must have a fitness value *r* times the best fitness value found so far. We expect the true best chromosome to be included in this group with high confidence. Also, the value *r* can be changed in accordance with the heuristic algorithm parameters. For example, if σ=3 is used, *r*=1.05 may be necessary, while if σ=0.5 is used, *r*=1.02 may be sufficient. Note that a larger *r* value leads to more chromosomes to be evaluated in this stage, which takes longer.

The evaluation is made with the help of a mathematical model solver. We call an LP model after fixing the location decisions and solve the assignment model with the solver to obtain the allocation decisions. After the selected chromosomes are evaluated optimally, we select the best solution among them and terminate the algorithm.

Algorithm 5.9 describes the steps of the optimal evaluation process. It takes the final population produced in the evolutionary algorithm and returns the best solution according to the fitness values obtained through the LP model.

Algorithm 5.9 Optimum Evaluation (OE)

```
Input: Last population
Output: Best solution generated
1: for $i = 1\ to\ PSize$ do
2:     $Z_{Pop_i}$ ← optimum cost from the LP model (location decisions are fixed)
3: end for
4: sort $Pop$ in descending order of $Z_{Pop}$
5: $Best \leftarrow Pop_1$
```

Algorithm 5.10 describes the main steps of the evolutionary algorithm. It takes population size, number of matings, selection rate, replacement rate, crossover rate, mutation rate, range value, threshold value and target gap as inputs. It returns the best solution generated until convergence.

Algorithm 5.10 Main Algorithm

```
Input: $PSize, NMatings, SRate, RRate, CRate, MRate, Limit, Threshold,$
       $CGap$
Output: Best solution generated
 1: $gen \leftarrow 0, stop \leftarrow 0$
 2: $Pop \leftarrow IPG()$
 3: for $i = 1$ $to$ $PSize$ do
 4:     $Z_{Pop_i} \leftarrow HA(Pop_i)$
 5:     register $Pop\_i$ and $Z_{Pop_i}$ to database
 6: end for
 7: while $stop = 0$ do
 8:     $gen \leftarrow gen + 1$
 9:     $MPool \leftarrow PS(Pop)$
10:     $Offs \leftarrow CO(MPool)$
11:     $Offs \leftarrow MO(Offs)$
12:     for $i = 1$ $to$ $NMatings * 2$ do
13:         if $Offs_i$ is in database do
14:             $Z_{Offs_i} \leftarrow database(Offs_i)$
15:         else
16:             $Z_{Offs_i} \leftarrow HA(Offs_i)$
17:             register $Offs_i$ and $Z_{Offs_i}$ to database
18:         end if
19:     end for
20:     $Pop, Z_{Pop} \leftarrow RA(Pop, Z_{Pop}, Offs, Z_{Offs})$
21:     $stop \leftarrow SC(Pop)$
22: end while
23: $Best \leftarrow OE(Pop)$
```

5.10 Experimental Design

We make a factorial design analysis to determine the values of important algorithm parameters. In our analyses, we consider the selected algorithm parameters as factors.

We experiment with several levels to observe the effect of the factors on the algorithm performance. Table 5.3 presents the list of factors that we considered in our experimental design.

Table 5.3: Factors and Levels Considered in the Experiments

Factor	Level 1	Level 2
Population Size	50	100
Number of Matings	25	50
Replacement Rate	0.95	0.97
Crossover Rate	0.5	0.9
Mutation Rate	0.3	0.5

Population size is clearly one of the most important parameters of a EA, since it affects the behavior of the algorithm on several levels. Selecting the population size small may lead to a rapid convergence and lower solution times, but selecting it larger could make it more difficult for the population converge but possibly increase solution quality. Finding a good balance between computational requirements and solution quality is crucial for an effective implementation of EA. After a few preliminary experiments, we selected the levels 50 and 100 for the population size, because a larger population would not be justified by any advantages and a smaller population does not produce the sufficient diversity needed for detecting near-optimal solutions.

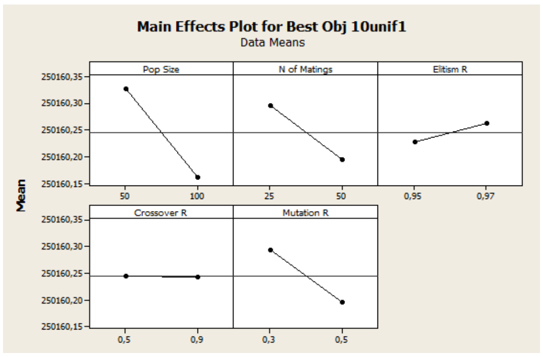
Number of matings is selected in accordance with the population size levels. When the number of matings is 25 and 50, the algorithm produces 50 and 100 child chromosomes at each generation, respectively. These numbers are sufficient to ensure that a new generation is significantly different than the previous one, and they are not too large to require too much computational time.

As explained earlier, we consider large replacement rates, so that the chromosomes with worse fitness still have a chance of being represented in the new generation. We experimented with two values, 0.95 and 0.97, to see the effect of replacement rate on the algorithm performance.

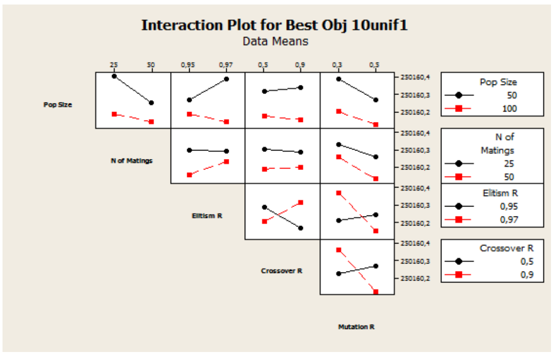
Levels for crossover and mutation rates are determined according to our preliminary experiments. We compare a small rate and a large rate for crossover, 0.5 and 0.9. The main difference between is that a small rate significantly reduces the number of distinct chromosomes to be evaluated, thus reducing overall solution time. However, a small rate leads to less diversity in the population, since a large part of child chromosomes are exactly the same as their parents. This may result in a narrow search of the solution space and convergence to suboptimal solutions. On the other hand, a large rate leads to greater diversity and longer solution time. Similar things can also be said for the mutation rate.

We considered two problem instances to measure the effect of different parameter combinations, small (called 10unif1, 10 customers) and medium (called 100unif1, 100 customers). We use two performance measures for comparison: total cost of best solution found (Best Obj) and solution time (Time). We created a 2^5 full factorial design containing the factors and levels in Table 5.3. 5 replications were made with each factor combination.

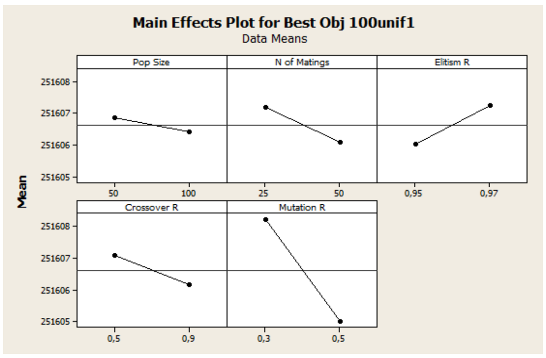
We provide the main effects plots for both performance measures in Figure 5.6 and residual plots for solution times in Figure 5.7. Since Best Obj values are mostly at the optimal level with some occasional divergence, the residual plots and half effect plots are not reliable for this performance measure.



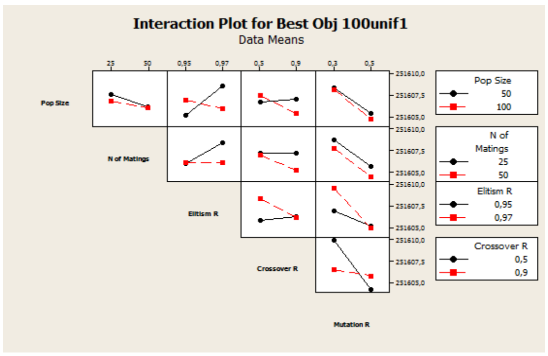
(a) Main effects plots for small instance



(b) Interaction plots for small instance



(c) Main effects plots for medium instance



(d) Interaction plots for medium instance

Figure 5.6: Main Effects and Interaction Plots of Objective Function Values

Main effects plots for “Best Obj” show significant effect by all factors. Especially the population size factor has very large impact on the solution performance, which can be attributed to the larger population diversity that can be maintained in a larger population. A similar observation can be made for the number of matings factor. We select the larger levels for both of these factors.

When the elitism rate is higher, we can observe a slight deterioration of the solution quality. As mentioned earlier, a higher elitism rate leads to weaker elitism; based on the analysis, we prefer applying stronger elitism with p=0.95.

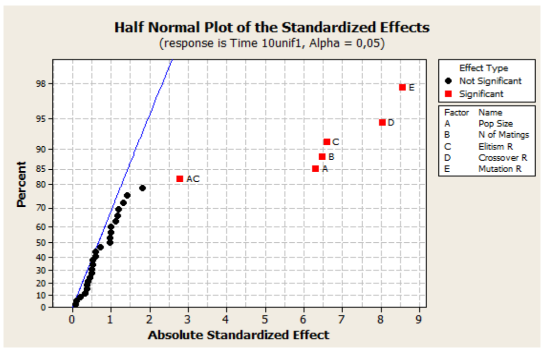
Increasing the crossover rate seems to have a significant effect only for the medium instance, still the level 0.9 seems to work well for both instances. Mutation rate is the most significant factor for the medium instance and also has significant effect for the small instance. We simply select the level 0.5.

In the light of the main effect plots, we select the following factor levels for use in our final computational experiments as in Table 5.4:

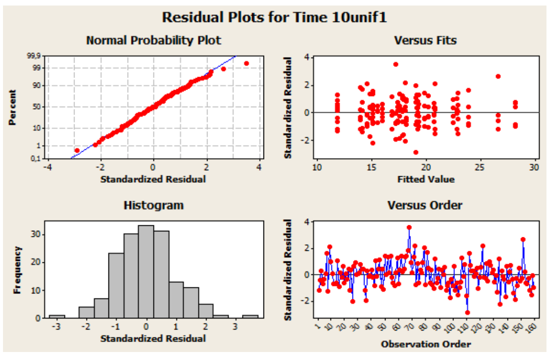
Table 5.4: Selected Factor Levels

Factor	Selected Level
Population Size	100
Number of Matings	50
Replacement Rate	0.95
Crossover Rate	0.9
Mutation Rate	0.5

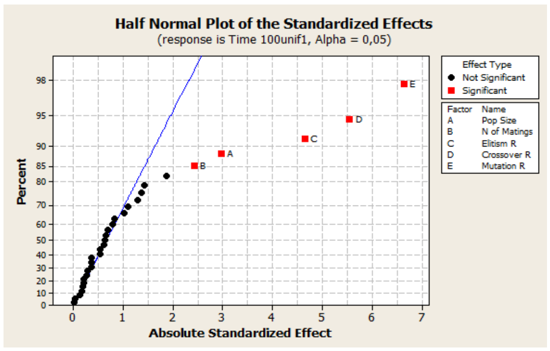
When we take a look at the interaction plots, we see that the best possible solution quality is obtained with the selected levels most of the time. There are four cases where the interaction plots disagree with our selection, but these are isolated cases and their effects are also neutralized by other interaction effects. Therefore, we continue with the levels listed in Table 5.4.



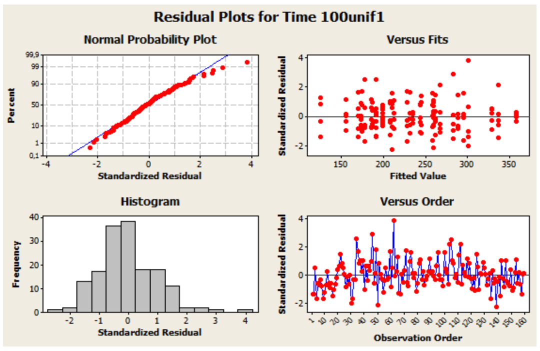
(a) Half normal plot for small instance



(b) Residual plots for small instance



(c) Half normal plot for medium instance



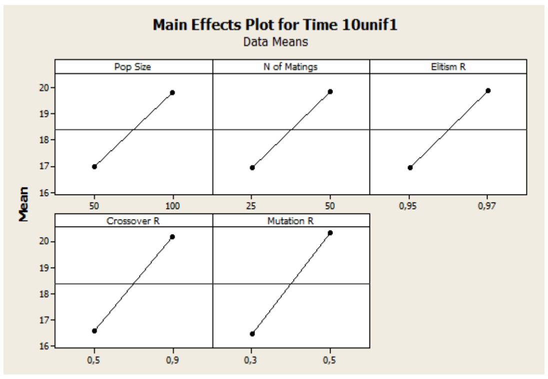
(d) Residual plots for medium instance

Figure 5.7: Half Normal and Residual Plots for Solution Times

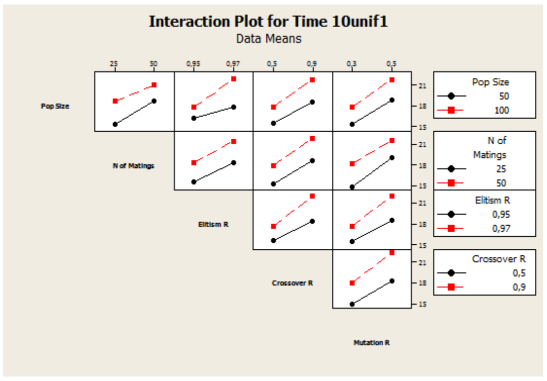
From the significance of factor effects we considered extending the experiments with additional levels. Especially the mutation rate seems to improve solution quality as it increases. However, applying mutation to more than half of the child chromosomes would conflict with the principle of carrying information from the older generation to the younger one and bring the risk of creating too many child chromosomes with low quality. This behavior would slow down the convergence, as well as aggravating the possibility of losing the optimal solution due to mutation. Also, we do not consider larger levels for population size and number of matings, since larger populations would take longer to be constructed with our large instances. Larger levels are rarely used in the literature anyway.

In this study, we prefer solution quality rather than solution time. However, we still analyze the effect of factors on the solution time. From the half normal plots in Figure 5.8, we easily see that the main effects of all factors are significant at the *p* level of 0.05. For the small instance, the significance of the effects is obvious, since the solution takes shorter time, thus a slight increase or decrease create a significant difference. We also observe that the interaction effect AC is significant only for the small instance, but we will not take that into account, as the results of two instances do not agree.

