CAPACITATED FACILITY LOCATION PROBLEM WITH CUSTOMER AND FACILITY DIFFERENTIATION

by

Özlem Çavuş

B.S., Industrial Engineering, Boğaziçi University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2007

CAPACITATED FACILITY LOCATION PROBLEM WITH CUSTOMER AND FACILITY DIFFERENTIATION

APPROVED BY:

Prof. İ. Kuban Altınel .....
(Thesis Supervisor)

Assist. Prof. Deniz Aksen .....

Assoc. Prof. Necati Aras .....

DATE OF APPROVAL: 12.06.2007

ACKNOWLEDGEMENTS

First and foremost, I am grateful to my thesis supervisor Prof. Kuban Altınel for his continuous guidance, support and endless patience. Even in the hardest times, he never gave up on me, and made possible the constitution of this thesis through his leading comments and contributions. Apart from our thesis studies, he has always been the person whose suggestions I truly needed and whom I confided in.

I would like to express my gratitude to Assoc. Prof. Necati Aras and Assist. Prof. Deniz Aksen for taking time to examine this thesis and taking part in my thesis jury.

I would like to thank my friends Hande Küçükaydın, Bilgen Öztürk, Evrim Dalkıran, Gönül Tanuğur, Buket Avcı, Güray Güler, Engin Durmaz, and Mehmet Gönen for their friendship, for their help and company at my hardest times.

Along with these people, I wish to thank all Industrial Engineering faculty, assistants, and secretaries.

Finally, and above all, I would like to thank my parents and my brother, whose love and support I could always count on. This thesis is dedicated to them, and especially to my mother, whom I owe everything in my life.

This research has been partly supported by Boğaziçi University Research Fund Grant No: 06HA304D.

ABSTRACT

CAPACITATED FACILITY LOCATION PROBLEM WITH CUSTOMER AND FACILITY DIFFERENTIATION

This thesis focuses on an extension of the capacitated facility location problem. Every demand point consists of multiple customer classes whose demands are satisfied by facilities having different capacities and costs for each facility class. We give integer and mixed-integer linear programming formulations respectively for single-source and multi-source versions of this problem. Then we propose exact and heuristic solution procedures.

Capacitated facility location problems are difficult to solve exactly. However, many accurate and efficient heuristic methods have been introduced for these problems. In the light of these researches, we develop Lagrangean heuristics in order to find near optimal solutions for our problems. The Lagrangean dual problems are solved using three different subgradient optimization methods, namely classical subgradient optimization, deflected subgradient optimization and volume algorithm. The Lagrangean heuristics with the mentioned subgradient optimization methods are implemented and computational results based on extensive experiments are also provided.

Furthermore, an exact solution technique based on Benders' decomposition is developed and implemented for the multi-source problem. Although optimal solutions are found for some small test problems, this exact solution algorithm converges very slowly.

ÖZET

MÜŞTERİ SINIFLARINI GÖZ ÖNÜNE ALARAK EN UYGUN TESİSİ AÇAN TAMSAYI PROGRAMLAMA MODELİNİN YAKLAŞIK ÇÖZÜMÜ

Bu çalışmada, müşteri sınıfları ve tesis tipleri göz önünde bulundurularak, klasik sığa kısıtlı tesis yerseçimi problemini daha gerçekçi hala dönüştürmek amaçlanmaktadır. Müşteri istemleri sınıf farkları dikkate alınacak şekilde sığaları açıldıkları yere ve tiplerine bağlı tesislerden sağlanmaktadır. Bu problemin tek kaynaklı ve çok kaynaklı versiyonları için sırasıyla tamsayı ve karışık tamsayı programlama modelleri verilmekte ve her iki problem için çözüm yöntemleri önerilmektedir.

Sığa kısıtlı tesis yerseçimi problemlerinin en iyi çözümünü bulmak zordur. Sezgisel yöntemler kullanılarak bu problemler için etkin ve doğru sonuçlar elde edilmiştir. Bu çalışmalar doğrultusunda hareket edilerek, en iyi çözüme yakın çözümler bulmak için Lagrange sezgisel yöntemleri geliştirilmiştir. Lagrange ikili problemlerin çözümü için klasik altgradyan, saptırılmış altgradyan ve hacim algoritması olmak üzere üç farklı altgradyan yöntemi kullanılmıştır. Bütün bu yaklaşımlar rassal üretilmiş problemler üzerinde denenmiş ve sonuçlar verilmiştir.

Yaklaşık çözümlerin yanısıra, çok kaynaklı problemin en iyi çözümü bulmak için Benders ayrıştırması kaynaklı bir çözüm yöntemi geliştirilmiştir. Her ne kadar bazı küçük sınama problemleri için en iyi çözümler bulunmuşsa da, algoritma en iyi çözüme çok yavaş yaklaşmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

c_{ikjl}	Cost of serving all demand of a customer of class l at location j from a facility of class k at location i
d_{jl}	Total demand of class l customer at location j
f_{ik}	Fixed cost of opening a facility of class k at location i
g_{ikjl}	Fixed cost of serving a customer of class l at location j from a facility of class k at location i
i	Index of the facility locations
j	Index of the customer locations
k	Index of the facility classes
l	Index of the customer classes
s_{ik}	Capacity of class k facility at location i
u	Lagrange multiplier vector
v_{ik}	Binary variable indicating whether a facility of class k is opened at location i
x_{ikjl}	Binary variable indicating whether a customer of class l at location j is served by a facility of class k at location i
y_{ikjl}	Binary variable indicating whether a customer of class l at location j is served by a facility of class k at location i
μ	Step length parameter
λ	Step length
CFLP	Capacitated facility location problem
FLP	Facility location problem
IP	Integer programming
LB	Lower bound
LDP	Lagrangean dual problem
LP	Linear programming
LR	Lagrangean relaxation
LSP	Lagrangean subproblem

MCFLP	Multi-source capacitated facility location problem
MIP	Mixed integer programming
SCFLP	Single-source capacitated facility location problem
UB	Upper bound
UFLP	Uncapacitated facility location problem

1. INTRODUCTION

Facility Location Problems (FLPs) are combinatorial optimization problems formulated to locate new facilities such as retailers, warehouses or factories and to assign customers to these facilities. In general, the objective is to minimize the total cost, which consists of the fixed cost of opening a facility and variable cost of serving a customer. If there is no limit on the number of the customers that can be assigned to a facility, the problem becomes the Uncapacitated Facility Location Problem (UFLP). On the other hand, if a facility can serve a limited number of customers, the problem is called the Capacitated Facility Location Problem (CFLP). Both the UFLP and the CFLP are NP-hard problems [13] and many exact and heuristic solution approaches have been proposed for these problems in the literature [34].

In the CFLP, each facility has a capacity so there is a limit on the demand that each facility can supply. CFLP can be categorized into two categories: Single-Source Capacitated Facility Location Problem (SCFLP) and Multi-Source Capacitated Facility Location Problem (MCFLP). When there is a restriction that a customer can be served from only one facility, the problem becomes SCFLP which is also known as the simple plant location problem. This is also the case when there is no capacity restriction for the facilities, each customer is served from the closest facility. If there is no single-source restriction, meaning that a customer can be served from more than one facility, the problem is the MCFLP.

The problem we consider in this work is the Capacitated Facility Location Problem with Customer and Facility Differentiation and aims to locate facilities in such a way that each customer class is served from the appropriate facility class. It differs from conventional CFLPs in two aspects:

1. Conventional CFLPs assume that there is only one class of customers at a given location. However, this assumption may be unrealistic if customers with different profiles exist at the same location. Thus, in our problem we include customer dif-

ferentiation; customers at a location are differentiated according to their profiles.

2. In today's competitive environment, suppliers prefer opening facilities which are equipped according to the demand profiles of customers they serve. However, conventional CFLPs may not exactly meet the needs of these suppliers since they assume that facilities have the same class. The CFLP with Customer and Facility Differentiation assumes that there are different classes of facilities that can be opened at a location and aims to open a facility which is equipped according to the profiles of potential customers.

In this work, we consider both single-source and multi-source versions of this problem. In the single-source problem, each class of customer is served by only one facility, which is not necessary for the multi-source problem. We use the Lagrangean relaxation technique and develop a Lagrangean heuristic for both problems in order to find near optimal solutions. Then, we try to find an exact solution for the multi-source problem using Benders' decomposition method.

The rest of this thesis is organized as follows. Chapter 2 gives a brief overview of different capacitated facility location problem formulations and solution methods. New formulations with customer profiles are presented in Chapter 3. In Chapter 4, we remind the principles of Lagrangean relaxation and subgradient optimization. Different variants of subgradient methods are investigated in Chapter 5. Next, a new subgradient optimization method which was proposed for nonconvex minimization problems is described in Chapter 6. We provide a review of the Benders' decomposition method which can be used to find an exact solution for mixed-integer programming problems in Chapter 7. In Chapter 8, our solution methodology is explained and the experimental results are given in Chapter 9. Finally, Chapter 10 includes the general conclusions drawn from the research of this thesis.

2. LITERATURE SURVEY

The facility location problems are not new to the operations research society, they have been studied for many years. In the literature, different model formulations and solution methodologies are proposed for these problems. The formulations and solution methods vary depending on the assumptions, mathematical complexity and computational performance [29]. In this chapter, we first give a brief review of models for deterministic CFLPs and then summarize the solution methods which are proposed for these problems.

2.1. Formulations

In deterministic CFLPs, the input is assumed to be known with certainty. The input is generally the demands of customers, the capacities of facilities, the locations of customers, the candidate locations of facilities, variable and fixed costs. The problem formulations introduced in the literature range in complexity from single-product to multi-product, single-facility to multi-facility, single-source to multi-source and single-stage to two-stage models. First, we review the models for single-stage problems and then we briefly mention the models for two-stage ones. While in single-stage problems the products are transported from facilities to customers, in two-stage problems products flow from facilities to warehouses and from warehouses to customers.

Classical CFLPs assume that there is only one class of facility that can be opened at a candidate location. Agar and Salhi [1] work on a problem called multi-capacitated facility location problem (multi-CFLP) where each facility has different possible capacities and corresponding fixed costs. Both multi-source and single-source problems are considered in this study and their formulations are presented. In these formulations, a set N_k is defined as the set of possible capacities for each facility site k. Then an additional constraint, which depends on this set definition and restricts that at most one class of facility is opened at each site, is added to the classical CFLP formulation. Correia and Captivo [14] present a different formulation, which does not require a set

definition, for this problem.

Mazzola and Neebe [33] introduce a model for a problem named multi-product CFLP with choice of facility type, where demand for a number of different product families is supplied from a set of facility sites. Each site offers a common series of facility classes with different capacities. The capacities of facilities only depend on the class of the facility, they do not depend on the site. The model decides on both the class of the facility that will be opened at a site and the product composition that each opened facility produces. Different from our problem, this problem takes into consideration that a customer demand may be composed of different product families. However, it ignores the existence of different customer profiles. In addition, in our problem the capacities of facilities depend not only on the class of the facility but also on the site.

The location problems we have considered so far are single-stage problems. Klose [30] presents a two-stage capacitated facility location problem which aims to find the optimal locations of warehouses in a distribution network where products flow from facilities to warehouses and from warehouses to customers. Two mixed-integer models are introduced. In one of them unit transportation cost has three indices: c_{ijk} is the unit cost of transporting products from facility i to customer k through warehouse j. In the other model, this cost is split into two linear parts. The first part deals with the transport from facilities to warehouses and the second one deals with the transport from warehouses to customers. Although the first formulation is more realistic, the second one is preferable since it is tractable. The formulations of Klose [30] assume that there is only one type of product. On the other hand, the second formulation, where the transportation cost is split into two, has already been extended to include many product types in the earlier works by Geoffrion and Graves [20] and by Bramel and Simchi-Levi [9].