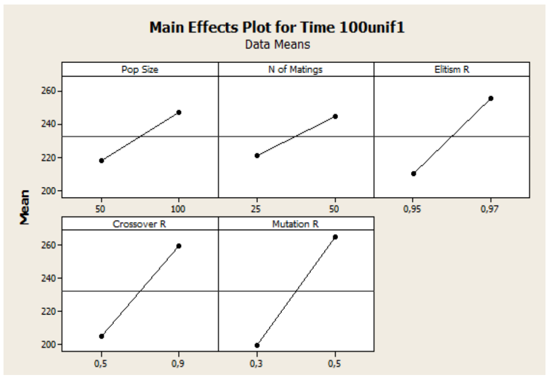
The residual plots show that the assumptions of the factorial design analysis are satisfied. Residuals are normally distributed in both cases. No bias can be observed with the distribution of residuals neither to negative or positive sides. Variance of residuals seem to be constant over time. Lastly, the observation order does not affect residual levels, which must be expected since the experiments are made on a computer.



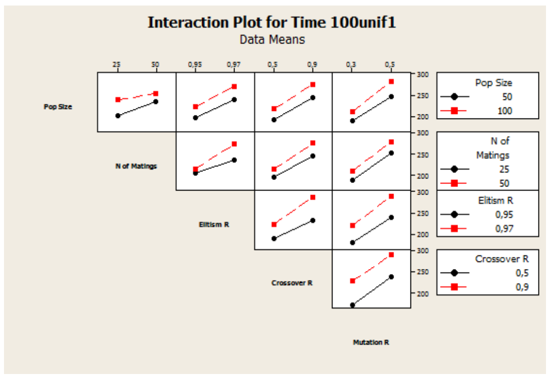
(a) Main effects plots for small instance



(b) Interaction plots for small instance



(c) Main effects plots for medium instance



(d) Interaction plots for medium instance

Figure 5.8: Main Effects and Interaction Plots for Solution Times

The main effect plots in Figure 5.8 agree with the half normal plots in that all main effects are significant. The factor levels that produce longer solution time are parallel to our expectations.

With a larger population size or a larger number of matings the computational requirement naturally increases, since there are more chromosomes to be evaluated by the algorithm.

With a larger elitism rate (with weaker elitism), the algorithm assigns a larger probability to the chromosomes with lower fitness, thus it takes longer for the algorithm to weed out the bad chromosomes and converge.

Lastly, larger crossover and mutation rates increase genetic diversity and lead the algorithm to search a larger part of the feasible region. Thus, it takes longer for the algorithm to converge with larger crossover and mutation rates.

None of the interaction effects are significant.

At this point, it is important to remember that we are implementing the evolutionary algorithm to find good solutions to medium and large instances due to the drawbacks of other solution methods. Standard solvers fail with large instances and they take too long with medium instances. Our algorithm based on the L-Shaped method does not converge within the time limit, thus we cannot guarantee if it find the optimal solutions. In this context, our primary expectation from the evolutionary algorithm is finding good solutions. Therefore, we do not aim to obtain the solutions in shorter time at the expense of lower solution quality. We use the factor levels selected according to the Best Obj main effect plots. Also, we use the same factor levels for instances of all sizes: small, medium, and large.

CHAPTER 6

COMPUTATIONAL RESULTS

In this section, we list and compare the computational results obtained with the help of different solution methods. As explained in the previous chapters, we consider three solution methods: solving a standard mathematical model by Cplex, using the Benders Decomposition algorithm, and using the evolutionary algorithm. Details on the last two methods are introduced in Chapters 4 and 5 respectively.

6.1 Framework

Our purpose in conducting the computational experiments is twofold:

- To identify the strengths and weaknesses of each method
- To observe the disparities between different instance groups with respect to solution efforts

Details of the methods are given in the previous two chapters. After observing the performances of several methods based on Benders Decomposition, we decided to use the L-Shaped method with scenario group cuts for our problem. For the evolutionary algorithms, we developed a hybrid fitness evaluation algorithm that combines greedy and 2-opt heuristics. We tune this algorithm for a precise evaluation and we make an exact evaluation at the last step of the evolutionary algorithm.

We use three sets of instances to test the solution methods. They are: small (10 customers), medium (100 customers), and large (1000 customers) instance sets. All test instances include 20 candidate CDC locations and 100 scenarios.

Each of these sets includes 24 instances in two groups. These groups are constructed according to the ratio of total fixed cost over total transportation cost. We believe this ratio may significantly affect solution time. Each group, instance groups with large and small ratios, contains 6*2 instances constructed in parallel to the square patterned instances used for the Value of Information analyses given in section 5. The six patterns are as follows: R_u, R_n, C_u, C_n, RC_u, and RC_n. In total, there are 72 instances we consider. The list of instances are as follows:

- Small:
    - Large Ratio: R_u_L_10_1&2, R_n_L_10_1&2, C_u_L_10_1&2, C_n_L_10_1&2, RC_u_L_10_1&2, RC_n_L_10_1&2
    - Small Ratio: R_u_S_10_1&2, R_n_S_10_1&2, C_u_S_10_1&2, C_n_S_10_1&2, RC_u_S_10_1&2, RC_n_S_10_1&2
- Medium:
    - Large Ratio: R_u_L_100_1&2, R_n_L_100_1&2, C_u_L_100_1&2, C_n_L_100_1&2, RC_u_L_100_1&2, RC_n_L_100_1&2
    - Small Ratio: R_u_S_100_1&2, R_n_S_100_1&2, C_u_S_100_1&2, C_n_S_100_1&2, RC_u_S_100_1&2, RC_n_S_100_1&2
- Large:
    - Large Ratio: R_u_L_1000_1&2, R_n_L_1000_1&2, C_u_L_1000_1&2, C_n_L_1000_1&2, RC_u_L_1000_1&2, RC_n_L_1000_1&2
    - Small Ratio: R_u_S_1000_1&2, R_n_S_1000_1&2, C_u_S_1000_1&2, C_n_S_1000_1&2, RC_u_S_100_1&2, RC_n_S_1000_1&2

We refer to the mixed integer program solved by the standard solver as TLAP. We refer to the scenario-group cuts and partial decomposition approaches as SGC and PD respectively. We refer to the evolutionary algorithm as EA.

For the standard model, we provide optimum cost value and solution time, if an optimal solution can be found. For SGC and PD methods, we provide the cost of best integer solution, % gap between upper bound and lower bound and solution time. Zero gap indicates that the algorithm has converged and cost value is optimal. Positive gap indicates that the algorithm could not converge within time limit. For the evolutionary algorithm, we make 5 replications for each instance. We provide the cost of best integer solution in 5 replications, best/average gap found in 5 replications, and average time for the algorithm to complete. The gap values are the percent difference from the optimal, if optimal solution is known. Zero gap indicates that the optimal solution is found by the evolutionary algorithm. If optimal solution is not known, gap indicates the percent deviation from the best known solution found by the exact solution methods (TLAP, SGC and PD). All solution times are shown in seconds.

SGC/PD: $Gap\ \% = \frac{UB-LB}{LB}*100$

EA: $Best\ Gap\ \% = \frac{Best\ of\ Repl-Opt}{Opt} * 100$

EA: $Ave\ Gap\ \% = \frac{Avg\ of\ Repl-Opt}{Opt} * 100$

We impose a time limit of 4 hours (14400 seconds) on each method. Once the time limit is reached, TLAP immediately stops and reports the most recent bounds found by the solver. However, SGC, PD and EA are allowed to complete the ongoing iteration before they report the results. For SGC and PD, we find it necessary for the last iteration to be completed, because the iterations tend to be gradually longer due to the cuts added. Enforcing the time limit without flexibility would mean the last iteration would be completely disregarded, which makes the effective time limit to be a lot shorter than 4 hours. A similar argument can be made for EA, since each iteration take a lot of time to complete with large instances.

All experiments are run on identical PCs with 3.00 GHz CPU and 16.00 GB RAM running MS Windows.

6.2 Results

We report the optimal/best known solution in the second columns of the Tables 6.1-6.3 and we omit the solutions found by each method. The UB and LB values obtained in the experiments can be found in the corresponding tables in the Appendix. In this section, we only report the gap and CPU time results for each method.

In Table 6.1 we provide the results for the small instances.

Table 6.1: Computational Results for Small Instances

Results for Small Instances	Optimal Solution	TLAP		SGC		PD		EA		
		GAP%	CPU Time (sec)	GAP%	CPU Time (sec)	GAP%	CPU Time (sec)	Best Gap from Optimal%	Avg Gap from Optimal%	Avg CPU Time (sec)
R_u_L_10_1	250160.1	0.0	2.6	0.0	618.5	0.0	375.2	0.0	0.0	16.2
R_u_L_10_2	250145.6	0.0	3.0	0.0	3293.4	0.0	1026.8	0.0	0.0	35.0
R_n_L_10_1	250179.6	0.0	7.2	0.0	4543.0	0.0	770.6	0.0	0.0	16.7
R_n_L_10_2	250162.6	0.0	1.4	0.0	654.2	0.0	192.2	0.0	0.0	23.3
C_u_L_10_1	250180.5	0.0	2.9	0.0	789.2	0.0	241.9	0.0	0.0	16.7
C_u_L_10_2	250139.0	0.0	1.0	0.0	480.3	0.0	151.9	0.0	0.0	21.4
C_n_L_10_1	250167.2	0.0	2.4	0.0	224.9	0.0	105.2	0.0	0.0	14.8
C_n_L_10_2	250181.8	0.0	1.9	0.0	2139.4	0.0	334.0	0.0	0.0	27.0
RC_u_L_10_1	250180.7	0.0	4.2	0.0	1248.2	0.0	271.3	0.0	0.0	14.3
RC_u_L_10_2	250159.0	0.0	5.1	0.0	1197.1	0.0	352.4	0.0	0.0	30.4
RC_n_L_10_1	250155.7	0.0	3.2	0.0	281.5	0.0	174.5	0.0	0.0	14.6
RC_n_L_10_2	250174.7	0.0	2.1	0.0	1018.8	0.0	147.8	0.0	0.0	24.4
R_u_S_10_1	730244.1	0.0	4.4	0.0	1911.0	0.0	494.5	0.0	0.0	16.3
R_u_S_10_2	686878.5	0.0	4.7	0.0	9008.9	0.0	1721.5	0.0	0.0	33.9
R_n_S_10_1	788669.3	0.0	7.1	0.0	9435.6	0.0	1089.4	0.0	0.0	17.3
R_n_S_10_2	737856.4	0.0	1.5	0.0	671.5	0.0	194.2	0.0	0.0	23.6
C_u_S_10_1	791586.6	0.0	3.0	0.0	1594.1	0.0	386.4	0.0	0.0	14.9
C_u_S_10_2	666935.7	0.0	0.9	0.0	472.4	0.0	149.3	0.0	0.0	23.3
C_n_S_10_1	751561.4	0.0	2.1	0.0	595.7	0.0	150.5	0.0	0.0	15.5
C_n_S_10_2	795264.1	0.0	2.0	0.0	2288.5	0.0	371.7	0.0	0.0	25.6
RC_u_S_10_1	792066.5	0.0	4.0	0.0	2297.8	0.0	418.6	0.0	0.0	13.9
RC_u_S_10_2	727048.0	0.0	5.6	0.0	1560.1	0.0	411.7	0.0	0.0	29.9
RC_n_S_10_1	708776.3	0.0	3.3	0.0	829.3	0.0	268.8	1.2	1.2	14.6
RC_n_S_10_2	774116.4	0.0	2.4	0.0	1070.9	0.0	239.4	0.0	0.0	26.8

For small instances, we observe that TLAP, SGC and PD always find optimal solutions. EA also manages to find optimal solution for most instances. Since optimal solution can be found for all instances, we compare the solution times. Due to the size of the instances, TLAP is able to find an optimal solution in very short time, while SGC and PD struggles for convergence and sometimes takes more than one hour to complete. EA finds an optimal solution again in short time, but not as short as TLAP. The average solution time for TLAP is 3.2 seconds, while SGC and PD take 2009.3 418.3 seconds, respectively. EA converges in 21.3 seconds on average.

For this set of instances, we find that TLAP offers the best performance in terms of solution time. Nevertheless, EA should be considered, too, since it offers alternative solutions that may be useful for the decision maker.

In Table 6.2, we provide the results for the medium instances.

Table 6.2: Computational Results for Medium Instances

Results for Medium Instances	Optimal Solution	TLAP		SGC		PD		EA		
		GAP%	CPU Time (sec)	GAP%	CPU Time (sec)	GAP%	CPU Time (sec)	Best Gap from Optimal%	Avg Gap from Optimal%	Avg CPU Time (sec)
R_u_L_100_1	251602.1	0.0	9706.6	0.1	*	0.2	*	0.0	0.0	156.8
R_u_L_100_2	251557.4	0.0	6954.7	0.1	*	0.1	*	0.0	0.0	236.1
R_n_L_100_1	251744.3	0.0	7538.3	0.1	*	0.1	*	0.0	0.0	137.6
R_n_L_100_2	251700.0	0.0	9328.1	0.1	*	0.1	*	0.0	0.0	337.8
C_u_L_100_1	251551.2	0.0	6305.9	0.0	*	0.0	*	0.0	0.0	126.3
C_u_L_100_2	251653.3	0.0	5482.6	0.0	*	0.1	*	0.0	0.0	188.7
C_n_L_100_1	251560.9	0.0	5139.4	0.0	*	0.1	*	0.0	0.0	139.9
C_n_L_100_2	251702.5	0.0	6060.7	0.1	*	0.1	*	0.0	0.0	232.7
RC_u_L_100_1	251455.0	0.0	6361.9	0.1	*	0.1	*	0.0	0.0	139.2
RC_u_L_100_2	251619.4	0.0	5923.0	0.1	*	0.1	*	0.0	0.0	341.6
RC_n_L_100_1	251677.3	0.0	5553.7	0.0	*	0.0	*	0.0	0.0	122.9
RC_n_L_100_2	251460.3	0.0	6751.9	0.1	*	0.1	*	0.0	0.0	216.0
R_u_S_100_1	730624.5	0.0	1754.9	16.1	*	19.2	*	0.0	0.1	173.6
R_u_S_100_2	717207.5	0.0	1563.3	10.6	*	13.1	*	0.0	0.0	232.7
R_n_S_100_1	773292.2	0.0	1994.9	11.7	*	12.3	*	0.0	0.0	149.7
R_n_S_100_2	760011.0	0.0	1926.9	9.5	*	11.2	*	0.0	0.0	329.0
C_u_S_100_1	715369.3	0.0	436.7	3.5	*	3.4	*	0.0	0.0	145.8
C_u_S_100_2	745984.6	0.0	1148.3	4.9	*	4.8	*	0.0	0.0	188.9
C_n_S_100_1	718274.1	0.0	769.0	6.1	*	6.6	*	0.0	0.0	124.5
C_n_S_100_2	760758.1	0.0	1265.3	6.8	*	7.3	*	0.0	0.0	270.7
RC_u_S_100_1	686507.6	0.0	769.0	9.8	*	12.5	*	0.0	0.0	145.0
RC_u_S_100_2	735827.8	0.0	1145.2	8.9	*	10.5	*	0.0	0.0	340.2
RC_n_S_100_1	753200.8	0.0	857.8	4.1	*	4.1	*	0.0	0.0	133.2
RC_n_S_100_2	688074.3	0.0	1090.5	9.0	*	11.3	*	0.0	0.0	213.4

* Time Limit (4 hours) Exceeded

SGC and PD do not converge within the time limit for any of the instances. For EA, best and average gaps are 0% for all instances except R_u_S_100, whose average gap is equal to 0.1%.

For medium instances, we observe that TLAP and EA always find the optimal. SGC and PD find the optimal solution for most instances as their best integer solutions, but they cannot converge within the time limit. They leave very small gaps when the ratio of fixed cost over transportation cost is large and large gaps when the opposite is true. EA misses the optimal only in one replication for one instance.

Regarding the solution times, we reach a different conclusion than the previous set of instances. While TLAP has the advantage of proving optimality, it takes much longer to complete compared to the EA. EA finds the optimal in shorter times and missed optimality in only 1 replication among 120. Thus it can also be considered a reliable method. For this set of instances, we find that TLAP offers proof of optimality, while EA offers solution in short time as well as alternative solutions for the decision maker.

The average solution times for TLAP and EA are 3992.9 and 200.9 seconds, respectively. SGC and PD keep running until the time limit of four hours is exceeded.

In Table 6.3, we provide the results for the large instances.

Since we do not have the optimal solutions of large instances for comparison, this time we cannot provide gap values. Instead, we make a comparison between best solutions found by exact methods (BSEM) and EA.

EA: $Best\ Gap'\% = \frac{Best\ of\ Repl - BSEM}{BSEM} * 100$

EA: $Ave\ Gap'\% = \frac{Avg\ of\ Repl - BSEM}{BSEM} * 100$

Table 6.3: Computational Results for Large Instances

Results for Large Instances	Best Known Soln. from Ex. M.	TLAP		SGC		PD		EA		
		GAP%	CPU Time (sec)	GAP%	CPU Time (sec)	GAP%	CPU Time (sec)	Best Gap from Best Exact%	Avg Gap from Best Exact%	Avg CPU Time (sec)
R_u_L_1000_1	265593.4	**	*	1.7	*	1.6	*	0.0	0.0	6463.9
R_u_L_1000_2	265595.6	**	*	1.6	*	1.5	*	0.0	0.0	6345.0
R_n_L_1000_1	267840.4	**	*	1.2	*	1.0	*	0.0	0.0	7093.3
R_n_L_1000_2	266886.5	**	*	1.4	*	1.1	*	0.0	0.0	8454.9
C_u_L_1000_1	265947.9	**	*	1.0	*	0.9	*	0.0	0.0	4594.1
C_u_L_1000_2	265442.7	**	*	0.9	*	0.9	*	0.0	0.0	4694.6
C_n_L_1000_1	267120.0	**	*	1.2	*	1.0	*	0.0	0.2	5138.8
C_n_L_1000_2	265864.7	**	*	0.9	*	0.7	*	0.0	0.0	4336.9
RC_u_L_1000_1	266456.9	**	*	1.3	*	1.2	*	0.0	0.0	6052.1
RC_u_L_1000_2	265511.1	**	*	1.3	*	1.2	*	0.0	0.0	6175.2
RC_n_L_1000_1	265920.5	**	*	1.4	*	1.2	*	0.0	0.0	6651.8
RC_n_L_1000_2	267154.6	**	*	1.0	*	0.8	*	0.0	0.0	4855.6
R_u_S_1000_1	717802.7	**	*	22.8	*	21.2	*	0.0	0.0	7246.1
R_u_S_1000_2	717867.8	**	*	20.7	*	19.3	*	0.0	0.0	8238.7
R_n_S_1000_1	785211.0	**	*	13.8	*	10.6	*	0.0	0.0	7444.5
R_n_S_1000_2	757366.8	**	*	16.6	*	12.9	*	-0.1	-0.1	8373.3
C_u_S_1000_1	728436.7	**	*	11.8	*	11.1	*	0.0	0.0	4853.7
C_u_S_1000_2	713279.5	**	*	10.7	*	10.2	*	0.0	0.1	4346.1
C_n_S_1000_1	763599.7	**	*	14.8	*	12.1	*	0.0	0.0	4941.7
C_n_S_1000_2	725942.1	**	*	10.5	*	8.4	*	0.0	0.0	4306.0
RC_u_S_1000_1	743705.8	**	*	17.1	*	14.3	*	0.0	0.0	5523.9
RC_u_S_1000_2	715332.6	**	*	16.9	*	15.6	*	0.0	0.0	6475.4
RC_n_S_1000_1	727614.1	**	*	17.4	*	14.9	*	0.0	0.0	7441.2
RC_n_S_1000_2	764637.5	**	*	11.4	*	9.3	*	0.0	0.1	5081.3

* Time Limit (4 hours) Exceeded

** No Feasible Solutions

SGC and PD do not converge within the time limit for any of the instances. For EA, best and average gaps are close to 0% for all instances.

We observe that TLAP does not find any feasible solutions within time limit for large instances (When the time limit was increased to 48 hours, TLAP still left gap around 30%.). SGC and PD do not converge for any of the instances either. However, they leave small gaps for instances with large ratio and large gaps for others. EA converges within time limit, but it sometimes takes more than two hours to complete.

We see that for 20 instances out of 24, the best solutions found by the exact methods have the same objective function value as the best solution found in all of the replications of EA. Since these methods use completely different approaches for convergence, this is an important result signaling the probability of these solutions being optimal. For the remaining 4 instances, both SGC and PD find the best solution in one case, and EA finds the best solution in three cases.

In conclusion, we observe that TLAP obtains optimal results for small and medium instances. With small instances it reaches optimum in very short time, but with medium instances it takes much longer. TLAP cannot produce any feasible solutions for large instances within the time limit. SGC and PD are able to converge only for small instances, but they find the optimal solution most of the time as its best integer solution. EA is very successful in finding good solutions, and always converges within time limit. It offers the fastest solution times in medium and large instances and loses to TLAP only with small instances.

In the light of these observations, we may suggest the decision maker to use standard solvers when dealing with small instances. If obtaining alternative solutions is desirable for the decision maker, EA may also be used despite a small increase in solution time.

For medium instances, EA offers the best performance among the three methods. It also produces alternative solutions, which can be compared with respect to other (secondary) objectives by the decision maker. However, since EA does not prove optimality, TLAP may also be used according to the preference of the decision maker. Since this is a strategic decision making process, solution times may be considered to be much less important.

In real life application, we expect the problem to include hundreds of customers, therefore finding good solutions for large instances is critical. For solving large instances, we only have two alternatives, since TLAP fails to produce any feasible solutions. We suggest using EA for these instances, since the algorithm converges and finds the best known solutions. While the best solutions of the remaining methods agree most of the time, the decision maker may also prefer to apply both methods and compare their results.

When we compare the instance sets, we also observe several disparities. A clear difference can be seen between the performance of SGC and PD on large ratio and small ratio instances. For small instances, SGC and PD reach convergence for all instances, but small ratio instances take considerably longer to be solved. For SGC, the average time it takes until convergence is 1374.0 seconds for large ratio instances, but 2644.7 seconds for small ratio instances. For PD, the figures are 345.3 and 491.3 seconds, respectively. It is safe to argue that small ratio instances are harder to solve, since the algorithm has more incentive to look for solutions with more open CDCs with the hope of reducing total variable cost. This is not possible when fixed costs are much larger than variable costs. We do not have the convergence times for medium and large instances, but gap values indicate that the same argument can be made. Gap values are larger for small ratio instances mainly due to the larger share of variable costs in the objective function value. When fixed cost occupies a larger share and the optimal number of CDCs can be found in the early iterations, we obtain small gap values, but this is not the case with small ratio instances.

The solution time differences are not as clear with EA. Since EA only evaluates sets of locations decisions (chromosomes) instead of trying to come up with an optimal decision set, the algorithm is not affected a lot from the differences in instances. For small instances, the average time until completion is 21.2 and 21.3 seconds for large and small ratio instances, respectively. For medium instances, the figures are 198.0 and 203.9. For large instances, they are 5904.7 and 6189.3.

With TLAP, we observe similar results to SGC and PD for small instances. The average solution time for large ratio instances is 3.1 seconds, while it is 3.4 seconds for small ratio instances. We observe an interesting difference for medium instances. This time the average solution time is 6758.9 seconds and 1226.8 seconds respectively. For large instances, we do not have the data for such comparison.

When we compare the instances with respect to distribution, we find that instances with normally distributed customers seem to be harder to solve with all three methods. This may be due to the fact that the customer density in the city center is higher in those instances. When this is the case, there are fewer customers that can be assigned to the closest open CDC with confidence, because many customers are closely packed and swapping their assignments may bring benefit. This phenomenon can be observed most clearly in the hybrid evaluation algorithm of EA. As explained in Chapter 5, the algorithm starts by assigning all customers to the closes available CDC, but then employs a 2-opt procedure to swap the assignments. The swaps are made only when assignment distance is above some predetermined range value. This approach is effective in reducing the solution time, but fails to find good solutions in short time when customer density is high in the city center. In a similar fashion, exact solution methods encounter many alternative paths to take when it comes to assigning the customers in the city center. This certainly makes it harder to solve the instances with customers concentrated in the center.

More detailed data on the experiments can be found in the Appendix.

CHAPTER 7

ROUTING DECISIONS: POSSIBLE SOLUTION APPROACHES

So far, we have focused on the strategic decisions of the City Logistics system. As explained in section 4, we decomposed the decisions with respect to their strategic or operational nature and considered only the strategic level decisions in our solution approaches. The main idea was that if these strategic decisions are made near optimally, the lower level decisions could be made accordingly and the overall system would work efficiently.

7.1 Making Lower Level Decisions

Once the higher level decisions are made, the remaining decisions, namely fleet sizing and routing, can also be made optimally. As explained in section 4, the solution space for these decisions are directly determined by the higher level decisions. More importantly, as the higher level decisions are fixed, the remaining ones can be separated with respect to scenarios and to open CDCs.

Previously, we needed to combine the location decisions with scenario based allocation decisions in order to produce meaningful solutions. However, this is not the case for the lower level, since routing decisions for different scenarios are completely unrelated to each other. Each scenario can be solved on its own. A similar observation can be made for each open CDC. Each set of customers assigned to a CDC constitutes a separate TSP or VRP problem depending on the number of vehicles that will be used for delivery.

There is an extensive literature on this kind of the problem. In our case, where the customers may have delivery time preferences, the city administration may have delivery time restrictions and balancing workload (both in terms of tour length and delivered amount) between vehicles may be of concern and several extensions of the VRP literature are of interest. For example, VRP with time-windows (VRPTW), capacitated VRP (CVRP), multi-trip VRP (MTVRP), fleet size and mix VRP, multi-commodity VRP (MCVRP) and their combinations are extensively studied in the literature (Note that there are fewer studies considering these problems under uncertainty.). Each of these problems relax one assumption of the classical vehicle routing problem.

We mention the following example based on VRPTW to give an idea of the capabilities of methods used in the literature. Generally, the state-of-the-art methods proposed in the VRPTW literature are sufficient for making the lower level decisions in our case. Since we separate the problem into smaller pieces by fixing the higher level decisions, it is possible to reach optimal even with large instances. In this process, we create 5-6 single depot VRPTW problems with about 160-200 customers for each scenario and instances of this size can be solved efficiently with the methods reviewed above. It is also important to note that the decisions maker does not have to solve for all scenarios, since only the decisions related to the current scenario are relevant for practice. As the conditions change, the lower level decisions can be reoptimized periodically.

In conclusion, we propose the methods for making strategic level decisions in the CL system and we suggest the decision makers to use the state-of-the-art methods for the selected VRP extensions for the remaining decisions. Once the higher level decisions are fixed, the decision maker has the advantage of considering only the scenario of interest and remake the computations only when necessary. At that stage, it is also possible to obtain the lower level decisions optimally for all scenarios and make simple adjustments on the higher level decisions accordingly. The decision making process is simplified a lot with this approach.

7.2 Approximating Lower Level Costs

An important point to note is that, by not considering the lower level decisions, like routing, we make a simplifying assumption that the distance between customers and CDCs is a good estimator of the additional cost of serving the customer. In real life, this cost would be calculated for a group of customers that share a delivery route, instead of for individual customers. The only way to avoid this assumption is to make all location, allocation, fleet sizing and routing decisions at the same time. This is not possible for our case, since uncertainty is also taken into account. A possible way of incorporating the lower level decisions into the strategic decision making is approximating the related costs instead of trying to find them optimally.

Due to the NP-hard nature of the VRP and its appearance in larger problems as in our case, a large number of studies has been made in order to accurately estimate the routing cost. All these studies suggest methods to estimate routing cost without explicitly making routing decisions. We investigate three of these methods and try to propose a method suitable for use in our case. Since the lower level decisions cannot be made optimally, it is better to consider these methods within metaheuristics, possibly as a module of the evolutionary algorithm explained in section 7.

7.2.1 Tour Length Approximation

Tour Length Approximation is a method of estimating route length without explicitly determining the route. For estimation, the inputs usually required are the distance metric used, spatial distribution of customers, the size of the region, number of customers in the region, number of customers to be visited, and average distance of customers to the warehouse. Approximation methods use equations to estimate the length of the tour with these specified characteristics.