Here, we have only provided a review of models for deterministic CFLPs. Further literature on plant location models can be seen in the review papers presented by Klose and Drexl [29], ReVelle and Eiselt [35], ReVelle and Laporte [36].

2.2. Solution Methods

Both MCFLP and SCFLP are integer-programming (IP) problems belonging to the class of NP-hard problems. Many hard integer-programming problems, including the CFLP, consist of an easy problem which is complicated by addition of a small set of side constraints. Thus, dualizing these constraints by a method called Lagrangean relaxation provides an easy problem that can be solved in polynomial time. Lagrangean relaxation combined with subgradient optimization can give good lower bounds for the optimal objective value of these kinds of problems. Lagrangean heuristics generally include upper bound heuristics which use Lagrangean relaxation solutions in order to construct feasible solutions to the original problem, so provide upper bounds for the optimal objective value of the original problem [16]. Since Lagrangean heuristics give good bounds at reasonable computation time, this approach has been widely used to solve the CFLP.

Cornuéjols, Sridharan, and Thizy [12] compare heuristics and relaxations proposed in the literature for CFLP. Their computational results reveal that relaxing demand constraints or capacity constraints in a Lagrangean fashion provides better bounds than other relaxations. They test a Lagrangean heuristic, a greedy heuristic, and an interchange heuristic on a set of test problems and observe that Lagrangean heuristic gives better solutions and is also superior in terms of computational time.

Beasley [5] presents a framework for developing Lagrangean heuristics for location problems. In his research, both capacity constraints and customer demand constraints are dualized. However, it is common for CFLP applications to dualize either capacity constraints or customer demand constraints since the quality of the bounds obtained by Lagrangean relaxation decreases or remains same as the number of dualized constraints increases [12].

Agar and Salhi [1] describe Lagrangean heuristics for a variety of CFLPs. Their solution procedure is similar to the one described by Beasley [5]. They adopt this procedure to solve multi-source multi-CFLP and single-source multi-CFLP, which are

not addressed much in the literature. They model these problems via adding an additional constraint, which allows the opening of at most one class of facility at a given site, to their classical CFLP formulations. In order to obtain a feasible solution, first a solution ignoring the additional constraint is obtained. This is essentially solving a classical CFLP. If the solution obtained does not satisfy the additional constraint, then a procedure is utilized to make the solution feasible.

Hindi and Pieńkosz [24] combined Lagrangean heuristic with restricted neighborhood and steepest descent searches to solve SCFLP. The proposed upper bound heuristic, which includes the mentioned search procedures, provides good upper bounds. However, it is computationally expensive. Therefore, for large problems, the restricted neighborhood and steepest descent searches are carried out only at the subgradient iterations that lead to an improvement of the lower bound. Cortinhal and Captivo [15] presented a different Lagrangean heuristic method for SCFLP. Their upper bound heuristic consists of two phases. In the first phase a simple heuristic is used to find a feasible solution from the lower bound solution. Then, in the second phase this solution is improved by implementing tabu search or local search procedures. Ahuja et al. [2] introduced a neighborhood search algorithm for the single-source problem, where the search is induced by customer multi-exchanges and facility moves.

Mazzola and Neebe [33] presented a Lagrangean heuristic and a branch and bound algorithm for the multi-product CFLP with choice of facility type. The proposed branch and bound algorithm utilizes bounds obtained by the Lagrangean heuristic described in their paper. Given computational results reveal that branch and bound algorithm solves all of the test problems to optimality, while Lagrangean heuristic generates solutions on the average within 2% of the optimal. On the other hand, Lagrangean relaxation is much more superior in terms of computational time.

Klose [29] developed a linear programming heuristic for his two-stage CFLP. At each iteration a lower bound is obtained via solving the linear programming (LP) relaxation of the problem. Then, valid inequalities and facets are added to the LP formulation iteratively in order to improve the lower bound which is provided at each

iteration. After each re-optimization step, that is the recalculation of the LP solution after the addition of valid inequalities, feasible solutions are obtained from the current LP solution by applying simple heuristics. Good lower and upper bounds were obtained by this algorithm. Bramel and Simchi-Levi [9] presented a Lagrangean relaxation based solution procedure for a similar problem, where product types are included, in order to obtain bounds for the optimal objective value.

In their pioneering work, Geoffrion and Graves [20] used Benders' decomposition algorithm to find an optimal solution for their two-stage CFLP. When the binary variables are fixed at a feasible value, the remaining LP problem separates into classical transportation problems which are much easier to solve. The algorithm was applied to a set of real life problems and some improvement methods were proposed in order to increase the efficiency of the algorithm.

3. PROBLEM FORMULATION

In this part, the mathematical programming formulations of both single-source and multi-source problems are given. Some common assumptions are made in order to formulate the problems. In both models, it is assumed that the locations of customers and potential facility locations are predetermined. Furthermore, customer demands and facility capacities are also known and they are deterministic. Since each facility has a capacity, there is a restriction on the demand that a facility can satisfy.

3.1. Single-Source Problem

Firstly, we will consider Single-Source CFLP with Customer and Facility Differentiation, where a customer can be served from only one facility. The aim of this problem is to locate facilities in such a way that each customer class is served from appropriate facility class. The model tries to minimize the sum of two types of incurred costs:

1. Fixed cost of opening a facility, which depends on the candidate location and class of a facility.
2. Cost of serving a customer class, which depends on facility class, customer class, and the distance between facility and assigned customer.

The indices and parameters used in the model are defined as follows:

I : the number of potential facility locations

J : the number of customer locations

K : the number of facility classes

L : the number of customer classes

f_{ik} : fixed cost of opening a facility of class k at location i

c_{ikjl} : cost of serving all demand of a customer of class l at location j from a facility of class k at location i

s_{ik} : capacity of class k facility at location i

d_{jl} : total demand of class l customer at location j

The decision variables used in the model are as follows:

v_{ik} : binary variable indicating whether a facility of class k is opened at location i

x_{ikjl} : binary variable indicating whether a customer of class l at location j is served by a facility of class k at location i

The model can be formulated as:

$$P1: \quad z_{IP} = \min \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L} c_{ikjl}x_{ikjl} + \sum_{i=1}^{I}\sum_{k=1}^{K} f_{ik}v_{ik} \tag{3.1}$$

s.t.

$$\sum_{i=1}^{I}\sum_{k=1}^{K} x_{ikjl} = 1 \qquad j = 1,\ldots,J; l = 1,\ldots,L \tag{3.2}$$

$$\sum_{j=1}^{J}\sum_{l=1}^{L} d_{jl}x_{ikjl} \le s_{ik}v_{ik} \quad i = 1,\ldots,I; k = 1,\ldots,K \tag{3.3}$$

$$\sum_{k=1}^{K} v_{ik} \le 1 \qquad i = 1,\ldots,I \tag{3.4}$$

$$v_{ik} = 0,1 \qquad i = 1,\ldots,I; k = 1,\ldots,K \tag{3.5}$$

$$x_{ikjl} = 0,1 \qquad i = 1,\ldots,I; k = 1,\ldots,K; j = 1,\ldots,J; \quad l = 1,\ldots,L \tag{3.6}$$

The objective (3.1) minimizes the sum of fixed costs and service costs. Constraints (3.2) are demand constraints which ensure that all demand of customer of class l at location j is met. Since the problem is a capacitated problem, capacity constraints should be also included in the model. Constraints (3.3) are capacity constraints, indicating that total number of units supplied from a facility cannot be bigger than its capacity. Because the right hand side of the constraints are multiplied with the decision variable v_{ik}, these constraints also ensure that a customer cannot be assigned to a facility which is not opened. It is assumed that at most one facility can be opened at a

potential location. This assumption is satisfied by constraints (3.4). They provide the selection of at most one facility class from K facility classes at each potential location. Constraints (3.5) and (3.6) are integrality constraints.

3.2. Multi-Source Problem

In this problem, a customer can be served from more than one facility, which is different from the single-source problem. In addition to the costs incurred in the single-source model, objective function of this model includes fixed cost of assigning a customer with a specific profile to a facility of a specific class. This cost takes into account the class mismatch between customers and facilities. When a customer class is assigned to an appropriate facility class, less cost is incurred than assigning these customers to an inappropriate facility class.

The indices and parameters used in the model are defined as follows:

I : the number of potential facility locations

J : the number of customer locations

K : the number of facility classes

L : the number of customer classes

f_{ik} : fixed cost of opening a facility of class k at location i

g_{ikjl} : fixed cost of serving a customer of class l at location j from a facility of class k at location i

c_{ikjl} : cost of serving all demand of a customer of class l at location j from a facility of class k at location i

s_{ik} : capacity of class k facility at location i

d_{jl} : total demand of class l customer at location j

The decision variables used in the model are as follows:

v_{ik} : binary variable indicating whether a facility of class k is opened at location i

x_{ikjl} : the fraction of demand of a customer of class l at location j supplied from a facility of class k at location i

y_{ikjl} : binary variable indicating whether a customer of class l at location j is served by a facility of class k at location i

The model can be formulated as below:

$$P2: \quad z_{MIP} = \min \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L} c_{ikjl}x_{ikjl} + \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L} g_{ikjl}y_{ikjl} + \sum_{i=1}^{I}\sum_{k=1}^{K} f_{ik}v_{ik} \tag{3.7}$$

s.t.

$$\sum_{i=1}^{I}\sum_{k=1}^{K} x_{ikjl} = 1 \qquad j = 1,\ldots,J; l = 1,\ldots,L \tag{3.8}$$

$$\sum_{j=1}^{J}\sum_{l=1}^{L} d_{jl}x_{ikjl} \le s_{ik} \qquad i = 1,\ldots,I; k = 1,\ldots,K \tag{3.9}$$

$$\sum_{k=1}^{K} v_{ik} \le 1 \qquad i = 1,\ldots,I \tag{3.10}$$

$$y_{ikjl} \le v_{ik} \qquad i = 1,\ldots,I; k = 1,\ldots,K; \quad j = 1,\ldots,J; l = 1,\ldots,L \tag{3.11}$$

$$x_{ikjl} \le y_{ikjl} \qquad i = 1,\ldots,I; k = 1,\ldots,K; \quad j = 1,\ldots,J; l = 1,\ldots,L \tag{3.12}$$

$$x_{ikjl} \ge 0 \qquad i = 1,\ldots,I; k = 1,\ldots,K; \quad j = 1,\ldots,J; l = 1,\ldots,L \tag{3.13}$$

$$y_{ikjl} = 0,1 \qquad i = 1,\ldots,I; k = 1,\ldots,K; \quad j = 1,\ldots,J; l = 1,\ldots,L \tag{3.14}$$

$$v_{ik} = 0,1 \qquad i = 1,\ldots,I; k = 1,\ldots,K \tag{3.15}$$

The objective function (3.7) minimizes the sum of fixed cost of opening a facility, fixed cost of assigning a customer to a facility, and service cost. Fixed cost of assigning a customer can also be included in the single-source model. However, since x_{ikjl}'s are binary variables in the single-source model, they represent the same thing as y_{ikjl}'s. Therefore, there is no need to use both decision variables in the model. As a result,

c_{ikjl} can be thought as a sum of service cost and fixed cost of assigning a customer to a facility in (3.1).

Constraints (3.8) and (3.9) are demand constraints and capacity constraints respectively. (3.10) provides that at most one facility class from K facility classes is opened at each potential location. Constraints (3.11) require that serving decision from a facility is valid only when such a facility is opened. (3.12) ensures that a customer can be served from a facility if it is assigned to this facility. Constraints (3.13) are non-negativity constraints, (3.14) and (3.15) are integrality constraints.

4. LAGRANGEAN RELAXATION AND CLASSICAL SUBGRADIENT OPTIMIZATION

Many integer-programming problems including the CFLP are in the class of hard problems for which all known algorithms require exponential time in the worst-case. Therefore, obtaining upper and lower bounds as close as possible to the optimal objective value is important for the efficient solution of these problems. Upper bounds are essentially generated by means of heuristic methods. There are different techniques to generate the lower bounds, two well-known methods among these are Linear programming (LP) and Lagrangean (LR) relaxations.

The idea of Lagrangean relaxation was first developed in 1970 by Held and Karp [23] in their study on the traveling salesman problem. Many hard integer programming problems can be reduced to an easy problem through adding complicating constraints to the objective function as penalty terms. The resulting relaxation problem is called the Lagrangean subproblem (LSP), which provides a lower bound for the optimal objective value of the original problem.

Consider the following linear integer-programming problem (IP):

$$\begin{aligned} IP: \quad z_{IP} &= \min \ c^T x && (4.1)\\ \text{s.t.} \quad & Ax \geq b && (4.2)\\ & Bx \geq d && (4.3)\\ & x \in Z_+^n && (4.4) \end{aligned}$$

where $c \in R^n$, (A, b) and (B, d) are $m \times (n+1)$ and $r \times (n+1)$ matrices respectively, finally x is an n-dimensional vector of non-negative integers. The above problem IP is called the primal problem and its optimal solution is called the primal solution. Suppose that constraints (4.2) are complicating constraints which make the primal

problem hard to solve and the following problem which excludes these constraints,

$$\min \left\{ c^T x : Bx \geq d, x \in Z_+^n \right\} \tag{4.5}$$

can easily be solved.

The complicating constraints (4.2) can be added to the objective function with penalty cost vector $u \in R_+^m$, known as Lagrange multiplier vector, and the following Lagrangean subproblem can be obtained:

$$\begin{aligned} LSP: \quad z_{LR}(u) &= \min \ c^T x + u(b - Ax) && (4.6) \\ \text{s.t.} \quad & Bx \geq d && (4.7) \\ & x \in Z_+^n && (4.8) \end{aligned}$$

Let x be any feasible solution of the primal problem (IP). Notice that for any $u \in R_+^m$ and any optimal solution x^* of (IP) we obtain,

$$z_{LR}(u) \leq c^T x + u(b - Ax) \leq c^T x \text{ and } z_{LR}(u) \leq c^T x^* + u(b - Ax^*) \leq c^T x^* = z_{IP}.$$

Using the fact $z_{LR}(u) \leq z_{IP}$, it can be said that the optimal solution of the Lagrangean subproblem provides a lower bound for the optimal objective value of the primal problem (IP). Different lower bounds can be obtained by changing the values of Lagrange multipliers u. Therefore, in order to obtain the largest lower bound, the following Lagrangean dual problem (LDP) is solved:

$$\begin{aligned} LDP: \quad z_{LD} &= \max_{u \geq 0} \left\{ z_{LR}(u) = \min \ c^T x + u(b - Ax) \right\} && (4.9) \\ \text{s.t.} \quad & Bx \geq d && (4.10) \\ & x \in Z_+^n && (4.11) \end{aligned}$$

It has been shown by Geoffrion [19] that Lagrangean dual problem provides better lower bounds than LP relaxation of the primal problem if the Lagrangean subprob-

lem does not have the integrality property. Otherwise, the bounds obtained by both relaxations are equal to each other. Lagrangean relaxation is also preferable to LP relaxation in the case where the primal problem includes complicating constraints or large number of constraints and variables. The LP relaxation problem is generally solved using a simplex based procedure. On the other hand, the Lagrangean subproblem can usually be decomposed to small problems which can be solved using heuristic methods. Furthermore, although the solution of the Lagrangean subproblem may not be feasible for the original problem, this solution can easily be modified by a heuristic method to be feasible. Since a feasible solution to a minimization problem gives an upper bound for that problem, we can obtain both an upper bound and a lower bound for the primal problem through Lagrangean heuristics.

Beasley [6] states that two key issues determine the quality of the lower bound obtained from Lagrangean relaxation: deciding which constraints to dualize; and deciding how to find numerical values for Lagrange multipliers. The values of Lagrange multipliers can be decided by using subgradient optimization or multiplier adjustment methods [6]. Subgradient optimization is an iterative procedure which, from an initial set of Lagrange multipliers, generates further multipliers in a systematic fashion and provides a solution to the Lagrangean dual problem. It attempts to maximize the lower bound obtained from Lagrangean relaxation by a suitable choice of multipliers. Subgradient optimization combined with Lagrangean relaxation gives good lower bounds for many combinatorial optimization problems, so it has been widely used in the literature. Multiplier adjustment changes a single multiplier in each iteration in contrast to subgradient optimization. It is computationally cheaper and it produces an improvement (or at least no deterioration) in the lower bound at each iteration. However, the quality of the lower bound obtained is not as good as the one obtained from subgradient optimization and different problems require different multiplier adjustment algorithms. On the other hand, subgradient optimization can directly be applied to many different problems.

The *classical subgradient optimization* is an adaptation of the gradient method in which a gradient vector is replaced by a subgradient vector at a point where the gradient

Table 4.1. The Subgradient Optimization Algorithm

Step 0: (Initialization) Initialize the Lagrange multiplier vector $u \in R_+^m$ and set $k = 1$.

Step 1: Solve the Lagrangean subproblem:

$$z_{LR}(u^{(k)}) = \min \; c^T x + u^{(k)}(b - Ax) \tag{4.12}$$

$$\text{s.t.} \quad Bx \geq d \tag{4.13}$$

$$x \in Z_+^n \tag{4.14}$$

Let $x^{(k)}$ be the solution of the subproblem. Then find the subgradient vector $s^{(k)}$ at $u^{(k)}$,

$$s^{(k)} = b - Ax^{(k)}$$

Step 2: If $s^{(k)} \leq 0$ and $u^{(k)}s^{(k)} = 0$, then $z_{LR}(u^{(k)}) = z_{IP}$ and STOP. This means that $x^{(k)}$ and $u^{(k)}$ are optimal solutions for the primal and Lagrangean dual problems respectively. Otherwise, go to Step 3.

Step 3: Update the Lagrange multiplier vector such that $u^{(k+1)} = P_{R_+^m}(u^{(k)} + \lambda^{(k)}s^{(k)})$, where

$$P_{R_+^m}(u) = \begin{cases} u_i & if \; u_i \geq 0 \\ 0 & otherwise. \end{cases} \tag{4.15}$$

$\lambda^{(k)} \geq 0$ is a step length calculated as:

$$\lambda^{(k)} = \frac{\mu^{(k)}(UB - LB)}{\left\| s^{(k)} \right\|^2}$$

where $\mu^{(k)}$ is step length parameter satisfying $0 < \mu^{(k)} \leq 2$, UB and LB are upper and lower bounds on optimal objective value. Set $k = k + 1$, and go to Step 1.

does not exist. The steps of the method are explained in Table 4.1. The stopping condition in Step 2 can only be satisfied if the duality gap does not exist. However, this is not usually the case for many integer programming problems. Therefore some alternative stopping conditions can be used. One possible way is stopping the algorithm after a predetermined number of iterations are completed. Another way is reducing the value of step length parameter μ during the procedure and stopping when this value gets below a threshold value.

If an equality constraint is relaxed in a Lagrangean fashion instead of an inequality constraint, then there is no need to have nonnegative Lagrange multiplier values. In this case, we do not use the projection (4.15) in Step 3.

5. IMPROVEMENTS IN THE SUBGRADIENT OPTIMIZATION

In this chapter, some improvements in the subgradient optimization method, which exist in the literature are explained. The *classical subgradient optimization* can result in slow convergence if the current subgradient vector forms an obtuse angle with the previous direction of motion and such a phenomenon is called *zigzagging*. The *deflected subgradient optimization* method is developed in order to overcome this phenomenon. Barahona and Anbil [3] present an extension of the subgradient algorithm that produces an approximate primal solution. This method is called *volume algorithm* and it uses the information generated in the process of deflected subgradient. Here, we only give a brief review of these methods. A detailed review including the optimality conditions and convergence properties can be found in the study presented by Guta [22].

5.1. Deflected Subgradient Optimization Method

Studies of Bazaraa et al. [4], Camerini et al. [11], Sherali and Ulular [41] show that during the subgradient optimization procedure current subgradient direction can form an obtuse angle with the previous direction of motion as the iterates progress. As a result, the distance between the next Lagrange multiplier and the optimal Lagrange multiplier value is closed only with a small step, therefore this case may result in slow convergence of the subgradient procedure. This phenomenon which is called zigzagging is explained in Figure 5.1. In order to overcome this behavior, the subgradient direction is deflected whenever it forms an obtuse angle with the previous direction. Camerini et al. [11] propose a method to find the deflected direction given by,

$$d^{(k)} = s^{(k)} + \delta^{(k)} d^{(k-1)} \tag{5.1}$$

where $s^{(k)}$ is the subgradient direction found at iteration k, $d^{(k)}$ is the deflected subgradient direction and $\delta^{(k)} \geq 0$ is the deflection parameter. The deflection parameter is defined as:

$$\delta^{(k)} = \begin{cases} -\tau^{(k)} \frac{s^{(k)} d^{(k-1)}}{\left\| d^{(k-1)} \right\|^2} & if\ s^{(k)} d^{(k-1)} < 0 \\ 0 & otherwise \end{cases} \tag{5.2}$$

with $1 \leq \tau^{(k)} < 2$.

Table 5.1. The Deflected Subgradient Optimization Algorithm

Step 0: (Initialization) Initialize the Lagrange multiplier vector u, set $k = 1$ and $d^{(k-1)} = 0$.

Step 1: Solve the Lagrangean subproblem and find the subgradient vector $s^{(k)}$ at $u^{(k)}$.

Step 2: If $s^{(k)} \leq 0$ and $u^{(k)} s^{(k)} = 0$, then $z_{LR}(u^{(k)}) = z_{IP}$ and STOP. Otherwise, go to Step 3.

Step 3: Calculate the deflected subgradient direction:

$$d^{(k)} = s^{(k)} + \delta^{(k)} d^{(k-1)}$$

Update the Lagrange multiplier vector such that $u^{(k+1)} = P_{R_+^m}(u^{(k)} + \lambda^{(k)} d^{(k)})$, where

$$P_{R_+^m}(u) = \begin{cases} u_i & if\ u_i \geq 0 \\ 0 & otherwise. \end{cases} \tag{5.3}$$

$\lambda^{(k)} \geq 0$ is a step length calculated as:

$$\lambda^{(k)} = \frac{\mu^{(k)}(UB - LB)}{\left\| d^{(k)} \right\|^2}$$

where $\mu^{(k)}$ is step length parameter satisfying $0 < \mu^{(k)} < 1$. Set $k = k + 1$, and go to Step 1.