The idea of estimating the length of tours with an analytical perspective started with Beardwood et al. (1959). Afterwards, Tour Length Approximation has been used in many studies, mostly in the 1980s and 1990s. Many studies on TSP, VRP and LRP used tour length approximation to avoid the complexity due to routing decisions. Methods to estimate the tour length gradually become more complex and problem-specific. Other important contributors to the literature on this subject are Christofides and Eilon (1969), Daganzo (1984a), Daganzo (1984b), Hall (1984), Castillo, J.M. (1999), Figliozzi (2009). Campbell et al. (1996) provide a review of most of the relevant studies.

Tour length approximation is a good method to estimate tour length when there is little information available about the customers' specific locations. However, in a city logistics system, customers' exact locations are known. We can use heuristics to compute better approximations using exact locations. Moreover, tour length approximation requires triangular inequality to hold. Due to the transportation cost uncertainty, triangular inequality does not always hold in our instances. In addition, even if all the required inputs are available, calculating the approximation might be challenging. For instance, equations might include integral calculations. Lastly, tour length approximation does not include the time dimension of tours. In a city logistics system, length of a tour is not the only cost item; penalty costs of violating time windows could also be considered. Therefore, when we need to take into account the time-windows, tour length approximation is not a suitable approach.

7.2.2 Heuristics

There are two main advantages of heuristics over the other approximation methods. Heuristics allow us to use the known information on an instance effectively. For instance, we can use the transportation costs among a couple of nodes which provide precise computations compared to other methods. Second, heuristics are more flexible. With a heuristic algorithm, we can consider different objectives concurrently, add or remove algorithm modules to fit our purpose. We demonstrate these advantages later in this subsection.

There are many heuristics and metaheuristics proposed in the literature for VRPTW. One of the most successful ones for solving the large scale VRPTW is proposed by Mester and Braysy (2005). This method employs guided evolution strategies and finds good solutions for instances with up to 1000 nodes. For instances with 100/200/400 nodes, CPU time requirement is around 80 seconds/8 minutes/17 minutes on average. If we used this method for the routing decisions, we would need to repeat it for each CDC and each scenario (around 500 times).

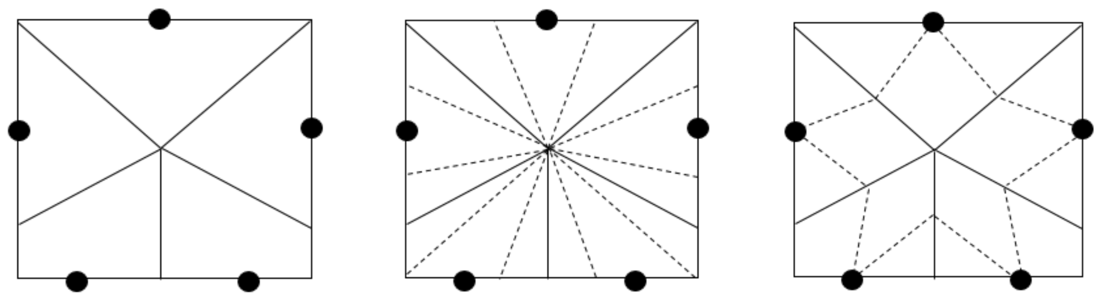
We use a different approach that aims to achieve sufficiently good results using shorter CPU time. First we decide which one of the two approaches should be used: route-first-cluster-second or cluster-first-route-second.

Since the number of nodes allocated to a particular CDC may be in the hundreds, routing first would require significantly more computation. Since we are using heuristics for the routing decisions, having a very large number of nodes to be routed would badly deteriorate the solution performance. It is logical to expect better routing performance when there are fewer nodes to be considered. Thus, by using cluster-first-route-second approach, we reduce the number of nodes to be routed by a large margin, since each time we have to consider only the nodes within a particular cluster.

Dondo and Cerda (2007) propose such a method. It starts by clustering customers. Then the clusters are assigned to depots. Next sub-clusters are constructed within the clusters and TSPTW is solved for each sub-cluster with heuristics. The method solves 25-node instances well, but struggles with larger instances.

During the clustering process, we have to take into account two parameters: the width of the time window and the capacity of vehicles. If the vehicle capacity is considered as a hard constraint (which is the case in real life) and time window is considered as a soft constraint (since violating time windows may be penalized but not forbidden in real life), vehicle capacity constraint is preemptive over time window constraint.

We first try to find the smallest number of vehicles (we call this number *k*) that can serve the set of nodes without violating the capacity constraint. Then, we create *k* clusters over the area allocated to the CDC of current interest. Now, we need to determine how to create the clusters in a meaningful way. The following explanations aim to find a meaningful way of partitioning regions. The results may be an initial solution for an algorithm that improves the clusters and routes step by step to reduce the difference between tour lengths.



(a) Customer assignment (b) Clusters in sectors-1 (c) Clusters in sectors-2

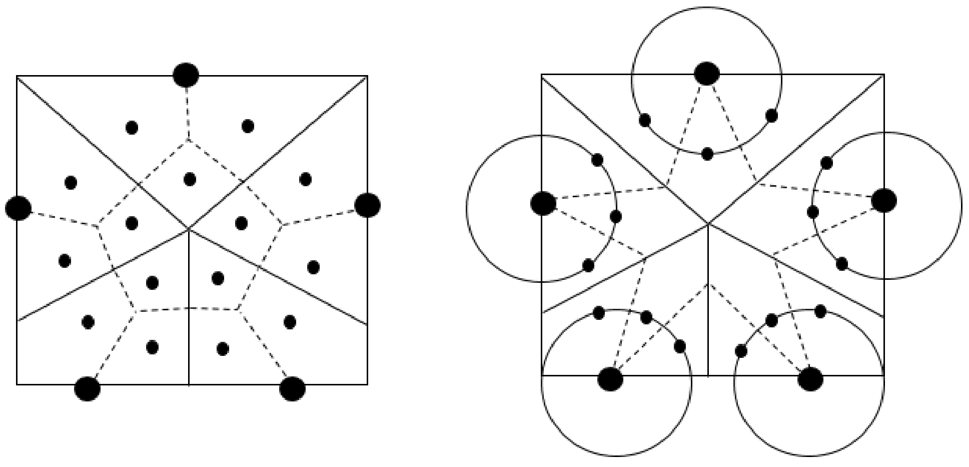
Figure 7.1: Clustering Approaches when CDCs are Located on City Boundaries

When we are dealing with square cities, the allocation of customers would partition the city area roughly in a way that is depicted in Figure 7.1 (a).

If the CDCs are near the center of the city, an intuitive way of partitioning the regions allocated to each CDC would look like the one given in Figure 7.1 (b).

But the CDCs being on the city boundaries, this setting obtains larger tours for the subregions that do not contain the CDCs. In other words, even if the total route length within each subregion is equal, the distance for reaching the subregions are significantly different. We may use a scheme to partition the regions as shown in Figure 7.1 (c) to solve this problem.

If we assume the city center to contain a dense population of customers, it is easy to change the boundaries separating subregions, so that demand constraint can be satisfied. To do that, we may start assigning the customers from the ones that would incur the highest cost if they are not assigned to their closest cluster center and continue until that cluster's capacity is filled. Such an approach is expected to give the partitioning in Figure 7.2 (with cluster centers marked with dots):



(a) Clusters within sections (b) Approx. locations of cluster centers

Figure 7.2: Cluster Centers when CDCs are Located on City Boundaries

This setting assigns larger areas to subregions on the sides, thus obtains longer tour lengths for them, while reducing the tour length of the central subregion. The following setting that tries to find larger central subregions and smaller side-subregions may be an alternative, but does not solve the problem completely. To preserve the star-shaped partitioning, we have to select cluster centers a certain distance (*r*) away from the CDC.

Cluster centers may be found in a usual way, but then changed with points *r* units away from the CDC to the same direction (angle).

When we are dealing with circular cities, the allocation of customers would partition the regions in the way given in Figure 7.3:

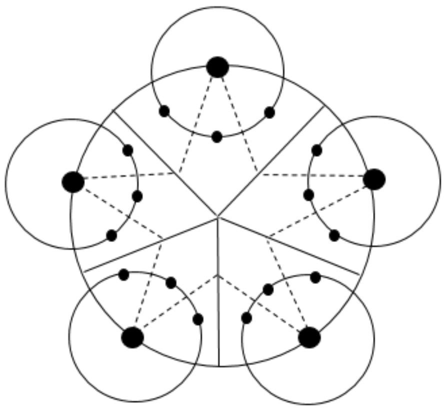


Figure 7.3: Cluster Centers in a Circular City

Such a partition guarantees feasibility of demand constraint and it gives us subregions with close tour length values. At this stage, we may take into account the TWs and try to reduce the tours that violate the time window, as much as possible. If we cannot make all tours shorter than the TW width, even after we obtain equal tour lengths for all, we have two alternatives.

If we violate the TW constraint too much, we may introduce an additional vehicle and restart the process. However, if this is not the case, we may introduce the extra vehicle according to a comparison between TW penalty incurring with the current solution and vehicle operating cost.

The steps of the heuristic algorithm are as follows:

Algorithm 7.1 Vehicle Routing Stage

Input: Customer assignments, Vehicle capacity, TW width and TW violation penalty, Vehicle operating cost

Output: New population

```
 1: for each CDC do
 2:     let S be the set of customers assigned to this CDC
 3:     $|V| \leftarrow \lceil \frac{\sum_{j \in S} dem_j}{VehCap} \rceil$
 4:     select $|V|$ cluster centers
 5:     divide $S$ into cluster subsets: $S = C_1 \cup C_2 \cup \ldots \cup C_{|V|}$
 6:     $Z_{C_v} \leftarrow$ solve TSP for each subset $C_v$
 7:     while $\exists v$: $\sum_{j \in C_v} dem_j > VehCap$ do
 8:         let $C_{v_1}$ and $C_{v_2}$ be the clusters with largest and smallest demand
 9:         for each $j \in C_{v_1}$ do
10:             $Z'_{C_{v_2}} \leftarrow$ solve TSP for $C_{v_2} \cup j$
11:             $Z'_{C_{v_1}} \leftarrow$ solve TSP for $C_{v_1} \setminus j$
12:             $Cost_j \leftarrow Z'_{C_{v_2}} + Z'_{C_{v_1}} - Z_{C_{v_2}} - Z_{C_{v_1}}$
13:         end for
14:         let $j$ be the customer with least $Cost_j$
15:         $C_{v_1} \leftarrow C_{v_1} \setminus j$
16:         $C_{v_2} \leftarrow C_{v_2} \cup j$
17:     end while
18:     while $\exists v$: $Z_{C_v} > TW$ do
19:         let $C_{v_1}$ and $C_{v_2}$ be the clusters with longest and shortest tour
20:         if [$Z_{C_{v_1}} > TW$ or $\sum_{k \in C_v \cup j} dem_k > VehCap \, \forall j \in C_{v_1}$]
                and $Penalty * \sum_v (Z_{C_v} - TW) > VehCost$ do
21:             $|V| \leftarrow |V| + 1$
22:             start again by selecting $|V|$ cluster centers (step 4)
23:         else if $Z_{C_{v_1}} > TW$ or $\sum_{k \in C_v \cup j} dem_k > VehCap \, \forall j \in C_{v_1}$
24:             break
25:         else
```

```
26:            for each $j \in C_{v_1}: \sum_{k \in C_v \cup j} dem_k < VehCap$ do
27:                $Z'_{C_{v_2}} \leftarrow$ solve TSP for $C_{v_2} \cup j$
28:                $Z'_{C_{v_1}} \leftarrow$ solve TSP for $C_{v_1} \setminus j$
29:                $Cost_j \leftarrow Z'_{C_{v_2}} + Z'_{C_{v_1}} - Z_{C_{v_2}} - Z_{C_{v_1}}$
30:            end for
31:            let $j$ be the customer with least $Cost_j$
32:            $C_{v_1} \leftarrow C_{v_1} \setminus j$
33:            $C_{v_2} \leftarrow C_{v_2} \cup j$
34:        end if
35:    end while
36: end for
```

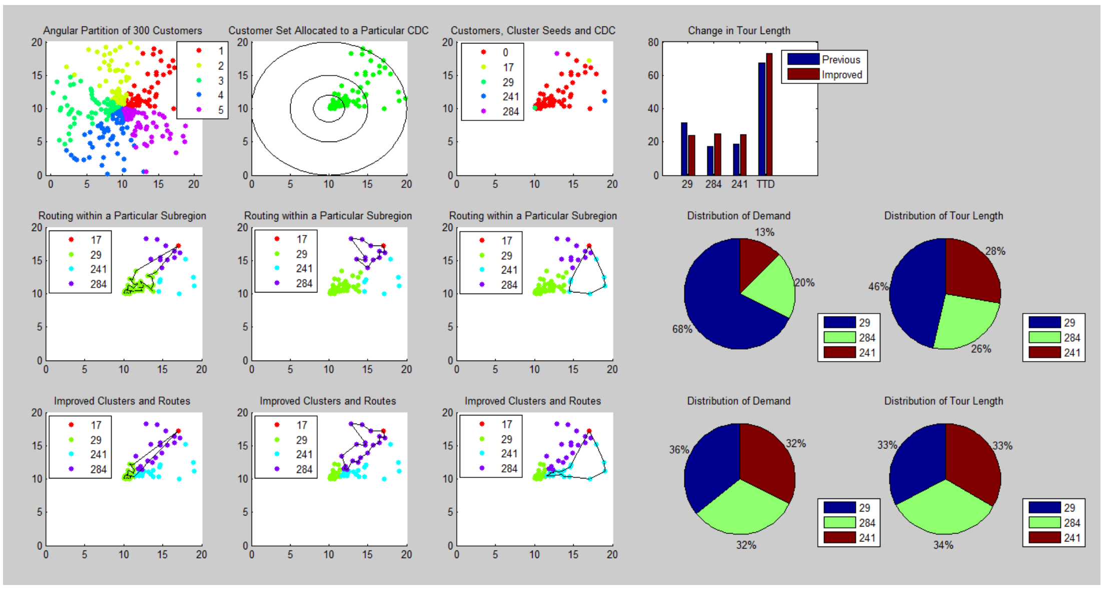


Figure 7.4: Output of the Algorithm

Figure 7.4 illustrates the effect of the improvement heuristic.