Notice that the deflected direction is a linear combination of previous direction

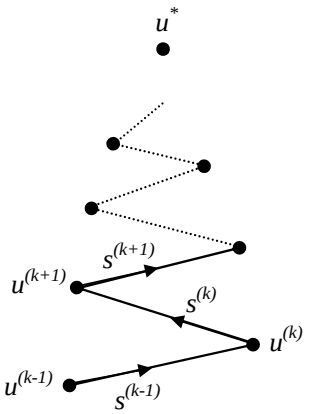

Figure 5.1. The zigzagging phenomenon in the classical subgradient optimization method.

and current subgradient direction. If the deflection parameter is taken as in (5.2), the new direction always forms an acute angle with the previous direction, therefore zigzagging is avoided. If the subgradient direction $s^{(k)}$ forms an acute angle with the previous direction $d^{(k-1)}$, then $\delta^{(k)}$ is zero and the deflected direction is equal to the current subgradient direction $s^{(k)}$ (see Figure 5.2.(a)). Otherwise if $s^{(k)}$ forms an obtuse angle then it is deflected (see Figure 5.2.(b)). During the deflected subgradient procedure the sequence $\left\{\left\|u^{(k)} - u^*\right\|\right\}$ decreases strictly, therefore a point which gets closer and closer to an optimal Lagrange multiplier is obtained at each iteration [22].

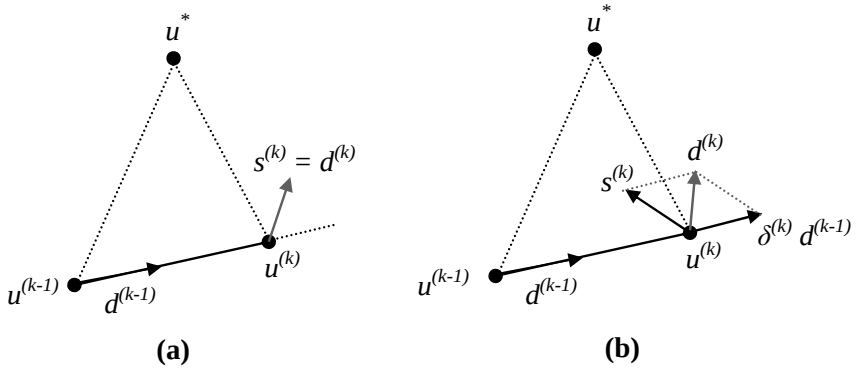

Figure 5.2. (a) $s^{(k)}$ forms an acute angle with previous direction $d^{(k-1)}$ so it is not deflected. (b) $s^{(k)}$ forms an obtuse angle with $d^{(k-1)}$ therefore it is deflected.

Table 5.1 summarizes the steps of the deflected subgradient optimization method. As in the classical subgradient optimization, the ideal stopping condition in Step 2 cannot be satisfied in general. Therefore, the stopping conditions proposed for the

classical method can also be used in the deflected subgradient method. The choice of $\tau^{(k)}$ is also important and computational results show that taking $\tau^{(k)} = 1.5$ gives good results [11].

5.2. Volume Algorithm

Although the classical subgradient optimization is an attractive method due to its low computational cost, it has some disadvantages. This method usually provides good approximations to optimal dual variables however it does not produce values for the primal variables. Another drawback is that classical subgradient algorithm does not have a well-defined stopping condition. In order to solve these problems, Barahona and Anbil [3] present an extension of the subgradient algorithm called the *volume algorithm* that produces an approximation to a primal solution. While this method preserves the low computational cost of the classical subgradient method, it also provides well-defined stopping criterion.

The idea behind the volume algorithm depends on the Lagrangean dual problem formulation stated in the following theorem by Geoffrion [19]:

Theorem 5.1 *[19] Given a problem (IP) (see Chapter 4), the Lagrangean dual problem (LDP) (see Chapter 4) can be written as:*

$$\begin{aligned} z_{LD} &= \min\ c^T x \\ s.t. &\quad Ax \geq b \\ &\quad x \in conv(X) = \left\{x \in Z_+^n : Bx \geq d\right\}. \end{aligned} \tag{5.4}$$

The Lagrangean dual problem in Theorem 5.1 is equivalent to the following problem,

$$
\begin{aligned}
z_{LD} \quad &= \quad \min \ c^T x \qquad (5.5)\\
\text{s.t.} \quad & \quad Ax \geq b\\
& \quad x \in conv(X) = \left\{ x : x = \sum_{s=1}^{S} \alpha_s x^s, \sum_{s=1}^{S} \alpha_s = 1, \alpha_s \geq 0, x^s \in X \right\}
\end{aligned}
$$

where $\left\{ x^s \in Z_+^n : s \in S \right\}$ is the set of extreme points of conv(X).

If we insert $\sum_{s=1}^{S} \alpha_s x^s$ in the place of x, we can write the problem (5.5) as:

$$
\begin{aligned}
z_{LD} \quad &= \quad \min \ \sum_{s=1}^{S} (c^T x^s)\alpha_s \qquad (5.6)\\
\text{s.t.} \quad & \quad \sum_{s=1}^{S} (Ax^s - b)\alpha_s \geq 0\\
& \quad \sum_{s=1}^{S} \alpha_s = 1\\
& \quad \alpha_s \geq 0 \qquad s = 1, ..., S.
\end{aligned}
$$

The problem (5.6) is also the master problem in Dantzig-Wolfe decomposition. Taking the dual of (5.6), we obtain the following problem:

$$
\begin{aligned}
& \text{maximize } \eta \qquad (5.7)\\
\text{s.t.} \quad & \eta + u(Ax^s - b) \leq c^T x^s \qquad s = 1, ..., S.
\end{aligned}
$$

The basic idea of the volume algorithm comes from applying the theorem of volume and duality due to Barahona and Anbil [3] to problem (5.7). This theorem tells that in a neighborhood of an optimal solution of (5.7), the subproblem (LSP) (see Chapter 4) produces a face s with probability α_s, which is an optimal solution of the master problem (5.6) in Dantzig-Wolfe decomposition. The volume algorithm is developed as an extension of the classical subgradient algorithm in order to estimate these probabilities. The steps of the volume algorithm are described in Table 5.2. Let

$x^{(0)}$, ...,$x^{(k)}$ be the solutions of the Lagrangean subproblem at iteration k. Then,

$$\overline{x} = \beta x^{(k)} + (1-\beta)\beta x^{(k-1)} + ... + (1-\beta)^k x^{(0)}. \tag{5.8}$$

As you can see that $\overline{x}$ is a convex combination of the solutions of Lagrangean subproblem. The algorithm uses the coefficients β, $(1-\beta)\beta$,...,$(1-\beta)^k$ as an approximate optimal solutions of the master problem (5.6), that is to say the approximate values of α_s.

Table 5.2. The Volume Algorithm

Step 0: (Initialization) Initialize the Lagrange multiplier vector u and solve the Lagrangean subproblem (LSP). Let $\overline{x}$ and $\overline{z}$ be the optimal solution and optimal objective function of the subproblem respectively. Set $x^{(0)} = \overline{x}$, $z^{(0)} = \overline{z}$, $k = 1$.

Step 1: Compute $d^{(k)} = b - A\overline{x}$ and $u^{(k)} = P_{R^m_+}(\overline{u} + \lambda^{(k)} d^{(k)})$ for the step size

$$\lambda^{(k)} = \frac{\mu^{(k)}(UB - \overline{z})}{\|d^{(k)}\|^2} \tag{5.9}$$

where $\mu^{(k)}$ is step length parameter satisfying $0 < \mu^{(k)} \leq 2$. Solve the subproblem with $u^{(k)}$, let $x^{(k)}$ and $z_{LR}(u^{(k)})$ be the solutions obtained. Then $\overline{x}$ is updated as,

$$\overline{x} = \beta x^{(k)} + (1-\beta)\overline{x} \tag{5.10}$$

where β is a number between 0 and 1.

Step 2: If $z_{LR}(u^{(k)}) > \overline{z}$ update $\overline{u}$ and $\overline{z}$ as,

$$\overline{u} = u^{(k)}, \qquad \overline{z} = z_{LR}(u^{(k)}). \tag{5.11}$$

Let $k = k + 1$ and go to Step 1.

In order to determine the value of β Barahona and Anbil suggest to set it to a

fixed value for a number of iterations and then decrease. Another method they suggest is to find the values by solving the following problem:

$$\begin{aligned} \min \quad & \left\| b - A(\beta x^{(k)} + (1-\beta)\overline{x}) \right\| \\ \text{s.t.} \quad & \frac{a}{10} \leq \beta \leq a \end{aligned} \tag{5.12}$$

The value of a is initialized to 0.1, and after every 100 iterations it is checked whether $\overline{z}$ has increased by at least 1 %, if not a is divided by 2. When a becomes less than 10^{-5}, it is kept constant [22]. The algorithm stops when $\|d\|$ and $|c\overline{x} - \overline{z}|$ are both below a certain threshold. Barahona and Anbil claim that they provided a well-defined stopping condition for their algorithm. However, it may not be an easy subject to find good thereshold values which are common for many problem sets. Therefore, it may be better to stop the algortihm when $\mu^{(k)}$ becomes smaller than a predefined value.

Barahona and Anbil propose the following method in order to update the step length parameter $\mu^{(k)}$ during the algorithm:

- If the lower bound is not improved, this iteration is called *red*. After a sequence of 20 red iterations μ is decreased by 0.66 ($\mu^{(k)} = 0.66\mu^{(k-1)}$).
- If the lower bound is improved, we compute

$$\theta = d^{(k)} s^{(k)} \tag{5.13}$$

where $d^{(k)}$ is the direction of movement and $s^{(k)} = b - Ax^{(k)}$.

- If $\theta < 0$, this iteration is called *yellow* and the value of μ is not changed.

- If $\theta > 0$, this iteration is called *green* and the value of μ is increased by 1.1 ($\mu_{(k)} = 1.1\mu_{(k-1)}$).

6. AUGMENTED LAGRANGEAN AND THE MODIFIED SUBGRADIENT ALGORITHM BASED ON FEASIBLE VALUES

Classical Lagrangean method works for some special classes of constrained optimization problems. It may fail to find a zero duality gap for a nonconvex problem. Augmented Lagrangean method is therefore introduced in order to overcome this drawback [8], [25], [37], [38].

The dual problem obtained via augmented Lagrangean can be solved using subgradient methods. Gasimov [17] studies nonconvex continuous minimization problems with equality constraints and proposes a subgradient method called *modified subgradient (MSG) algorithm* that solves the dual problem obtained via sharp augmented Lagrangean. The convergence properties of the algorithm are provided in [17] and [10]. This algorithm guarantees zero duality gap for a wide class of nonconvex optimization problems. On the other hand the MSG algorithm uses the global minimum of the augmented Lagrangean problem and the approximate upper bound of the primal problem in order to update step length parameters at each iteration. This drawback is also shared in many subgradient algorithms. Gasimov et al. [18] then propose a new subgradient method (F-MSG) based on feasible values, which does not require to know the optimal value of the primal problem and seeks it iteratively beginning from an arbitrary value. In addition, global minimum of the augmented Lagrangean problem is not essential to update the step length parameters. The performance of the algorithm is tested on a set of nonconvex test problems and also the convergence properties of the new algorithm are given [18].

Gasimov et al. devise the F-SMG algorithm to solve the dual problem obtained via sharp augmented Lagrangeans which are introduced by Rockafellar and Wets [39].

Consider the nonlinear programming problem:

$$\begin{aligned} P: \quad z &= \min \ f_0(x) \\ \text{s.t.} \quad & f(x) = 0 \\ & x \in X \end{aligned} \tag{6.1}$$

where X is a subset of R^n, $f_0 : R^n \to R$ and $f_0 : R^n \to R^m$ are given functions. Let $\|.\|$ and $\langle ., . \rangle$ denote the Euclidean norm and the inner product respectively. Then, the sharp augmented Lagrangean associated with problem (6.1) is given by,

$$L(x, u, c) = \min \ f_0(x) + c \|f(x)\| - \langle u, f(x) \rangle . \tag{6.2}$$

Here, $u \in R^m$ and $c \in R_+$ are the Lagrange multipliers and c penalty parameter respectively. The associated dual function is defined by,

$$H(u, c) = \min_{x \in X} \ [f_0(x) + c \|f(x)\| - \langle u, f(x) \rangle] . \tag{6.3}$$

Then we can write the dual problem for problem (6.1) as,

$$P^*: \quad \max_{(u,c) \in R^m \times R_+} \ H(u, c). \tag{6.4}$$

The F-MSG algorithm which is devised by Gasimov et al. to solve the dual problem (6.4) is explained in Table 6.1. The algorithm starts with an arbitrary H value, and solves the constraint satisfaction problem in Step 2. The update methodology of H depends whether the problem (6.5) is feasible or not. If a feasible solution is not obtained, then H is increased by Δ. When a feasible solution is found, the value of $f(x_k)$ is checked. If $f(x_k) = 0$, meaning that a feasible solution for the original problem 6.1 is obtained, H is updated as $H_{n+1} = \min \{f_0(x_k^n), H_n - \Delta_{n+1}\}$. Otherwise, the value of H is not changed and the Lagrange multipliers and penalty parameter are updated.

Table 6.1. F-MSG Algorithm

Step 0: Choose positive numbers ϵ_1, ϵ_2, Δ_1 and a number H_1. Set $n = 1$, $p = 0$, $q = 0$.

Step 1: Then initialize the Lagrange multipliers and penalty parameter such that $(u_1^n, c_1^n) \in R^m \times R_+$. Set $k = 1$, $u^{(k)} = u_1^n$, $c^{(k)} = c_1^n$.

Step 2: Given $(u^{(k)}, c^{(k)})$, solve the following constraint satisfaction problem:

$$\text{Find an element } x \in X \text{ such that } f_0(x) + c^{(k)} \|f(x)\| - \left\langle u^{(k)}, f(x) \right\rangle \leq H_n. \quad (6.5)$$

If a solution does not exist, go to Step 5. Otherwise, if a solution $x^{(k)}$ exists, check $f(x^{(k)})$. If $f(x^{(k)}) = 0$ (or $\left\|f(x^{(k)}) \leq \epsilon_1\right\|$) then go to Step 4, otherwise go to Step 3.

Step 3: Update Lagrange multipliers and penalty parameter as:

$$u^{(k+1)} = u^{(k)} - \alpha s^{(k)} f(x^{(k)}) \quad (6.6)$$

$$c^{(k+1)} = c^{(k)} + (1+\alpha) s^{(k)} \left\|f(x^{(k)})\right\| \quad (6.7)$$

where $s^{(k)}$ is a positive step length parameter defined as,

$$0 < s^{(k)} = \frac{\delta\alpha(H_n - L(x^{(k)}, u^{(k)}, c^{(k)}))}{[\alpha^2 + (1+\alpha^2)] \|f(x^{(k)})\|^2} \quad (6.8)$$

or

$$0 < s^{(k)} = \frac{\delta \left[\alpha(H_n - L(x^{(k)}, u^{(k)}, c^{(k)})) + (\overline{c} - c^{(k)}) \left\|f(^{(k)})\right\|\right]}{[\alpha^2 + (1+\alpha^2)] \|f(^{(k)})\|^2} \quad (6.9)$$

with $\alpha > 0$ and $0 < \delta < 2$. Step length parameter $s^{(k)}$ should also satisfy the following property:

$$s^{(k)} \left\| f(x^{(k)} \right\| + c^{(k)} - \left\| u^{(k)} \right\| > \ell(k) \tag{6.10}$$

where $\ell(k) \to +\infty$ as $k \to +\infty$. Set $k = k + 1$ and go to Step 2.

Step 4: Let $x^{(k)}$ be a solution of problem (6.5) with $f(x^{(k)}) = 0$, then $L(x^{(k)}, u^{(k)}, c^{(k)}) = f_0(x^{(k)})$. Set $q = q + 1$ and check p. If $p = 0$ then set $\Delta_{n+1} = \Delta_n$, otherwise set $\Delta_{n+1} = \frac{1}{2}\Delta_n$. If $\Delta_{n+1} < \epsilon_2$ then stop, $f_0(x_n^{(k)})$ is an approximate optimal value, $x_n^{(k)}$ is an approximate primal solution and $(u_n^{(k)}, c_n^{(k)})$ is an approximate dual solution; otherwise set $H_{n+1} = \min\left\{ f_0(x_n^{(k)}), H_n - \Delta_{n+1} \right\}$, $n = n + 1$ and go to Step 1.

Step 5: Set $p = p + 1$. If $q = 0$, set $\Delta_{n+1} = \Delta_n$. Otherwise, set $\Delta_{n+1} = \frac{1}{2}\Delta_n$. Set $H_{n+1} = H_n + \Delta_{n+1}$, $n = n + 1$ and go to Step 1.

Since the method is very sensitive to the initial values, in some cases the solution time may be very large or even no solution may be obtained [42].

7. BENDERS' DECOMPOSITION

Benders' decomposition [7] is a method which can be used to solve stochastic or mixed-integer programming problems. Linear mixed-integer problems are complicated by the integer variables since once these variables are fixed the resulting problem is a linear programming problem. Therefore, the idea behind Benders' decomposition is dividing the primal problem into two problems called *Benders' subproblem* and *Benders' master problem* via separating the continuous variables from integer variables.

Consider the following linear mixed-integer programming problem:

$$\begin{aligned} MIP: \quad z &= \min \; c^T x + f^T y \\ \text{s.t.} \quad & Ax + By \geq b \\ & y \in Y \subseteq Z_+^n \\ & x \in R_+^p \end{aligned} \tag{7.1}$$

where $b \in R^m$.

Suppose that the integer variables y are fixed at $\overline{y}$, then the resulting problem is a linear programming problem

$$\begin{aligned} LP: \quad z &= \min \; c^T x \\ \text{s.t.} \quad & Ax \geq b - B\overline{y} \\ & x \in R_+^p \end{aligned} \tag{7.2}$$

and its dual is

$$\begin{aligned} DP: \quad z &= \max \; (b - B\overline{y})^T u \\ \text{s.t.} \quad & A^T u \leq c \\ & u \in R_+^m. \end{aligned} \tag{7.3}$$

Notice that, we can write the MIP as,

$$MIP: \quad z = \min_{y \in Y} \; [f^T y + \max_{u \geq 0} \quad (b - By)^T u] \qquad (7.4)$$
$$\text{s.t.} \quad A^T u \leq c.$$

Let $\left\{u^s \in R_+^m : s \in S\right\}$ be the set of extreme points of $Q = \left\{u \in R_+^m : A^T u \leq c\right\}$ and let $\left\{v^o \in R_+^m : o \in O\right\}$ be the set of extreme rays of $\left\{u \in R_+^m : A^T u \leq 0\right\}$. If $Q \neq \emptyset$, then $\left\{v^o \in R_+^m : o \in O\right\}$ is also the set of extreme rays of Q. Then, the MIP (7.4) can be written as:

$$MIP': \quad z = \min_{y} \; \eta \qquad (7.5)$$
$$\text{s.t.} \quad \eta \geq f^T y + (b - By)^T u_s \quad \text{s} = 1,...,\text{S} \qquad (7.6)$$
$$(b - By)^T v_o \leq 0 \quad \text{o} = 1,...,\text{O} \qquad (7.7)$$
$$y \in Y \subseteq Z_+^n \qquad (7.8)$$
$$\eta \in R. \qquad (7.9)$$

The problem MIP′ is called Benders' master problem. Then the Benders' subproblem can be stated as:

$$z = \max \; f^T \overline{y} + (b - B\overline{y})^T u \qquad (7.10)$$
$$\text{s.t.} \quad A^T u \leq c$$
$$u \in R_+^m.$$

Benders' decomposition algorithm (Table 7.1) iterates between the master problem and the subproblem. At each iteration one of the cuts (7.6) and (7.7) is added to the master problem according to the solution of the subproblem. If the subproblem is bounded, meaning that the solution of the master problem satisfies the integer constraints of the original problem, then extreme point cut (7.6) is added. Otherwise, if it is unbounded, extreme ray cut (7.7) is added to the master problem. The solution of the master problem provides a lower bound for the original problem while the solution

Table 7.1. Benders' Decomposition Algorithm

Step 0: (Initialization) Initialize y_0 to a feasible solution of MIP. Then set lower bound $LB = -\infty$, upper bound $UB = +\infty$, $\overline{y} = y_0$, $k = 1$, and ϵ to a small nonnegative number.

Step 1: Solve the Benders' subproblem (7.10). If the problem is bounded, get extreme point $u^{(k)}$. And add cut $\eta \geq f^T y + (b - By)^T u^{(k)}$ to the master problem. Set $UB = \min\left\{UB, f^T\overline{y} + (b - B\overline{y})^T u^{(k)}\right\}$. If the subproblem is unbounded, get unbounded ray $v^{(k)}$, and add cut $(b - By)^T v^{(k)} \leq 0$ to the master problem. Go to Step 2.

Step 2: Solve the master problem. If the master problem is infeasible, then the original problem is infeasible, STOP. Otherwise, let $\overline{\eta}$ and $y^{(k)}$ be the optimal objective value and optimal solution of the master problem respectively. Set $LB = \overline{\eta}$ and $\overline{y} = y^{(k)}$. If $(UB - LB) <= \epsilon$, STOP. Otherwise, set $k = k + 1$ and go to Step 1.

of the subproblem gives an upper bound. As the algorithm iterates, the number of cuts added to the master problem increases providing an improvement in the lower bound.

The subproblem is a linear programming problem which can be solved in a short time. However, the master problem is a mixed-integer programming problem and we have to solve it at each iteration. The efficiency of the Benders' decomposition algorithm mainly depends on the solution time of the master problem. Therefore, it is important to solve the master problem efficiently. There are different methods proposed in the literature in order to accelerate the Benders' algorithm.

Geoffrion and Graves [20] propose the so-called ϵ-method in 1974. In this method, the master problem is not solved to optimality rather the η value is fixed to $UB - \epsilon$ at each iteration so a feasible solution which produces a value below $UB - \epsilon$ is found. Since the master problem is not solved to optimality, we could not further obtain a lower bound. In this case the algorithm stops whenever the master problem has no feasible solution with value below $UB - \epsilon$.

The LP problem (7.2) obtained by fixing the integer variables y is generally highly degenerate so the Benders' subproblem formed by taking the dual of problem (7.2) has many optimal solutions. Since the solution of the subproblem provides a cut for the master problem, it is crucial to find the solution that gives the strongest cut. Magnanti and Wong [32] present the concept of dominance and pareto-optimality and introduce a linear model for the construction of pareto-optimal cuts. A cut is called pareto-optimal if no cut dominates it. Let $w(\widehat{y})$ be the optimal value of the subproblem (7.10) and $\mathrm{U}(\widehat{y})$ is the set of alternate optimal solutions. Then, in order to find a pareto-optimal point among the points in set $\mathrm{U}(\widehat{y})$, Magnanti and Wong provide the following linear program:

$$
\begin{aligned}
z = \max_{u} \;\; & f^T y^0 + (b - By^0)^T u \qquad (7.11)\\
\text{s.t.} \;\; & f^T \widehat{y} + (b - B\widehat{y})^T u = w(\widehat{y}) \\
& A^T u \leq c \\
& u \in R^m_+
\end{aligned}
$$

where y^0 is a core point. A point $y \in Y$ is a core point if $y \in ri(conv(Y))$, where $ri(Y)$ and $conv(Y)$ are respectively the relative interior and the convex hull of the set Y.