- Top row:
    - First figure: Customer allocations in a 300-customer instance
    - Second figure: Customers that will be considered in the current run
    - Third figure: Customers corresponding to CDC and the customers used as cluster centers

    - Fourth figure: Tour lengths (for each cluster and in total) before and after the improvement heuristic
- Middle row:
    - First, second, and third figures: Delivery routes in respective clusters
    - Fourth figure: Distribution of demand across clusters
    - Fifth figure: Distribution of tour length across clusters
- Bottom row:
    - First, second and third figures: Delivery routes in respective clusters after improvement
    - Fourth figure: Distribution of demand across clusters after improvement
    - Fifth figure: Distribution of tour length across clusters after improvement

Similar results have been observed with different instances. Thus the ones seen in the figure show the typical behavior of the algorithm. The most important observation is that, as explained before, most customers are assigned to the cluster closest to the city center. As a result of this, the delivery route of the central cluster is longer.

The demand and tour length differences between clusters have two implications for the system. When demand is concentrated in one cluster, it is a clear indicator that the vehicle capacities in other clusters are not used efficiently. Also, there is a higher chance of the current solution exceeding the vehicle capacity in the central cluster. Therefore, it is desirable to construct clusters with close total demand values. When the tour length of one cluster is significantly larger than others, it can be inferred that this tour will be more susceptible to be affected by changes in traffic conditions. In other words, vehicles serving clusters in short tours afford to allocate some of their time as a buffer against unexpected circumstances during delivery. However, other vehicles do not have that luxury. Therefore, balancing the tour length as well as demand is a desirable objective in this problem.

The pie charts in Figure 7.4 show the distribution of demand and tour length across clusters. Before the improvement algorithm, we observe that 68% of demand and 46% of total tour length are in the central cluster. After the improvement, we observe that all clusters share the demand almost equally and their tour lengths are also very close.

As can be seen in the bar chart, the tour length of the central cluster decreased while tour length of corner clusters increased. For this instance, the total tour length increased by 8.3%. This increase in total tour length is the price of achieving almost perfect equity between clusters. Note that achieving perfect equity is not possible, since there are two distinct equity objectives. The balance between equity and tour length may be struck on different points within the feasible region. These different points can be found by changing the algorithm parameters. Considering these decisions are effective on the operational level, selection of algorithm parameters is left to the decision maker.

The computational efficiency of the algorithm is an important consideration. Figure 7.4 above illustrates only a small part of the problem, a group of customers assigned to one of the open CDCs in a single scenario among a hundred. To embed the algorithm within a metaheuristic, we need to obtain fast results. The algorithm takes on average 5.5 seconds and 2.3 seconds with and without the improvement part respectively, for each set of location decisions. If all scenarios are to be evaluated, the time requirement would be prohibitive. However, if only the scenarios in a representative group are to be evaluated, the process would also take much shorter, while maintaining sufficient accuracy.

CHAPTER 8

DISCUSSION AND COMMENTS

In this thesis, we pursued three main objectives. The first objective is to analyze the City Logistics literature in terms of its relations and overlaps with other fields in the relevant OR literature. We have seen that there are several well-known OR/IE problems which are strongly related to CL. In fact, the whole set of decisions to be made in a CL system can be decomposed into subproblems and there are studies considering similar problems in the literature. We have seen that facility location, fleet management, vehicle routing problems and several of their combinations have been studied extensively in a deterministic setting and less frequently under uncertainty.

The relevant literature fails to offer adequate modeling and solution methods due to two shortcomings. CL systems have several characteristics that need to be taken into account, especially the CDC locations and the significance of transportation cost uncertainty. The effect of these characteristics makes it crucial for the decision makers to use methods that are tailored for the CL setting. We provide the methods for the CL system from a strategic view.

Modeling all required decisions together may be desirable to achieve a holistic view of the system. However, the computational complexity of such a model prohibits the use of any exact solution method. With the current computer technology, it is certainly not possible to find optimal solutions for all these decisions for the instances of realistic size. Thus, there remains two options: optimizing the higher level decisions and solving for the lower level decisions when necessary, or considering all decisions in a heuristic manner. The former is a natural path that follows the strategic view of the problem, but we considered the latter. While we did not fully implement the algorithm and obtain computational results, we provide a sample algorithm that outlines the important points to be considered when the fleet management and routing decisions are made.

The second objective is to analyze the value of information in a CL setting. Due to the dynamic nature of cities, CL systems are operating in an ever-changing environment. This situation makes it crucial for the system to be adaptable to the changes in parameters from an optimization perspective. Having a strategic view towards the problem, we see that the location decisions are not subject to change in the short-term, but it is certainly possible for the allocation decisions to be updated regularly.

We measured the benefit brought by changeable allocation decisions and found that a large amount of cost reduction is possible. Though the amount of this reduction depends on the spatial distribution of customers and shape of the city, it is always significant. Moreover, the reduction is larger in the CL setting than in the classical facility location setting. For the sake of updating allocation decisions, the CL system only has to deal with the necessity to coordinate customers.

The third objective of this thesis is to create efficient solution methods that are able to deal with instances of realistic size. While standard commercial MIP solvers can solve small instances very quickly and medium instances in long time, they cannot produce even any feasible solutions for large instances. Since any instance of realistic size would fall into the large group, these standard solvers cannot be considered for use in a real-life application.

We propose (in)exact solution methods for solving the problem in a realistic setting. First one is an enhancement of the L-Shaped method with scenario-group cuts. Due to the way we construct scenarios, the use of scenario-group cuts is natural to our solution approach and does not require any additional algorithm to construct scenario groups. This enhancement proves to be effective in striking a balance between the computational cost of adding multiple cuts and collective weakness of adding single cuts. While it does not converge within time limit, the best upper bound found by the algorithm is usually optimal.

The second method we proposed is an evolutionary algorithm. Parallel to the approach we followed so far, we separate location and allocation decisions from each other and handle only the location decisions explicitly. On the other hand, allocation decisions found with a separate decoding/evaluation algorithm. It is a hybrid algorithm using greedy and 2-opt heuristics that constructs a feasible solution and improves it. We tailored this algorithm for the CL setting, using the unconventional CDC locations to our advantage. Both methods are able to obtain good solutions for instances of realistic size and they find the same solutions most of the time.

In this context, there are several future research directions that we identified. The first one is the addition of lower level decisions to the optimization model. We believe that more sophisticated decomposition methods would be necessary to achieve this, and solving large instances optimally may not be possible even with such methods. A suitable starting point may be a three-stage formulation with routing decisions on the last stage, extending on our two-stage stochastic formulation. On the other hand, it seems possible to consider these decisions within a metaheuristic algorithm, as suggested in Chapter 7. Still we would need a slightly faster way to evaluate routing decisions, so that large instances can be solved.

Future studies may also consider other factors under uncertainty. After transportation cost uncertainty that affects the whole system all the time, demand uncertainty may be an important factor especially for lower level decisions. It is quite possible that some customers have less or zero demand in some periods, thus the decision-maker may prefer operating a smaller fleet of delivery vehicles than what would be required by a scenario. The change in routing decisions would be less significant in such a case, but we believe that a model that involves fleet management decisions would work best under demand uncertainty.

While we tried our best to create efficient solution methods, it is also possible to apply different methods on the problem, so that larger instances can be solved. Making further problem-specific enhancements on the L-Shaped method and implementing different operators within the evolutionary algorithm are two possibilities.

A future study may also generalize our results to multi-echelon case, most importantly to the two-echelon setting. In this study, we assumed that the CDCs are always located on the city boundaries and they directly serve the customers with the help of delivery vehicles. If this does not have to be the case in a CL network since, in a large city, a second level of facilities, called satellites, need to be located inside the city as an intermediate level between CDCs and customers. When satellites are used, CDCs supply the satellites and satellites serve customers. This network configuration is described as two-echelon and leads to a slightly complicated model. The ideas in this thesis can be modified for use in the two-echelon setting.

Although CL is a relatively a new research area, it is gaining momentum due to CL projects currently under development. The number of studies that consider CL is increasing rapidly both in the OR/IE literature and other fields. We expect this trend to continue until the obstacles we mentioned above can be overcome. Modeling the CL system with fewer assumptions and using more efficient solution methods have the potential to bring large benefits in real-life applications. We believe CL offers important solutions to the every-day problems of urban life and the scientific community has the responsibility and capability to help with the effective implementation of these solutions.

REFERENCES

1. Adulyasak, Y., Cordeau, J.-F. and Jans, R., 2015a. Benders decomposition for production routing under demand uncertainty. Operations Research, Articles in Advance, 1-17.

2. Adulyasak, Y., Cordeau, J.-F., Jans, R., 2015b. The production routing problem: a review of formulations and solution algorithms. Computers & Operations Research 55, 141-152.

3. Albareda-Sambola, M., Fernandez, E. and Laporte, G., 2007. Heuristic and lower bound for a stochastic location-routing problem. European Journal of Operational Research 179(3), 940-955.

4. Altınel, İ.K., Durmaz, E., Aras, N. and Özkısacık, K.C., 2009. A location–allocation heuristic for the capacitated multi-facility Weber problem with probabilistic customer locations. European Journal of Operational Research, 198(3), 790–799.

5. Alumur, S., Nickel, S., & Saldanha-da-Gama, F., 2012. Hub location under uncertainty. Transportation Research Part B 46, 529–543.

6. Ando, N. and Taniguchi, E., 2006. Travel time reliability in vehicle routing and scheduling with time windows. Netw Spat Econ 6, 293–311.

7. Atamtürk, A. and Zhang, M., 2007. Two-stage robust network row and design under demand uncertainty. Operations Research 55, 662–673.

8. Averbakh, I., 2000. Minimax regret solutions for minimax optimization problems with uncertainty. Operations Research Letters 27, 57-65.

9. Averbakh, I. and Berman, O., 2000. Minimax regret median location on a network under uncertainty. INFORMS Journal on Computing 12(2), 104-110.

10. Baldi, M.M., Ghirardi, M., Perboli, G. and Tadei, R., 2012. The capacitated transshipment location problem under uncertainty: a computational study. Procedia – Social and Behavioral Sciences, Seventh International Conference on City Logistics 39, 424-436.

11. Beardwood, J., Halton, J.H. And Hammersley, J.M., 1959. The shortest path through many points. Mathematical Proceedings of the Cambridge Philosophical Society 55(4), 299-327.

12. Benders, J.F., 1962. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4, 238–252.

13. Bent, R.W. and Van Hentenryck, P., 2004. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. Operations Research 52(6), 977-987.

14. Berman, O., Jaillet, P. and Simchi-Levi, D., 1995. Location–routing problems with uncertainty. (In: Drezner, Z., Ed.), Facility Location: A Survey of Applications and Methods. New York. Springer, 427–452.

15. Berman, O. and Larson, R.C., 2001. Deliveries in an inventory/routing problem using stochastic dynamic programming. Transportation Science 35(2), 192-213.

16. Bertsimas, D.J. SimchiLevi, D., 1996. A new generation of vehicle routing research: robust algorithms, addressing uncertainty. Operations Research 44(2), 286-304.

17. Bertsimas, D., Brown, D.B. and Caramanis, C., 2011. Theory and applications of robust optimization. SIAM Review 53(3), 464–501.

18. Binart, S., Dejax, P., Gendreau, M. and Semet, F., 2015. A 2-stage method for a field service routing problem with stochastic travel and service times. Computers & Operations Research, Articles in Advance.

19. Birge, J.R., 1982. The value of the stochastic solution in stochastic linear-programs with fixed recourse. Mathematical Programming 24(3), 314-325.

20. Birge, J.R. and Louveaux, F. V., 1988. A multicut algorithm for two-stage stochastic linear programs. European Journal of Operational Research 34, 384–392.

21. Birge, J. R. and Louveaux F., 2011. Introduction to Stochastic Programming. Springer Series in Operations Research and Financial Engineering. Springer.

22. Bischoff, M. and Dächert, K., 2009. Allocation search methods for a generalized class of location–allocation problems. European Journal of Operational Research 192(3), 793–807.

23. Brandao, J., 2009. A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem. European Journal of Operational Research 195(3), 716-728.

24. Branston, D., 1976. Link capacity functions: a review. Transportation Research 10, 223-236.

25. Bundschuh, M., Klabjan, D. and Thurston, D., 2006. Modeling robust and reliable supply chains. Working Paper, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA.

26. Campbell, J.F., Langevin, A. and Mbaraga, P., 1996. Continuous approximation models in freight distribution: an overview. Transportation Research Part E 30(3), 163-188.

27. Castillo, J.M., 1999. A heuristic for the traveling salesman problem based on a continuous approximation, Transportation Research Part B 33, 123-152.

28. Chan, Y., Carter, W.B. and Burnes, M.D., 2001. A multiple-depot, multiple-vehicle, location-routing problem with stochastically processed demands. Computers & Operations Research 28(8), 803-826.

29. Christofides, N. And Eilon, S., 1969. Expected distances in distribution problems. Operations Research Quarterly 20(4), 437-443.

30. Coelho, L.C., Cordeau, J.-F. and Laporte, G., 2014. Thirty years of inventory routing. Transportation Science 48(1), 1-19.

31. Crainic, T.G., Ricciardi, N. and Storchi, G., 2009. Models for evaluating and planning city logistics systems. Transportation Science 43(4), 432-454.

32. Crainic T. G., Errico, F., Rei, W. and Ricciardi, N., 2012a. Modeling demand uncertainty in two-tiered CL tactical planning. CIRRELT-2012-65.

33. Crainic T. G., Hewitt, M. and Rei, W., 2012b. Scenario-clustering in a progressive hedging-based meta-heuristic for stochastic network design CIRRELT-2012-41.

34. Crainic T. G., Hewitt M. and Rei W., 2014. Partial decomposition strategies for two-stage stochastic integer programs. CIRRELT-2014-13.

35. Daganzo, C.F., 1984a. The distance traveled to visit N points with a maximum of C stops per vehicle: an analytic model and an application. Transportation Science 18(4), 331-350.

36. Daganzo, C., 1984b. The length of tours in zones of different shapes. Transportation Research B 188(2), 135-145.

37. Daskin, M.S., Coullard, C.R. and Shen, Z.-J.M., 2002. An inventory location model: formulation, solution algorithm and computational results. Annals of Operations Research 110, 83–106.

38. Doyen, A., Necati, A. and Barbarosoglu, G., 2012. A two-echelon stochastic facility location model for humanitarian relief logistics. Optimization Letters 6, 1123–1145.

39. Fazel-Zarandi, M.M., Berman, O. and Christopher, B.J., 2013. Solving a stochastic facility location/fleet management problem with logic-based Benders decomposition. IIE Transactions 45(8), Special Issue SI, 896-911.

40. Figliozzi, M.A., 2009. Planning approximations to the average length of vehicle routing problems with time window constraints. Transportation Research Part B 43, 438–447.

41. Fischetti, M. and A. Lodi., 2003. Local branching, Mathematical Programming 98, 23-47.

42. Fischetti, M., Salvagnin, D. and Zanette, A., 2008. Minimal infeasible subsystems and Benders cuts. Technical Report, DEI, University of Padova, Italy.

43. Fischetti, M, Salvagnin, D. and Zanette, A., 2010. A note on the selection of benders' cuts. Mathematical Programming 124, 175–182.

44. Fortz, B. and Poss, M., 2009. An improved Benders decomposition applied to a multi-layer network design problem, Operations Research Letters 37, 359–364.

45. Fouskakis, D. and Draper, D., 2002. Stochastic programming: a review. International Statistical Review, 70(3), 315-349.

46. Fukasawa, R., Longo, H., Lysgaard, J., de Aragao, M.P., Reis, M., Uchoa, E. and Werneck, R.F., 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical Programming Series A 106, 491–511.

47. Gabrel, V., Murat, C. and Thiele A., 2014a. Recent advances in robust optimization: an overview. European Journal of Operational Research 235(3), 471-483.

48. Gabrel, V., Lacroix, M., Murat, C. and Remli, N., 2014b. Robust location transportation problems under uncertain demands. Discrete Applied Mathematics 164(1), 100–111.

49. Gendreau, M., Laporte, G. and Séguin, R., 1996. Stochastic vehicle routing. European Journal of Operational Research 88, 3-12.

50. Geoffrion, M. and Graves, G.W., 1974. Multicommodity distribution system design by Benders decomposition. Management Science 20(5), 822–844.

51. Gheysens, F., Golden, B. and Assad, A., 1984. A comparison of techniques for solving the fleet size and mix vehicle-routing problem. OR Spektrum 6(4), 207-216.

52. Gounaris, C. E., Wiesemann, W. and Floudas, C.A., 2013. The robust capacitated vehicle routing problem under demand uncertainty. Operations Research 61(3), 677–693.

53. Gutierrez, G.J., Kouvelis, P. and Kurawarwala, A.A., 1996. A robustness approach to uncapacitated network design problems. European Journal of Operational Research 94, 362-376.

54. Hall, R.W., 1984. Travel distance through transportation terminals on a rectangular grid. Journal of Operational Research Society 35(12), 1067-1078.

55. Han, J., Lee, C. and Park S., 2013. A robust scenario approach for the vehicle routing problem with uncertain travel times. Transportation Science 48(3), 373-390.

56. Inuiguchi, M. and Sakawa, M., 1995. Minimax regret solution to linear programming problems with an interval objective function. European Journal of Operational Research 86, 526-536.

57. Kenyon, A.S. and Morton, D.P., 2003. Stochastic vehicle routing with random travel times. Transportation Science 37(1), 69-82.

58. Kleywegt, A.J., Nori, V.S. and Savelsbergh, M.W.P., 2002. The stochastic inventory routing problem with direct deliveries. Transportation Science 36(1), 94-118.

59. Koulakezian, A., Soliman, H. M., Tang, T. and Leon-Garcia, A., 2012. Robust traffic assignment in transportation, networks using network criticality. Vehicular Technology Conference (VTC Fall), IEEE.

60. Kung-Jeng, W., Makonda, B. and Liub, S.Y., 2011. Location and allocation decisions in a two-echelon supply chain with stochastic demand – A genetic-algorithm based solution. Expert Systems with Applications 38(5), 6125–6131.

61. Laporte, G., Louveaux, F. and Mercure, H., 1989. Models and exact-solutions for a class of stochastic location-routing problems. European Journal of Operational Research 39(1), 71-78.

62. List, G. F., Wood, B., Nozick, L.K., Turnquist, M.A., Jones, D.A., Kjeldgaard, E. A. and Lawton, C. R., 2003. Robust optimization for fleet planning under uncertainty. Transportation Research Part E 39(3), 209-227.

63. Listes, O. and Dekker,R., 2005. A stochastic approach to a case study for product recovery network design. European Journal of Operational Research 160(2), 268–287.

64. Liu, Z., Guo, S., Snyder, L.V., Lim, A. and Peng, P., 2010. A p-robust capacitated network design model with facility disruptions. Advanced manufacturing and sustainable logistics: Proceedings of the 8th International Heinz Nixdorf Symposium 21(22), 269-280.

65. Liu, S., Huang, W. and Ma, H., 2009. An effective genetic algorithm for the fleet size and mix vehicle routing problems. Transportatıon Research Part E 45(3), 434-445.

66. Louveaux, F.V. and Peeters, D., 1992. A dual-based procedure for stochastic facility location. Operations Research 40(3), 564-573.

67. Madansky, A., 1960. Inequalities for stochastic linear programming problems. Management Science 6, 197-204.

68. Maggioni, F. and Wallace, S.W., 2012. Analyzing the quality of the expected value solution in stochastic programming. Annals of Operations Research 200, 37–54.

69. Magnanti, T. and Wong, R., 1981. Accelerating Benders decomposition algorithmic enhancement and model selection criteria. Operations Research 29, 464–484.

70. Mausser, H.E. and Laguna, M., 1999. Minimizing the maximum relative regret for linear programs with interval objective function coefficients. Journal of the Operational Research Society 50, 1063-1070.

71. McDaniel, D. and Devine, M., 1977. A modified Benders' partitioning algorithm for mixed integer programming. Management Science 24(3), 312-319.

72. Mudchanatongsuk, S., Ordonez, F. and Liu, J., 2008. Robust solutions for network design under transportation cost and demand uncertainty. Journal of the Operational Research Society 59, 652-662.

73. Noyan, N., 2012. Risk-averse two-stage stochastic programming with an application to disaster management. Computers & Operations Research 39, 541–559.

74. Oliveira, F., Grossmann, I. E. and Hamacher, S., 2014. Accelerating Benders stochastic decomposition for the optimization under uncertainty of the petroleum product supply chain. Computers & Operations Research 49, 47-58.

75. Papadakos, N., 2008. Practical enhancements to the Magnanti-Wong method. Operations Research Letters 36, 444–449.

76. Peng, P., Snyder, L. V., Lima, A. and Liu, Z., 2011. Reliable logistics networks design with facility disruptions. Transportation Research Part B 45, 1190–1211.

77. Rajagopalan, H. K., Vergara, E., Saydam, C. and Xiao, J., 2007. Developing effective meta-heuristics for a probabilistic location model via experimental design. European Journal of Operational Research 177(1), 83–101.

78. Rei, W., Cordeau, J.-F., Gendreau, M. and Soriano, P., 2009. Accelerating Benders decomposition by local branching. INFORMS Journal on Computing 21(2), 333–345.

79. Remli, N., 2011. Robustesse en programmation linéaire. Ph.D. thesis, Université Paris-Dauphine. Paris, France.

80. Ricciardi, N., Tadei, R. and Grosso, A., 2002. Optimal facility location with random throughput costs. Computers & Operations Research 29, 593–607.

81. Rosenhead, J., Elton, M. and Gupta, S.K., 1972. Robustness and optimality as criteria for strategic decisions. Operational Research Quarterly 23(4), 413–431.

82. Van Roy, T. J., 1986. A cross decomposition algorithm for capacitated facility location. Operations Research 34(1), 145-163.

83. Roy, B., 2010. Robustness in operational research and decision aiding: a multi-faceted issue. European Journal of Operational Research 200, 629–638.

84. Saharidis, G.K.D. and Ierapetritou, M.G., 2010a. Improving Benders decomposition using maximum feasible subsystem cut generation strategy. Computers and Chemical Engineering 34, 1237–1245.

85. Saharidis, G.K.D., Minoux, M., and Ierapetritou, M.G., 2010b. Accelerating Benders method using covering cut bundle generation. International Transactions in Operational Research 17, 221–237.

86. Saharidis, G., Boile, M. and Theofanis, S., 2011. Initialization of the Benders master problem using valid inequalities applied to fixed-charge network problems. Expert Systems with Applications 38(6), 6627-6636.

87. Saharidis, G.K.D. and Ierapetritou, M.G., 2013. Speed-up Benders decomposition using maximum density cut generation. Annals of Operations Research 210, 101–123.

88. Santoso, T., Ahmed, S., Goetschalckx, M. and Shapiro, A., 2005. A stochastic programming approach for supply chain network design under uncertainty. European Journal of Operational Research 167, 96–115.

89. Sherali, H. and Lunday, B., 2013. On generating maximal nondominated Benders cuts. Annals of Operations Research 210, 57-72.

90. Sheu, J.B., 2006. A novel dynamic resource allocation model for demand-responsive city logistics distribution operations. Transportation Research Part E 42(6), 445-472.

91. Van Slyke, R. and Wets, R.J.-B., 1969. L-shaped programs with applications to optimal control and stochastic programming. SIAM Journal on Applied Mathematics 17, 638–663.

92. Snyder, L.V., 2006. Facility location under uncertainty: a review. IIE Transactions 38(7), 547-564.

93. Snyder, L.V. and Daskin, M.S., 2006. Stochastic p-robust location problems. IIE Transactions 38, 971–985.

94. Solyalı, O., Cordeau, J.-F. and Laporte, G., 2012. Robust inventory routing under demand uncertainty. Transportation Science 46(3), 327-340.

95. Stanimirovic, Z. and Kratica, J., 2007. Genetic algorithms for solving the discrete ordered median problem. European Journal of Operational Research 182(3), 983–1001.

96. Taniguchi, E., Thompson, R. G. and Yamada, T., 2010. Incorporating risks in city logistics. Procedia Social and Behavioral Sciences 2, 5899–5910.

97. Taş, D., Gendreau, M., Dellaert, N., van Woensel, T. and de Kok, A.G., 2014. Vehicle routing with soft time windows and stochastic travel times: a column generation and branch-and-price solution approach. European Journal of Operational Research 236(3), 789-799.

98. Tsiakis, P., Shah, N. and Pantelides, C.C., 2001. Design of multi-echelon supply chain networks under demand uncertainty. Industrial and Engineering Chemistry Research 40, 3585–3604.

99. Ukkusuri, S.V., Mathew, T.V. and Waller, S.T., 2007. Robust transportation network design under demand uncertainty. Computer-Aided Civil and Infrastructure Engineering 22, 6–18.

100. Uster, H., Agrahari, H., 2011. A Benders decomposition approach for a distribution network design problem with consolidation and capacity considerations. Operations Research Letters 39(2), 138–143.

101. Van Woensel, T., Kerbache, L., Peremans, N. and Vandaele, N., 2008. Vehicle routing with dynamic travel times: a queueing approach. European Journal of Operations Research 186(3), 990-1007.

102. Wang, Q., Batta, R. and Rump, C.M., 2002. Algorithms for a facility location problem with stochastic customer demand and immobile servers. Annals of Operations Research 111(1-4), 17-34.

103. Wang, N., Lu, J.C. and Kvam, P., 2006.Reliability modeling in spatially distributed logistics systems. IEEE Transactions on Reliability 55(3), 525-534.

104. Wentges, P., 1996. Accelerating Benders' decomposition for the capacitated facility location problem. Math. Methods Oper. Res. 44, 267–290.

105. Yin, Y. and Lou, Y., 2009. Robust optimization approach for transportation network design under demand uncertainty. Critical Issues in Transportation Systems Planning, Development and Management, ICCTP, 1-12.

106. You, F. and Grossmann I.E., 2013. Multicut Benders decomposition algorithm for process supply chain planning under uncertainty. Annals of Operations Research 210, 191–211.

107. Yu, C.S. and Li, H.L., 2000. A robust optimization model for stochastic logistic problems. Int. J. Production Economics 64, 385-397.

APPENDIX

Table A.1: Results of TLAP Experiments with Small Instances

Results for Small Instances	TLAP			
	Best Integer	Lower Bound	CPU Time (sec)	GAP%
R_u_L_10_1	250160.1	250160.1	2.6	0.0
R_u_L_10_2	250145.6	250145.6	3.0	0.0
R_n_L_10_1	250179.6	250179.6	7.2	0.0
R_n_L_10_2	250162.6	250162.6	1.4	0.0
C_u_L_10_1	250180.5	250180.5	2.9	0.0
C_u_L_10_2	250139.0	250139.0	1.0	0.0
C_n_L_10_1	250167.2	250167.2	2.4	0.0
C_n_L_10_2	250181.8	250181.8	1.9	0.0
RC_u_L_10_1	250180.7	250180.7	4.2	0.0
RC_u_L_10_2	250159.0	250159.0	5.1	0.0
RC_n_L_10_1	250155.7	250155.7	3.2	0.0
RC_n_L_10_2	250174.7	250174.7	2.1	0.0
R_u_S_10_1	730244.1	730244.1	4.4	0.0
R_u_S_10_2	686878.5	686878.5	4.7	0.0
R_n_S_10_1	788669.3	788669.3	7.1	0.0
R_n_S_10_2	737856.4	737856.4	1.5	0.0
C_u_S_10_1	791586.6	791586.6	3.0	0.0
C_u_S_10_2	666935.7	666935.7	0.9	0.0
C_n_S_10_1	751561.4	751561.4	2.1	0.0
C_n_S_10_2	795264.1	795264.1	2.0	0.0
RC_u_S_10_1	792066.5	792066.5	4.0	0.0
RC_u_S_10_2	727048.0	727048.0	5.6	0.0
RC_n_S_10_1	708776.3	708776.3	3.3	0.0
RC_n_S_10_2	774116.4	774116.4	2.4	0.0