When we fix the integer variables of the problem MIP to a solution given by the master problem, we cannot always obtain a feasible solution. Adding inequalities which are valid for the MIP but not valid for the master problem eliminates some of infeasible solutions. Therefore, the chance of obtaining solutions which are feasible for MIP increases and the number of iterations required for the convergence of the algorithm may decrease.

8. SOLUTION METHODS

The literature on CFLP reveals that Lagrangean heuristics provide good bounds for these problems in a reasonable computational time. Therefore, we decide on using Lagrangean relaxation in order to solve single-source and multi-source problems.

Beasley [6] states that two key issues determine the quality of the solution obtained from Lagrangean relaxation: deciding which constraints to dualize; and deciding how to find numerical values for Lagrange multipliers. Cornuéjols, Sridharan, and Thizy [12] denote that dualizing demand constraints generally provide good bounds. We have also stated before that Lagrangean relaxation with subgradient optimization gives good results. Taking all these facts into consideration, we decide to work on a Lagrangean problem formed by dualizing demand constraints and use *subgradient optimization* to update Lagrange multipliers. Although classical subgradient optimization is widely used in the literature to solve the Lagrangean dual problem, there are some extensions of this method which are proposed in order to overcome some limitations of it. Two of these methods are deflected subgradient [4], [11], [41] and volume algorithm [3]. In addition to the classical subgradient method, we also consider these two methods and compare their performances.

We then focus on exact solution methods and develop a Benders' decomposition method for the multi-source problem only. We do not try to find exact solutions for the single-source model since Lagrangean heuristic gives good results and furthermore CPLEX solves these problems to optimality in reasonable time.

8.1. Solution Procedure for Single-Source Problem

8.1.1. Computation of a Lower Bound

As mentioned above, we dualize the demand constraints (3.2) and obtain the following Lagrangean subproblem:

$$LSP1: \quad z(u) = \min \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L}(c_{ikjl} - u_{jl})x_{ikjl} + \sum_{i=1}^{I}\sum_{k=1}^{K} f_{ik}v_{ik} \tag{8.1}$$

$$+ \sum_{j=1}^{J}\sum_{l=1}^{L} u_{jl} \tag{8.2}$$

s.t.

$$\sum_{j=1}^{J}\sum_{l=1}^{L} d_{jl}x_{ikjl} \leq s_{ik}v_{ik} \quad i = 1,\ldots,I; k = 1,\ldots,K \tag{8.3}$$

$$\sum_{k=1}^{K} v_{ik} \leq 1 \quad i = 1,\ldots,I \tag{8.4}$$

$$v_{ik} = 0,1 \quad i = 1,\ldots,I; k = 1,\ldots,K \tag{8.5}$$

$$x_{ikjl} = 0,1 \quad i = 1,\ldots,I; k = 1,\ldots,K; j = 1,\ldots,J; \quad l = 1,\ldots,L \tag{8.6}$$

where u is the Lagrange multiplier vector.

It can be seen that the above problem is decomposable over facility locations. When we decompose the problem LSP1 over i, we obtain the following subproblem:

$$SP1: \quad z_i(u) = \min \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L}(c_{ikjl} - u_{jl})x_{ikjl} + \sum_{k=1}^{K} f_{ik}v_{ik} \tag{8.7}$$

s.t.

$$\sum_{j=1}^{J}\sum_{l=1}^{L} d_{jl}x_{ikjl} \leq s_{ik}v_{ik} \quad k = 1,\ldots,K \tag{8.8}$$

$$\sum_{k=1}^{K} v_{ik} \leq 1 \tag{8.9}$$

$$v_{ik} = 0,1 \quad k = 1,\ldots,K \tag{8.10}$$

$$x_{ikjl} = 0,1 \quad k = 1,\ldots,K; j = 1,\ldots,J; l = 1,\ldots,L \tag{8.11}$$

Assume that the value of variable v_{ik} is known for each k, meaning that if a facility is opened at location i, we know the class of the facility that is opened. In this case, we can further decompose the problem over the index k. If $v_{ik} = 0$ for all k, no

facility is opened at location i, then no customer can be assigned and $x_{ikjl} = 0$ for all j and l. Therefore, $z_i(u) = 0$.

However, if $v_{ik} = 1$, namely a facility of class k is located at location i, then we end up with the following optimization problem:

$$SP2: \quad z_{ik}(u) = \min \sum_{j=1}^{J} \sum_{l=1}^{L} (c_{ikjl} - u_{jl}) x_{ikjl} + f_{ik} \tag{8.12}$$

s.t.

$$\sum_{j=1}^{J} \sum_{l=1}^{L} d_{jl} x_{ikjl} \leq s_{ik} \tag{8.13}$$

$$x_{ikjl} = 0, 1 \qquad j = 1, \ldots, J; l = 1, \ldots, L \tag{8.14}$$

Since f_{ik} is constant, we can ignore this term and the problem reduces to,

$$SP3: \quad \overline{z}_{ik}(u) = \min \sum_{j=1}^{J} \sum_{l=1}^{L} (c_{ikjl} - u_{jl}) x_{ikjl}$$

s.t. (8.13) – (8.14)

The optimal objective value of LSP1 can be obtained by solving SP3 to optimality, which is easier than solving LSP1. The following equations explain how to obtain the objective value of the relaxed problem LSP1:

$$z_{ik}(u) = \min \{0, \overline{z}_{ik}(u) + f_{ik}\} \tag{8.15}$$

$$z_i(u) = \min_k \{z_{ik}(u)\} \tag{8.16}$$

$$z(u) = \sum_{i=1}^{I} z_i(u) + \sum_{j=1}^{J} \sum_{l=1}^{L} u_{jl} \tag{8.17}$$

SP3 is a 0-1 knapsack problem (KP) which is smaller than the Lagrangean subproblem LSP1. However, 0-1 knapsack problems (KP) are known to be NP-complete

[27]. Therefore, we use a heuristic method in order to find a good feasible solution. We propose three different methods: A greedy heuristic which is described in Table 8.2, a Lagrangean heuristic obtained via relaxing constraint (8.13), and feasible modified subgradient algorithm (F-MSG) [18], [42].

The F-SMG algorithm is originally developed for nonconvex continuous minimization problems with equality constraints. Saraç and Sipahioğlu [42] modify it for solving the 0-1 knapsack problem. They use the theorem by Li [31] to make the binary variables continuous. In addition, a slack variable is added to transform the inequality constraint into an equality one. As a result, we modify our knapsack problem as:

$$\overline{z}_{ik}(u) = \min \sum_{j=1}^{J}\sum_{l=1}^{L}(c_{ikjl} - u_{jl})x_{ikjl} \tag{8.18}$$

s.t.

$$\sum_{j=1}^{J}\sum_{l=1}^{L} d_{jl}x_{ikjl} + w = s_{ik} \tag{8.19}$$

$$\sum_{j=1}^{J}\sum_{l=1}^{L}(x_{ikjl} - x_{ikjl}^2) = 0 \tag{8.20}$$

$$x_{ikjl} \in [0,1] \qquad\qquad j = 1,\ldots,J; l = 1,\ldots,L \tag{8.21}$$

$$w \geq 0 \tag{8.22}$$

where w is the slack variable. Notice that the above problem is equivalent to our knapsack problem SP3. Then, the corresponding augmented Lagrangean problem can be written as:

$$\begin{aligned}\overline{z}_{LP}(u) = \min & \sum_{j=1}^{J}\sum_{l=1}^{L}(c_{ikjl} - u_{jl})x_{ikjl} \\ & - \lambda_1(\sum_{j=1}^{J}\sum_{l=1}^{L} d_{jl}x_{ikjl} + w - s_{ik}) - \lambda_2(\sum_{j=1}^{J}\sum_{l=1}^{L}(x_{ikjl} - x_{ikjl}^2)) \\ & + \mu\left[(\sum_{j=1}^{J}\sum_{l=1}^{L} d_{jl}x_{ikjl} + w - s_{ik})^2 + (\sum_{j=1}^{J}\sum_{l=1}^{L}(x_{ikjl} - x_{ikjl}^2))^2\right]\end{aligned} \tag{8.23}$$

s.t.

$x_{ikjl} \in [0,1] \qquad j = 1,\ldots,J; l = 1,\ldots,L$

$w \geq 0$

Here, λ_1 and λ_2 denote the Lagrange multipliers used to relax constraints (8.19) and (8.20), μ is the penalty parameter.

Thirteen test problems are generated to evaluate the performances of the proposed methods. Since $c_{ikjl} - u_{jl}$ can take negative values when $u_{jl} > c_{ikjl}$, the coefficients of x_{ikjl} are allowed to have negative values when generating the test problems. The results of the methods, average percentage deviations of solutions from the optimal values and average execution times are provided in Table 8.1. In the table, the lower bounds given by the Lagrangean heuristic are not reported, only the upper bounds are presented. We use N/A to indicate that a solution is not available.

Table 8.1: Performances of the methods proposed for subproblem

Problem No	**J**	**L**	**opt.**	**Greedy**	**Lagrangean**	**F-MSG**
1	10	3	-666.06	-666.06	-666.06	-673.99
2	15	3	-1600.54	-1600.54	-1600.54	-1604.86
3	25	3	-2810.77	-2810.69	-2758.31	-2793.64
4	25	5	-4569.78	-4565.36	-4535.62	-4492.47
5	50	3	-9666.78	-9646.44	-9602.70	-8810.63
6	50	5	-3076.07	-3070.63	-2948.27	N/A
7	100	3	-13617.70	-13617.70	-13593.71	-13606.83
8	150	3	-26006.15	-26003.31	-25952.56	N/A
9	100	5	-42695.41	-42693.33	-42669.68	-42647.12
10	200	3	-22843.74	-22829.95	-22816.32	N/A
11	150	5	-11458.20	-11433.36	-11322.41	N/A
12	200	5	-22488.56	-22476.76	-22476.77	-22454.80
13	50	50	-112745	-112742	-112726	N/A
average CPU time				0.00	0.62	39.50
average deviation (%)				0.06	0.71	1.62

When we apply the F-MSG algorithm to Lagrangean problem (8.23), we observe that the algorithm is very sensitive to the initial values and does not give good solutions in reasonable amount of time for some test problems. Furthermore when the value of H is initialized randomly, no solution is obtained. Therefore, for each problem the initial value of H is set to a value close to optimal objective value of SP3. We can see that the greedy heuristic and the Lagrangean heuristic give good near optimal solutions in reasonable time. We prefer to use the greedy heuristic in particular because it is not only more accurate but also simple to use and efficient.

Table 8.2. A Greedy Heuristic for SP3

---

1. Calculate $a_{ikjl} = c_{ikjl} - u_{jl}$ for $j = 1, ..., J, l = 1, ..., L$
2. Set $r_{jl} = \frac{a_{ikjl}}{d_{jl}}$ for $j = 1, ..., J, l = 1, ..., L$
3. Sort r_{jl} in increasing order; starting from the smallest continue as,

   if $r_{jl} < 0$ and $s_{ik} \geq d_{jl}$

   set $x_{ikjl} = 1$

   set $s_{ik} = s_{ik} - d_{jl}$

   else

   set $x_{ikjl} = 0$.
4. Repeat until remaining capacity is not adequate to satisfy the demand of any unassigned customer or until $r_{jl} \geq 0$.

The steps of the greedy heuristic are expressed in Table 8.2. The unit cost r_{jl} is calculated for each customer and these costs are sorted in increasing order. Since it is aimed to minimize the objective function, only the a_{ikjl}'s which are smaller than zero contribute to the objective. Thus x_{ikjl}'s with nonnegative r_{jl} are set to zero. Then, starting from the smallest negative r_{jl}, corresponding x_{ikjl} is set to 1 until the remaining capacity is not adequate to satisfy the demand of any unassigned customer with negative r_{jl} value.

Notice that, the greedy heuristic provides a feasible solution for SP3. A feasible solution gives an upper bound for the optimal objective value. Consequently, obtaining

a feasible solution to the relaxed problem LSP1 may result in overestimating the lower bound of the original problem. However, experimental results show that this bound is quite accurate.

8.1.2. A Lagrangean Heuristic

In this section, our Lagrangean heuristic with classical subgradient optimization is explained. The steps of the heuristic are summarized below, using the following notation:

Z_{LB} : value of the maximum lower bound found
Z_{LB}^{C} : value of the current lower bound
Z_{UB} : value of the minimum upper bound found
Z_{UB}^{C} : value of the current upper bound
N : the number of subgradient iterations since the maximum lower bound last increased
f : the step length parameter for the subgradient optimization procedure

Step 1(Initialization): Set $Z_{LB} = -\infty$, $Z_{UB} = +\infty$, $N = 0$, $\mu = 2$; initialize Lagrange multipliers so that u_{jl} is equal to the difference between the minimum c_{ikjl} over all facilities that satisfy the demand of the customer of class l at location j and the second best serving cost [40], [24].

Step 2 (Initial Upper Bound): In our problem it may be possible that total capacity of some facility combinations does not satisfy the total demand. As a result the upper bound heuristic described in Table 8.3 does not guarantee a feasible solution at each iteration. In this case Z_{UB} is not updated. If this situation occurs at the first iteration, $Z_{UB} = +\infty$ is used to update the Lagrange multipliers which slows down the convergence. Therefore, we find an initial feasible solution for the general problem. The initial upper bound is provided through the following procedure:

1. The demands of the customers and the capacities of the facilities are sorted in decreasing order.

2. Starting from the facility with maximum capacity, the sorted customers are assigned to the facility until it cannot satisfy the demand of any customer.
3. If total demand is not satisfied, repeat 2 with the next facility.

The above heuristic does not take costs into consideration, meaning that it does not try to find a good feasible solution.

Step 3: Solve the Lagrangean subproblem with the current set of multipliers. If $Z_{LB}^{C} > Z_{LB}$ then set N=0 and $Z_{LB} = Z_{LB}^{C}$, else set $N = N+1$. If the solution obtained is not feasible for the original problem, go to Step 4. Otherwise go to Step 6.

Step 4: Apply the upper bound heuristic described in Table 8.3 in order to find a feasible solution to the original problem from the solution of relaxed problem LSP1. If $Z_{UB}^{C} < Z_{UB}$, then set $Z_{UB} = Z_{UB}^{C}$.

Since the demand constraints (3.2) are relaxed, the solution of the relaxed problem may not satisfy these constraints. Therefore it may be the case that some customers are assigned to more than one facility or some customers are not assigned to any facility. Table 8.3 tries to remedy these situations taking into consideration the cost issues. First of all, the customers which are assigned to more than one facility are considered. Each of these customers is assigned to the facility which possess the minimum serving cost among the assigned facilities and the remaining capacities are adjusted. Therefore, the real remaining capacity of each facility is obtained. Secondly, the customers which are not assigned to any facility are considered. If there are open facilities that can satisfy demand of a customer, the customer is assigned to one with minimum serving cost. Otherwise, a new facility is opened.

Step 5: If $N = 20$ then set $\mu = \frac{1}{2}\mu$ and $N = 0$.

Step 6: If the solution obtained is feasible to the original problem or $\mu \leq 0.005$ then go to Step 8.

Table 8.3. An Upper Bound Heuristic for P1

1. If a customer is assigned to more than one facility, assign the customer to the facility which has the minimum serving cost (c_{ikjl}) among the assigned facilities and adjust the remaining capacities. Repeat this step until no customer is assigned to more than one facility.
2. If there exist customers that are not assigned to any facility:
   - sort these customers according to their demands,
   - starting from the customer with maximum demand, assign each customer to an open facility with minimum serving cost for that customer,
   - if remaining capacity of any open facility does not satisfy demand of the customer, open a facility with minimum unit fixed cost $(\frac{f_{ik}}{s_{ik}})$,
   - repeat these steps until total demand is satisfied or until I facilities are opened and the remaining capacities of these facilities do not satisfy demand of any unassigned customer.
3. If total demand is satisfied, calculate Z_{UB}^{C}.

Step 7: Define the step size λ using the formula:

$$\lambda = \frac{\mu(Z_{UB} - Z_{LB})}{\sum_{j=1}^{J}\sum_{l=1}^{L}(G_{jl})^2} \tag{8.24}$$

where

$$G_{jl} = 1 - \sum_{i=1}^{I}\sum_{k=1}^{K} x_{ikjl} \tag{8.25}$$

and update the Lagrange multipliers using the formula:

$$u_{jl} = u_{jl} + \lambda G_{jl} \tag{8.26}$$

and go to Step 3.

Step 8(Stop): Record Z_{LB} and Z_{UB}.

8.2. Solution Procedure for Multi-Source Problem

We use two different methods to solve the multi-source problem, the *Lagrangean heuristic* and the *Benders' decomposition*. While the Lagrangean heuristic provides a feasible solution, Benders' decomposition finds the optimal solution for our problem.

8.2.1. Using Lagrangean Relaxation and Subgradient Optimization

As in the single-source problem, Lagrangean heuristic is used to find bounds for the multi-source problem. The proposed method is very similar to the one used for the single-source version.

8.2.1.1. Computation of a Lower Bound. When demand constraints (3.8) are relaxed, the following Lagrangean subproblem is obtained:

$$LSP2: \quad z(u) = \min \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L}(c_{ikjl} - u_{jl})x_{ikjl} + \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L} g_{ikjl}y_{ikjl} + \sum_{i=1}^{I}\sum_{k=1}^{K} f_{ik}v_{ik} + \sum_{j=1}^{J}\sum_{l=1}^{L} u_{jl} \tag{8.27}$$

s.t.

$$\sum_{j=1}^{J}\sum_{l=1}^{L} d_{jl}x_{ikjl} \leq s_{ik} \qquad i = 1,\ldots,I; k = 1,\ldots,K \tag{8.28}$$

$$\sum_{k=1}^{K} v_{ik} \leq 1 \qquad i = 1,\ldots,I \tag{8.29}$$

$$y_{ikjl} \leq v_{ik} \qquad i = 1,\ldots,I; k = 1,\ldots,K; \; j = 1,\ldots,J; l = 1,\ldots,L \tag{8.30}$$

$$x_{ikjl} \leq y_{ikjl} \qquad i = 1,\ldots,I; k = 1,\ldots,K; \; j = 1,\ldots,J; l = 1,\ldots,L \tag{8.31}$$

$$x_{ikjl} \geq 0 \qquad i = 1,\ldots,I; k = 1,\ldots,K; \; j = 1,\ldots,J; l = 1,\ldots,L \tag{8.32}$$

$$y_{ikjl} = 0, 1 \qquad i = 1, \ldots, I; k = 1, \ldots, K; \quad j = 1, \ldots, J; l = 1, \ldots, L \tag{8.33}$$

$$v_{ik} = 0, 1 \qquad i = 1, \ldots, I; k = 1, \ldots, K \tag{8.34}$$

Observe that the above problem can be decomposed with respect to facility locations. Hence for facility location i we have,

$$SP4: \ z_i(u) = \min \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L}(c_{ikjl} - u_{jl})x_{ikjl} + \sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L} g_{ikjl}y_{ikjl} + \sum_{k=1}^{K} f_{ik}v_{ik} \tag{8.35}$$

s.t.

$$\sum_{j=1}^{J}\sum_{l=1}^{L} d_{jl}x_{ikjl} \leq s_{ik} \qquad k = 1, \ldots, K \tag{8.36}$$

$$\sum_{k=1}^{K} v_{ik} \leq 1 \tag{8.37}$$

$$y_{ikjl} \leq v_{ik} \qquad k = 1, \ldots, K; j = 1, \ldots, J; l = 1, \ldots, L \tag{8.38}$$

$$x_{ikjl} \leq y_{ikjl} \qquad k = 1, \ldots, K; j = 1, \ldots, J; l = 1, \ldots, L \tag{8.39}$$

$$x_{ikjl} \geq 0 \qquad k = 1, \ldots, K; j = 1, \ldots, J; l = 1, \ldots, L \tag{8.40}$$

$$y_{ikjl} = 0, 1 \qquad k = 1, \ldots, K; j = 1, \ldots, J; l = 1, \ldots, L \tag{8.41}$$

$$v_{ik} = 0, 1 \qquad k = 1, \ldots, K \tag{8.42}$$

as the i^{th} subproblem.

When v_{ik} is known, we can further decompose the problem over the facility classes. If $v_{ik} = 0$ for all k, meaning that no facility is opened at location i, then $x_{ikjl} = 0$ and $y_{ikjl} = 0$ for all j and l. Therefore, no serving and assignment costs are occurred and $z_i(u) = 0$.

However, if $v_{ik} = 1$, namely a facility of class k is located at location i, then we

end up with the following optimization problem:

$$SP5: \quad z_{ik}(u) = \min \sum_{j=1}^{J}\sum_{l=1}^{L}(c_{ikjl} - u_{jl})x_{ikjl} + \sum_{j=1}^{J}\sum_{l=1}^{L} g_{ikjl}y_{ikjl} + f_{ik} \tag{8.43}$$

s.t.

$$\sum_{j=1}^{J}\sum_{l=1}^{L} d_{jl}x_{ikjl} \leq s_{ik} \tag{8.44}$$

$$x_{ikjl} \leq y_{ikjl} \qquad j = 1, \ldots, J; l = 1, \ldots, L \tag{8.45}$$

$$x_{ikjl} \geq 0 \qquad j = 1, \ldots, J; l = 1, \ldots, L \tag{8.46}$$

$$y_{ikjl} = 0, 1 \qquad j = 1, \ldots, J; l = 1, \ldots, L \tag{8.47}$$

Since f_{ik} is constant, we can ignore this term and the problem reduces to,

$$SP6: \quad \overline{z}_{ik}(u) = \min \sum_{j=1}^{J}\sum_{l=1}^{L}(c_{ikjl} - u_{jl})x_{ikjl} + \sum_{j=1}^{J}\sum_{l=1}^{L} g_{ikjl}y_{ikjl} \tag{8.48}$$

s.t. (8.44) – (8.47)

The objective value of LSP2 can be obtained by using the following equations:

$$z_{ik}(u) = \min \{0, \overline{z}_{ik}(u) + f_{ik}\} \tag{8.49}$$

$$z_i(u) = \min_k \{z_{ik}(u)\} \tag{8.50}$$

$$z(u) = \sum_{i=1}^{I} z_i(u) + \sum_{j=1}^{J}\sum_{l=1}^{L} u_{jl} \tag{8.51}$$

SP6 is a Single-Node Fixed-Charge Transportation (SNFCT) Problem [21]. Since this problem is known to be NP-hard [21], we use a heuristic method to find a feasible solution. We propose three heuristics: Lagrangean heuristic, F-MSG algorithm and a heuristic (Table 8.5) which is similar to the greedy heuristic used to solve the Knapsack Problem. The performances of the methods are tested on thirteen test problems. The

results are provided in Table 8.4. F-MSG algorithm is not able to find solutions for all problems. In addition, it is strictly dependent on the initial values and requires more execution time than the other two methods. Among the three methods, our simple heuristic performs the best so it is considered in the solution of SP6.