Table A.2: Results of TLAP Experiments with Medium Instances

Results for Medium Instances	TLAP			
	Best Integer	Lower Bound	CPU Time (sec)	GAP%
R_u_L_100_1	251602.1	251602.1	9706.6	0.0
R_u_L_100_2	251557.4	251557.4	6954.7	0.0
R_n_L_100_1	251744.3	251744.3	7538.3	0.0
R_n_L_100_2	251700.0	251700.0	9328.1	0.0
C_u_L_100_1	251551.2	251551.2	6305.9	0.0
C_u_L_100_2	251653.3	251653.3	5482.6	0.0
C_n_L_100_1	251560.9	251560.9	5139.4	0.0
C_n_L_100_2	251702.5	251702.5	6060.7	0.0
RC_u_L_100_1	251455.0	251455.0	6361.9	0.0
RC_u_L_100_2	251619.4	251619.4	5923.0	0.0
RC_n_L_100_1	251677.3	251677.3	5553.7	0.0
RC_n_L_100_2	251460.3	251460.3	6751.9	0.0
R_u_S_100_1	730624.5	730624.5	1754.9	0.0
R_u_S_100_2	717207.5	717207.5	1563.3	0.0
R_n_S_100_1	773292.2	773292.2	1994.9	0.0
R_n_S_100_2	760011.0	760011.0	1926.9	0.0
C_u_S_100_1	715369.3	715369.3	436.7	0.0
C_u_S_100_2	745984.6	745984.6	1148.3	0.0
C_n_S_100_1	718274.1	718274.1	769.0	0.0
C_n_S_100_2	760758.1	760758.1	1265.3	0.0
RC_u_S_100_1	686507.6	686507.6	769.0	0.0
RC_u_S_100_2	735827.8	735827.8	1145.2	0.0
RC_n_S_100_1	753200.8	753200.8	857.8	0.0
RC_n_S_100_2	688074.3	688074.3	1090.5	0.0

Table A.3: Results of TLAP Experiments with Large Instances

Results for Large Instances	TLAP			
	Best Integer	Lower Bound	CPU Time (sec)	GAP%
R_u_L_1000_1	1010631.3	239004.7	14463.2	322.8
R_u_L_1000_2	1010936.1	239634.9	14444.4	321.9
R_n_L_1000_1	1014487.2	241774.3	14467.1	319.6
R_n_L_1000_2	1013188.1	241038.1	14444.7	320.3
C_u_L_1000_1	1012673.4	240561.9	14462.5	321.0
C_u_L_1000_2	962374.4	240517.9	14444.8	300.1
C_n_L_1000_1	1013515.5	241295.0	14471.5	320.0
C_n_L_1000_2	1013179.9	240895.3	14445.3	320.6
RC_u_L_1000_1	1012550.9	240373.2	14474.4	321.2
RC_u_L_1000_2	961509.3	239934.2	14444.7	300.7
RC_n_L_1000_1	1011994.5	239795.3	14469.4	322.0
RC_n_L_1000_2	1014106.8	242099.6	14445.0	318.9
R_u_S_1000_1	1318937.5	635392.4	14464.9	107.6
R_u_S_1000_2	1328081.8	657559.7	14457.0	102.0
R_n_S_1000_1	1434617.2	722288.0	14644.4	98.6
R_n_S_1000_2	1395642.1	699721.9	14456.6	99.5
C_u_S_1000_1	1380202.2	683079.7	14466.4	102.1
C_u_S_1000_2	1371045.4	681245.1	14456.9	101.3
C_n_S_1000_1	1405464.6	705906.1	14464.8	99.1
C_n_S_1000_2	1395396.2	695293.2	14456.9	100.7
RC_u_S_1000_1	1376527.1	679669.9	14463.9	102.5
RC_u_S_1000_2	1345068.2	666934.7	14456.9	101.7
RC_n_S_1000_1	1359835.6	662088.2	14461.9	105.4
RC_n_S_1000_2	1423203.7	728960.7	14457.6	95.2

Table A.4: Results of MC Experiments with Small Instances

Results for Small Instances	MC				
	Best Integer	Lower Bound	Iterations (Nodes Checked)	CPU Time (sec)	GAP%
R_u_L_10_1	250160.1	250160.1	42	402.1	0.0
R_n_L_10_1	250179.6	250179.5	85	1973.3	0.0
C_u_L_10_1	250180.5	250180.5	53	483.1	0.0
C_n_L_10_1	250167.2	250167.1	27	160.2	0.0
RC_u_L_10_1	250180.7	250180.6	48	492.8	0.0
RC_n_L_10_1	250155.7	250155.7	37	267.4	0.0
R_u_S_10_1	730244.1	730244.1	47	698.2	0.0
R_n_S_10_1	788669.3	788669.3	91	3023.7	0.0
C_u_S_10_1	791586.5	791586.5	54	737.1	0.0
C_n_S_10_1	751561.4	751561.4	28	320.2	0.0
RC_u_S_10_1	792066.5	792066.5	49	714.0	0.0
RC_n_S_10_1	708776.3	708776.3	32	385.0	0.0

Table A.5: Results of MC Experiments with Medium Instances

Results for Medium Instances	MC				
	Best Integer	Lower Bound	Iterations (Nodes Checked)	CPU Time (sec)	GAP%
R_u_L_100_1	251602.1	251220.2	225	14448.7	0.2
R_n_L_100_1	251744.3	251424.0	218	14479.0	0.1
C_u_L_100_1	251551.2	251442.9	223	14421.6	0.0
C_n_L_100_1	251560.9	251384.3	215	14403.7	0.1
RC_u_L_100_1	251472.8	251209.9	220	14483.7	0.1
RC_n_L_100_1	251677.3	251545.7	217	14536.2	0.1
R_u_S_100_1	733263.9	614777.0	216	14535.1	19.3
R_n_S_100_1	774182.3	676434.5	216	14563.5	14.5
C_u_S_100_1	715369.3	681334.2	228	14514.3	5.0
C_n_S_100_1	718274.1	665192.2	209	14466.0	8.0
RC_u_S_100_1	686507.6	610984.3	217	14523.9	12.4
RC_n_S_100_1	753200.8	710940.2	207	14503.9	5.9

Table A.6: Results of MC Experiments with Large Instances

Results for Large Instances	**MC**				
	Best Integer	**Lower Bound**	**Iterations (Nodes Checked)**	**CPU Time (sec)**	**GAP%**
R_u_L_1000_1	265595.0	260861.2	292	14406.7	1.8
R_n_L_1000_1	267886.6	264619.8	298	14454.8	1.2
C_u_L_1000_1	265947.9	263036.6	300	14469.9	1.1
C_n_L_1000_1	267120.0	263687.2	296	14483.9	1.3
RC_u_L_1000_1	266456.9	262706.3	295	14453.5	1.4
RC_n_L_1000_1	265975.3	262143.1	287	14417.1	1.5
R_u_S_1000_1	718170.6	575208.1	289	14468.1	24.9
R_n_S_1000_1	785211.0	688490.6	289	14517.4	14.0
C_u_S_1000_1	728436.7	641946.1	300	14424.4	13.5
C_n_S_1000_1	763599.7	659980.2	289	14494.0	15.7
RC_u_S_1000_1	743705.8	630925.2	287	14502.7	17.9
RC_n_S_1000_1	728554.3	614219.8	278	14486.8	18.6

Table A.7: Results of SGC Experiments with Small Instances

Results for Small Instances	SGC				
	Best Integer	Lower Bound	Iterations (Nodes Checked)	CPU Time (sec)	GAP%
R_u_L_10_1	250160.1	250160.0	120	618.5	0.0
R_u_L_10_2	250145.6	250145.5	258	3293.4	0.0
R_n_L_10_1	250179.6	250179.5	345	4543.0	0.0
R_n_L_10_2	250162.6	250162.5	69	654.2	0.0
C_u_L_10_1	250180.5	250180.5	142	789.2	0.0
C_u_L_10_2	250139.0	250139.0	50	480.3	0.0
C_n_L_10_1	250167.2	250167.2	60	224.9	0.0
C_n_L_10_2	250181.8	250181.7	186	2139.4	0.0
RC_u_L_10_1	250180.7	250180.6	188	1248.2	0.0
RC_u_L_10_2	250159.0	250159.0	123	1197.1	0.0
RC_n_L_10_1	250155.7	250155.6	77	281.5	0.0
RC_n_L_10_2	250174.7	250174.6	104	1018.8	0.0
R_u_S_10_1	730244.1	730244.1	167	1911.0	0.0
R_u_S_10_2	686878.5	686878.5	456	9008.9	0.0
R_n_S_10_1	788669.3	788669.3	462	9435.6	0.0
R_n_S_10_2	737856.4	737856.4	71	671.5	0.0
C_u_S_10_1	791586.5	791586.5	150	1594.1	0.0
C_u_S_10_2	666935.7	666935.7	50	472.4	0.0
C_n_S_10_1	751561.4	751561.4	61	595.7	0.0
C_n_S_10_2	795264.1	795264.1	191	2288.5	0.0
RC_u_S_10_1	792066.5	792066.5	198	2297.8	0.0
RC_u_S_10_2	727048.0	727048.0	146	1560.1	0.0
RC_n_S_10_1	708776.3	708776.3	86	829.3	0.0
RC_n_S_10_2	774116.4	774116.4	108	1070.9	0.0

Table A.8: Results of SGC Experiments with Medium Instances

Results for Medium Instances	SGC				
	Best Integer	Lower Bound	Iterations (Nodes Checked)	CPU Time (sec)	GAP%
R_u_L_100_1	251610.9	251305.6	599	14430.4	0.1
R_u_L_100_2	251557.4	251361.9	626	14432.5	0.1
R_n_L_100_1	251747.3	251477.6	591	14401.50	0.1
R_n_L_100_2	251700.0	251481.2	609	14401.5	0.1
C_u_L_100_1	251551.2	251482.5	644	14424.0	0.0
C_u_L_100_2	251653.3	251554.2	632	14406.9	0.0
C_n_L_100_1	251560.9	251438.9	619	14419.4	0.0
C_n_L_100_2	251702.5	251541.4	622	14428.7	0.1
RC_u_L_100_1	251455.0	251287.8	615	14421.2	0.1
RC_u_L_100_2	251619.4	251418.8	602	14432.5	0.1
RC_n_L_100_1	251677.3	251578.9	637	14400.4	0.0
RC_n_L_100_2	251460.2	251300.8	611	14421.6	0.1
R_u_S_100_1	738260.1	635714.9	575	14414.9	16.1
R_u_S_100_2	717207.5	648436.0	589	14448.7	10.6
R_n_S_100_1	773292.2	692359.3	579	14401.0	11.7
R_n_S_100_2	760011.0	694352.5	615	14437.9	9.5
C_u_S_100_1	715369.3	691145.5	609	14408.1	3.5
C_u_S_100_2	745984.6	710877.3	578	14434.2	4.9
C_n_S_100_1	718274.1	677004.1	576	14417.6	6.1
C_n_S_100_2	760758.1	712524.6	616	14427.1	6.8
RC_u_S_100_1	686507.6	625165.4	564	14423.4	9.8
RC_u_S_100_2	735827.8	675498.8	594	14410.6	8.9
RC_n_S_100_1	753200.8	723280.8	623	14418.4	4.1
RC_n_S_100_2	688074.3	631148.8	568	14400.0	9.0

Table A.9: Results of SGC Experiments with Large Instances

Results for Large Instances	SGC				
	Best Integer	Lower Bound	Iterations (Nodes Checked)	CPU Time (sec)	GAP%
R_u_L_1000_1	265595.0	261182.8	612	14422.5	1.7
R_u_L_1000_2	265606.8	261511.3	621	14431.9	1.6
R_n_L_1000_1	267840.4	264764.7	611	14423.5	1.2
R_n_L_1000_2	266886.5	263320.3	619	14421.7	1.4
C_u_L_1000_1	265947.9	263409.0	669	14424.6	1.0
C_u_L_1000_2	265442.7	263169.6	654	14429.3	0.9
C_n_L_1000_1	267120.0	263842.6	613	14434.1	1.2
C_n_L_1000_2	265864.7	263564.5	632	14408.9	0.9
RC_u_L_1000_1	266456.9	262908.5	598	14410.6	1.3
RC_u_L_1000_2	265511.1	262077.9	615	14410.1	1.3
RC_n_L_1000_1	265940.7	262351.4	594	14404.4	1.4
RC_n_L_1000_2	267154.6	264553.4	617	14401.5	1.0
R_u_S_1000_1	717802.7	584472.5	593	14408.3	22.8
R_u_S_1000_2	717867.8	594527.9	605	14408.7	20.7
R_n_S_1000_1	785211.0	689833.0	451	14421.6	13.8
R_n_S_1000_2	757366.8	649543.8	616	14439.8	16.6
C_u_S_1000_1	728436.7	651724.2	660	14415.3	11.8
C_u_S_1000_2	713279.5	644600.9	654	14428.6	10.7
C_n_S_1000_1	763599.7	665099.5	603	14405.0	14.8
C_n_S_1000_2	725942.1	657190.7	627	14431.0	10.5
RC_u_S_1000_1	743705.8	635172.4	529	14421.7	17.1
RC_u_S_1000_2	715332.6	611778.2	605	14402.5	16.9
RC_n_S_1000_1	727614.1	619962.8	565	14411.8	17.4
RC_n_S_1000_2	764637.5	686384.6	611	14418.3	11.4

Table A.10: Results of PD Experiments with Small Instances

Results for Small Instances	PD				
	Best Integer	Lower Bound	Iterations (Nodes Checked)	CPU Time (sec)	GAP%
R_u_L_10_1	250160.1	250160.0	38	375.2	0.0
R_u_L_10_2	250145.6	250145.6	90	1026.8	0.0
R_n_L_10_1	250179.6	250179.5	68	770.6	0.0
R_n_L_10_2	250162.6	250162.6	19	192.2	0.0
C_u_L_10_1	250180.5	250180.5	30	241.9	0.0
C_u_L_10_2	250139.2	250139.2	14	151.9	0.0
C_n_L_10_1	250167.2	250167.3	13	105.2	0.0
C_n_L_10_2	250181.8	250181.8	30	334.0	0.0
RC_u_L_10_1	250180.7	250180.8	35	271.3	0.0
RC_u_L_10_2	250159.0	250159.0	34	352.4	0.0
RC_n_L_10_1	250155.7	250155.6	23	174.5	0.0
RC_n_L_10_2	250174.7	250174.7	21	147.8	0.0
R_u_S_10_1	730244.1	730244.1	41	494.5	0.0
R_u_S_10_2	686878.5	686878.5	130	1721.5	0.0
R_n_S_10_1	788669.3	788669.3	82	1089.4	0.0
R_n_S_10_2	737856.4	737856.4	19	194.2	0.0
C_u_S_10_1	791586.5	791586.5	33	386.4	0.0
C_u_S_10_2	667464.0	667464.0	14	149.3	0.0
C_n_S_10_1	751561.4	751561.4	13	150.5	0.0
C_n_S_10_2	795264.1	795264.1	32	371.7	0.0
RC_u_S_10_1	792066.5	792066.4	35	418.6	0.0
RC_u_S_10_2	727048.0	727048.0	36	411.7	0.0
RC_n_S_10_1	708776.3	708776.3	23	268.8	0.0
RC_n_S_10_2	774116.4	774116.4	22	239.4	0.0

Table A.11: Results of PD Experiments with Medium Instances

Results for Medium Instances	**PD**				
	Best Integer	**Lower Bound**	**Iterations (Nodes Checked)**	**CPU Time (sec)**	**GAP%**
R_u_L_100_1	251622.3	251222.2	111	14425.4	0.2
R_u_L_100_2	251564.5	251283.9	146	14458.2	0.1
R_n_L_100_1	251747.7	251462.3	105	14410.3	0.1
R_n_L_100_2	251703.2	251441.8	147	14512.0	0.1
C_u_L_100_1	251551.2	251474.7	223	14413.6	0.0
C_u_L_100_2	251653.3	251441.8	179	14402.4	0.1
C_n_L_100_1	251560.9	251408.3	168	14501.7	0.1
C_n_L_100_2	251702.5	251527.6	160	14504.4	0.1
RC_u_L_100_1	251462.0	251200.6	133	14551.3	0.1
RC_u_L_100_2	251619.4	251380.1	136	14523.5	0.1
RC_n_L_100_1	251677.3	251578.0	175	14536.5	0.0
RC_n_L_100_2	251460.2	251223.7	145	14442.2	0.1
R_u_S_100_1	736679.9	618221.1	128	14565.6	19.2
R_u_S_100_2	719352.1	635911.5	151	14459.2	13.1
R_n_S_100_1	774318.1	689661.6	128	14536.1	12.3
R_n_S_100_2	760971.7	684223.0	162	14551.9	11.2
C_u_S_100_1	715369.3	692121.1	218	14470.8	3.4
C_u_S_100_2	745984.6	712047.4	196	14444.1	4.8
C_n_S_100_1	718274.1	674071.3	178	14459.9	6.6
C_n_S_100_2	760758.1	708829.8	164	14472.0	7.3
RC_u_S_100_1	688607.4	611963.1	147	14510.2	12.5
RC_u_S_100_2	735827.8	665609.6	145	14403.8	10.5
RC_n_S_100_1	753200.8	723831.1	182	14474.6	4.1
RC_n_S_100_2	688074.3	618433.7	154	14433.3	11.3

Table A.12: Results of PD Experiments with Large Instances

Results for Large Instances	PD				
	Best Integer	Lower Bound	Iterations (Nodes Checked)	CPU Time (sec)	GAP%
R_u_L_1000_1	265593.4	261407.5	6	14732.5	1.6
R_u_L_1000_2	265595.6	261719.0	7	15631.8	1.5
R_n_L_1000_1	267886.9	265357.0	8	15944.8	1.0
R_n_L_1000_2	266925.9	264037.8	7	15381.0	1.1
C_u_L_1000_1	265947.9	263486.3	10	15753.6	0.9
C_u_L_1000_2	265442.7	263184.4	10	14930.4	0.9
C_n_L_1000_1	267120.0	264365.8	7	14517.8	1.0
C_n_L_1000_2	265864.7	263970.4	9	14478.7	0.7
RC_u_L_1000_1	266499.1	263392.3	8	15186.8	1.2
RC_u_L_1000_2	265511.1	262288.1	7	15534.5	1.2
RC_n_L_1000_1	265940.7	262794.1	8	16425.0	1.2
RC_n_L_1000_2	267154.6	264957.7	10	15809.1	0.8
R_u_S_1000_1	717802.7	592232.1	7	15670.3	21.2
R_u_S_1000_2	717867.8	601568.9	7	14777.6	19.3
R_n_S_1000_1	786605.8	711382.1	21	15407.8	10.6
R_n_S_1000_2	757778.0	671133.9	7	14529.6	12.9
C_u_S_1000_1	728436.7	655387.9	36	14674.6	11.1
C_u_S_1000_2	713279.5	647119.7	35	14685.2	10.2
C_n_S_1000_1	763599.7	680975.4	7	14729.7	12.1
C_n_S_1000_2	725942.1	669852.2	33	14684.4	8.4
RC_u_S_1000_1	744972.8	651763.6	7	14715.1	14.3
RC_u_S_1000_2	715332.6	618642.6	7	16495.3	15.6
RC_n_S_1000_1	728220.1	633822.2	7	16491.3	14.9
RC_n_S_1000_2	764637.5	699808.7	37	15031.4	9.3

Table A.13: Results of B&C-PD Experiments with Small Instances

Results for Small Instances	B&C-PD				
	Best Integer	Lower Bound	Iterations (Nodes Checked)	CPU Time (sec)	GAP%
R_u_L_10_1	250160.1	250160.1	38(993)	1295.4	0.0
R_n_L_10_1	250179.6	250179.6	93(313)	3781.9	0.0
C_u_L_10_1	250180.5	250180.5	50(655)	998.6	0.0
C_n_L_10_1	250167.2	250167.2	25(205)	456.7	0.0
RC_u_L_10_1	250180.7	250180.7	46(935)	1237.0	0.0
RC_n_L_10_1	250155.7	250155.7	39(495)	721.2	0.0
R_u_S_10_1	730244.1	730244.1	53(155)	1927.9	0.0
R_n_S_10_1	788669.3	788669.3	138(504)	6591.5	0.0
C_u_S_10_1	791586.5	791586.5	56(795)	1313.1	0.0
C_n_S_10_1	751561.4	751561.4	26(233)	496.7	0.0
RC_u_S_10_1	792066.5	792066.5	47(134)	1686.4	0.0
RC_n_S_10_1	708776.3	708776.3	31(791)	1023.6	0.0

Table A.14: Results of B&C-PD Experiments with Medium Instances

Results for Medium Instances	B&C-PD				
	Best Integer	Lower Bound	Iterations (Nodes Checked)	CPU Time (sec)	GAP%
R_u_L_100_1	251610.9	228394.3	653(3125)	14402.7	10.2
R_n_L_100_1	251744.3	228640.8	755(2402)	14411.8	10.1
C_u_L_100_1	251551.2	228574.8	307(6613)	14400.2	10.1
C_n_L_100_1	251560.9	228526.0	432(5122)	14401.5	10.1
RC_u_L_100_1	251455.0	228352.2	440(4935)	14402.0	10.1
RC_n_L_100_1	251677.3	228698.8	352(5909)	14401.7	10.0
R_u_S_100_1	733263.9	563750.2	390(6901)	14401.3	30.1
R_n_S_100_1	773292.2	637680.2	400(6593)	14400.9	21.3
C_u_S_100_1	715369.3	617891.5	276(8125)	14401.0	15.8
C_n_S_100_1	718274.1	603261.8	348(7316)	14401.7	19.1
RC_u_S_100_1	686507.6	551126.4	334(7697)	14401.8	24.6
RC_n_S_100_1	753200.8	657751.6	339(7221)	14406.8	14.5

Table A.15: Results of B&C-PD Experiments with Large Instances

Results for Large Instances	B&C-PD				
	Best Integer	Lower Bound	Iterations (Nodes Checked)	CPU Time (sec)	GAP%
R_u_L_1000_1	265593.4	235869.9	491(1188)	14403.9	12.6
R_n_L_1000_1	267840.4	239736.7	486(1205)	14405.7	11.7
C_u_L_1000_1	266002.4	237921.4	400(1372)	14415.7	11.8
C_n_L_1000_1	267120.0	238761.4	492(1193)	14416.5	11.9
RC_u_L_1000_1	266456.9	237798.4	454(1279)	14412.4	12.1
RC_n_L_1000_1	265920.5	237232.5	503(1171)	14414.0	12.1
R_u_S_1000_1	726146.0	544566.6	246(2024)	14405.7	33.3
R_n_S_1000_1	785211.0	660570.3	313(1888)	14402.0	18.9
C_u_S_1000_1	732680.1	606110.3	268(1975)	14406.3	20.9
C_n_S_1000_1	763599.7	631311.1	218(2101)	14414.3	21.0
RC_u_S_1000_1	750672.6	602420.7	270(1995)	14403.9	24.6
RC_n_S_1000_1	727614.1	585444.9	239(2015)	14402.3	24.3

Table A.16: Results of EA Experiments with Small Instances

Results for Small Instances	EA				
	Best OFV	Avg OFV	Ave CPU Time (sec)	Best Gap from Optimal%	Avg Gap from Optimal%
R_u_L_10_1	250160.1	250160.1	16.2	0.0	0.0
R_u_L_10_2	250145.6	250145.6	35.0	0.0	0.0
R_n_L_10_1	250179.6	250179.6	16.7	0.0	0.0
R_n_L_10_2	250162.6	250162.7	23.3	0.0	0.0
C_u_L_10_1	250180.5	250180.5	16.7	0.0	0.0
C_u_L_10_2	250139.0	250139.0	21.4	0.0	0.0
C_n_L_10_1	250167.2	250167.2	14.8	0.0	0.0
C_n_L_10_2	250181.8	250181.8	27.0	0.0	0.0
RC_u_L_10_1	250180.7	250180.7	14.3	0.0	0.0
RC_u_L_10_2	250159.0	250159.4	30.4	0.0	0.0
RC_n_L_10_1	250155.7	250155.7	14.6	0.0	0.0
RC_n_L_10_2	250174.7	250174.7	24.4	0.0	0.0
R_u_S_10_1	730244.1	730387.1	16.3	0.0	0.0
R_u_S_10_2	686878.5	686895.4	33.9	0.0	0.0
R_n_S_10_1	788669.3	788669.3	17.3	0.0	0.0
R_n_S_10_2	737856.4	737856.4	23.6	0.0	0.0
C_u_S_10_1	791586.5	791756.0	14.9	0.0	0.0
C_u_S_10_2	666935.7	667041.4	23.3	0.0	0.0
C_n_S_10_1	751561.4	751561.4	15.5	0.0	0.0
C_n_S_10_2	795264.1	795264.1	25.6	0.0	0.0
RC_u_S_10_1	792066.5	792066.5	13.9	0.0	0.0
RC_u_S_10_2	727048.0	727048.0	29.9	0.0	0.0
RC_n_S_10_1	716997.6	716997.6	14.6	1.2	1.2
RC_n_S_10_2	774116.4	774116.4	26.8	0.0	0.0

Table A.17: Results of EA Experiments with Medium Instances

Results for Medium Instances	**EA**				
	Best OFV	**Avg OFV**	**Ave CPU Time (sec)**	**Best Gap from Optimal%**	**Avg Gap from Optimal%**
R_u_L_100_1	251602.1	251602.1	156.8	0.0	0.0
R_u_L_100_2	251557.4	251558.8	236.1	0.0	0.0
R_n_L_100_1	251744.3	251744.3	137.6	0.0	0.0
R_n_L_100_2	251700.0	251700.3	337.8	0.0	0.0
C_u_L_100_1	251551.2	251551.2	126.3	0.0	0.0
C_u_L_100_2	251653.3	251653.3	188.7	0.0	0.0
C_n_L_100_1	251560.9	251560.9	139.9	0.0	0.0
C_n_L_100_2	251702.5	251703.3	232.7	0.0	0.0
RC_u_L_100_1	251455.0	251455.0	139.2	0.0	0.0
RC_u_L_100_2	251619.4	251619.4	341.6	0.0	0.0
RC_n_L_100_1	251677.3	251677.3	122.9	0.0	0.0
RC_n_L_100_2	251460.2	251460.2	216.0	0.0	0.0
R_u_S_100_1	730624.5	731152.4	173.6	0.0	0.1
R_u_S_100_2	717207.5	717207.5	232.7	0.0	0.0
R_n_S_100_1	773292.2	773292.2	149.7	0.0	0.0
R_n_S_100_2	760011.0	760011.0	329.0	0.0	0.0
C_u_S_100_1	715369.3	715369.3	145.8	0.0	0.0
C_u_S_100_2	745984.6	745984.6	188.9	0.0	0.0
C_n_S_100_1	718274.1	718355.7	124.5	0.0	0.0
C_n_S_100_2	760758.1	760758.1	270.7	0.0	0.0
RC_u_S_100_1	686507.6	686507.6	145.0	0.0	0.0
RC_u_S_100_2	735827.8	735827.8	340.2	0.0	0.0
RC_n_S_100_1	753200.8	753200.8	133.2	0.0	0.0
RC_n_S_100_2	688074.3	688074.3	213.4	0.0	0.0

Table A.18: Results of EA Experiments with Large Instances

Results for Large Instances	EA				
	Best OFV	Avg OFV	Ave CPU Time (sec)	Best Gap from Best Exact%	Avg Gap from Best Exact%
R_u_L_1000_1	265593.4	265593.4	6463.9	0.0	0.0
R_u_L_1000_2	265595.6	265595.6	6345.0	0.0	0.0
R_n_L_1000_1	267840.4	267840.4	7093.3	0.0	0.0
R_n_L_1000_2	266884.1	266884.1	8454.9	0.0	0.0
C_u_L_1000_1	265947.9	265947.9	4594.1	0.0	0.0
C_u_L_1000_2	265442.7	265465.5	4694.6	0.0	0.0
C_n_L_1000_1	267120.0	267560.0	5138.8	0.0	0.2
C_n_L_1000_2	265864.7	265864.7	4336.9	0.0	0.0
RC_u_L_1000_1	266456.9	266456.9	6052.1	0.0	0.0
RC_u_L_1000_2	265511.1	265511.1	6175.2	0.0	0.0
RC_n_L_1000_1	265920.5	265920.5	6651.8	0.0	0.0
RC_n_L_1000_2	267154.6	267154.6	4855.6	0.0	0.0
R_u_S_1000_1	717850.4	718032.0	7246.1	0.0	0.0
R_u_S_1000_2	717867.8	718035.6	8238.7	0.0	0.0
R_n_S_1000_1	785211.0	785211.0	7444.5	0.0	0.0
R_n_S_1000_2	756521.9	756521.9	8373.3	-0.1	-0.1
C_u_S_1000_1	728436.7	728436.7	4853.7	0.0	0.0
C_u_S_1000_2	713279.5	713965.8	4346.1	0.0	0.1
C_n_S_1000_1	763599.7	763599.7	4941.7	0.0	0.0
C_n_S_1000_2	725942.1	725942.1	4306.0	0.0	0.0
RC_u_S_1000_1	743705.8	743705.8	5523.9	0.0	0.0
RC_u_S_1000_2	715332.6	715332.6	6475.4	0.0	0.0
RC_n_S_1000_1	727614.1	727882.1	7441.2	0.0	0.0
RC_n_S_1000_2	764637.5	765476.9	5081.3	0.0	0.1