Table 8.4: Performances of the methods proposed for subproblem

Problem No	**J**	**L**	**opt.**	**Greedy**	**Lagrangean**	**F-MSG**
1	10	3	-498.44	-498.44	-498.44	-506.95
2	15	3	-1025.88	-1016.62	-1016.62	-1029.28
3	25	3	-2122.97	-2103.28	-2103.28	-2066.45
4	25	5	-3375.16	-3366.64	-3366.64	-3379.06
5	50	3	-7120.80	-7110.61	-7110.61	-7115.98
6	50	5	-2344.69	-2337.30	-2337.30	-2296.84
7	100	3	-9278.54	-9275.85	-9275.85	-9278.20
8	150	3	-19218.05	-19204.08	-19204.08	-19156.96
9	100	5	-31594.79	-31594.43	-30696.99	N/A
10	200	3	-17118.47	-17118.28	-17118.29	-17062.27
11	150	5	-9190.91	-9178.38	-9178.38	N/A
12	200	5	-14381.12	-14373.51	-14373.50	-14129.83
13	50	50	-71661.05	-71658.05	-69181.52	N/A
average CPU time				0.08	0.38	30
average deviation (%)				0.22	0.70	0.93

Steps of the greedy heuristic are summarized in Table 8.5. The unit cost r_{jl} is calculated for each customer and they are sorted in increasing order. This unit cost is composed of unit service cost and unit fixed service cost. Since it is aimed to minimize the objective function, only the unit costs which are smaller than zero contribute to the objective. Thus x_{ikjl}'s with nonnegative r_{jl} and the corresponding y_{ikjl}'s are set to zero. Starting from the smallest negative r_{jl}, corresponding x_{ikjl} and y_{ikjl} are set to 1 if the remaining capacity is adequate to satisfy the demand of the customer. Otherwise, x_{ikjl} and y_{ikjl} are set to $\frac{s_{ik}}{d_{jl}}$ and 1 respectively if cost improvement is obtained, namely if $(c_{ikjl} - u_{jl})\frac{s_{ik}}{d_{jl}} + g_{ikjl} < 0$.

Table 8.5. A Greedy Heuristic for SP6

1. Calculate unit carriage cost r_{jl}

   set $r_{jl} = \frac{(c_{ikjl} - u_{jl}) + g_{ikjl}}{d_{jl}}$ for $j = 1, ..., J; l = 1, ..., L$

2. Sort r_{jl} in increasing order; starting from the smallest continue as,

   if $r_{jl} < 0$ and $s_{ik} \geq d_{jl}$

   set $x_{ikjl} = 1$ and $y_{ikjl} = 1$

   set $s_{ik} = s_{ik} - d_{jl}$

   else if $r_{jl} < 0$, $s_{ik} < d_{jl}$, and $(c_{ikjl} - u_{jl})\frac{s_{ik}}{d_{jl}} + g_{ikjl} < 0$

   set $x_{ikjl} = \frac{s_{ik}}{d_{jl}}$ and $y_{ikjl} = 1$

   set $s_{ik} = 0$ and STOP

   else

   set $x_{ikjl} = 0$ and $y_{ikjl} = 0$.

4. Repeat 2 until all capacity is replenished ($s_{ik} = 0$) or until $r_{jl} \geq 0$

8.2.1.2. A Lagrangean Heuristic. We modify the heuristic we propose for the single-source problem in order to obtain an upper bound from the lower bound solution. The upper bound heuristic is defined in Table 8.6, using the following notation:

d^r_{jl} : amount of the unsatisfied demand for customer of class l at location j

s^r_{ik} : amount of the remaining capacity for facility of class k at location i

where r means remaining.

The initial upper bound heuristic is also similar to heuristic used in the single-source problem, except that in this case a customer can be assigned to more than one facility.

8.2.2. A Benders' Decomposition Based Exact Solution Procedure

The main purpose of Benders' decomposition for linear mixed-integer programming problems is to obtain an easy to solve LP subproblem and an MIP master problem with only one continuous variable by separating the continuous variables from the inte-

Table 8.6. An Upper bound Heuristic for P2

1. If a customer is assigned to more than one facility, such that more than its demand is provided, assign the customer to the facilities which have the minimum unit cost $\frac{c_{ikjl}x_{ikjl}+g_{ikjl}}{x_{ikjl}d_{jl}}$ among the assigned facilities until the customer demand is satisfied. Repeat this step until no customer is served with more than its demand.
2. If there exist customers with unsatisfied demand:
   - sort them according to their demands,
   - starting from the customer with maximum demand, assign each customer to open facilities with minimum unit cost
     $\frac{c_{ikjl}d^r_{jl}/d_{jl}+g_{ikjl}}{d^r_{jl}}$ if $s^r_{ik} \geq d^r_{jl}$
     $\frac{c_{ikjl}s^r_{ik}/d_{jl}+g_{ikjl}}{s^r_{ik}}$ if $s^r_{ik} < d^r_{jl}$
     for that customer,
   - if any open facility does not satisfy demand of the customer, open a facility with minimum unit fixed cost $(\frac{f_{ik}}{s_{ik}})$,
   - repeat these steps until total demand is satisfied or until number of I facilities are opened and their capacity does not satisfy total demand.
3. If total demand is satisfied, calculate Z^C_{UB}.

ger ones. The information obtained from one problem is passed repeatedly to another and this may lead to efficient solution procedures.

When the binary variables y and v are fixed to $\overline{y}$ and $\overline{v}$ respectively, the following LP subproblem is obtained:

$$LP: \quad z = \min \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L} c_{ikjl}x_{ikjl} + \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L} g_{ikjl}\overline{y}_{ikjl} + \sum_{i=1}^{I}\sum_{k=1}^{K} f_{ik}\overline{v}_{ik} \tag{8.52}$$

s.t.

$$\sum_{i=1}^{I}\sum_{k=1}^{K} x_{ikjl} = 1 \qquad j = 1,\ldots,J; l = 1,\ldots,L \tag{8.53}$$

$$\sum_{j=1}^{J}\sum_{l=1}^{L} d_{jl}x_{ikjl} \le s_{ik} \qquad i=1,\ldots,I; k=1,\ldots,K \tag{8.54}$$

$$x_{ikjl} \le \overline{y}_{ikjl} \qquad i=1,\ldots,I; k=1,\ldots,K; \quad j=1,\ldots,J; l=1,\ldots,L \tag{8.55}$$

$$x_{ikjl} \ge 0 \qquad i=1,\ldots,I; k=1,\ldots,K; \quad j=1,\ldots,J; l=1,\ldots,L \tag{8.56}$$

Taking the dual of the above problem LP, the Benders' subproblem is obtained as:

$$SP: \ z = \max \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L} g_{ikjl}\overline{y}_{ikjl} + \sum_{i=1}^{I}\sum_{k=1}^{K} f_{ik}\overline{v}_{ik} - \sum_{j=1}^{J}\sum_{l=1}^{L}\pi_{jl} - \sum_{i=1}^{I}\sum_{k=1}^{K} s_{ik}\mu_{ik} - \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L}\overline{y}_{ikjl}w_{ikjl} \tag{8.57}$$

s.t.

$$-\pi_{jl} - d_{jl}\mu_{ik} - w_{ikjl} \le c_{ikjl} \qquad i=1,\ldots,I; k=1,\ldots,K; \quad j=1,\ldots,J; l=1,\ldots,L \tag{8.58}$$

$$w_{ikjl} \ge 0 \qquad i=1,\ldots,I; k=1,\ldots,K; \quad j=1,\ldots,J; l=1,\ldots,L \tag{8.59}$$

$$\mu_{ik} \ge 0 \qquad i=1,\ldots,I; k=1,\ldots,K; \tag{8.60}$$

$$\pi_{jl} = free \qquad j=1,\ldots,J; l=1,\ldots,L \tag{8.61}$$

where w, μ, and π are dual variables.

Then the obtained Benders' master problem MP is in the form:

$$MP: \ z = \min \ \eta \tag{8.62}$$

s.t.

$$\eta \ge \Omega_1^m \qquad m=1,\ldots,M \tag{8.63}$$

$$\Omega_2^n \le 0 \qquad n=1,\ldots,N \tag{8.64}$$

$$\sum_{k=1}^{K} v_{ik} \leq 1 \qquad i = 1, \ldots, I \tag{8.65}$$

$$y_{ikjl} \leq v_{ik} \qquad i = 1, \ldots, I; k = 1, \ldots, K; j = 1, \ldots, J; l = 1, \ldots, L \tag{8.66}$$

$$y_{ikjl} = 0, 1 \qquad i = 1, \ldots, I; k = 1, \ldots, K; j = 1, \ldots, J; l = 1, \ldots, L \tag{8.67}$$

$$v_{ik} = 0, 1 \qquad i = 1, \ldots, I; k = 1, \ldots, K \tag{8.68}$$

where Ω_1^m and Ω_2^n are defined as:

$$\Omega_1^m = \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L} g_{ikjl} y_{ikjl} + \sum_{i=1}^{I}\sum_{k=1}^{K} f_{ik} v_{ik} - \sum_{j=1}^{J}\sum_{l=1}^{L} \pi_{jl}^m - \sum_{i=1}^{I}\sum_{k=1}^{K} s_{ik}\mu_{ik}^m - \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L} y_{ikjl} w_{ikjl}^m, \tag{8.69}$$

$$\Omega_2^n = -\sum_{j=1}^{J}\sum_{l=1}^{L} \pi_{jl}^n - \sum_{i=1}^{I}\sum_{k=1}^{K} s_{ik}\mu_{ik}^n - \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L} y_{ikjl} w_{ikjl}^n. \tag{8.70}$$

In the master problem MP, there are two kinds of cuts: extreme point cuts (8.63) and extreme ray cuts (8.64). If the Benders' subproblem SP is bounded, an extreme point cut is added to the master problem. On the other hand, if it is unbounded then the Benders' modified subproblem MSP is solved in order to obtain an extreme ray cut.

$$MSP: \quad z = \max \ 0 \tag{8.71}$$

s.t.

$$-\sum_{j=1}^{J}\sum_{l=1}^{L} \pi_{jl} - \sum_{i=1}^{I}\sum_{k=1}^{K} s_{ik}\mu_{ik} - \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{L} \overline{y}_{ikjl} w_{ikjl} = 1 \tag{8.72}$$

$$-\pi_{jl} - d_{jl}\mu_{ik} - w_{ikjl} \leq 0 \qquad i = 1, \ldots, I; k = 1, \ldots, K; \quad j = 1, \ldots, J; l = 1, \ldots, L \tag{8.73}$$

$$w_{ikjl} \geq 0 \qquad i = 1, \ldots, I; k = 1, \ldots, K; \quad j = 1, \ldots, J; l = 1, \ldots, L \tag{8.74}$$

$$\mu_{ik} \geq 0 \qquad i = 1, \ldots, I; k = 1, \ldots, K; \tag{8.75}$$

$$\pi_{jl} = free \qquad j = 1, \ldots, J; l = 1, \ldots, L \tag{8.76}$$

Notice that a feasible solution for the multi-source problem P2 may not be obtained when variables y and z are fixed to the values provided by the Benders's master problem. In this case, the Benders' subproblem SP is unbounded therefore we solve the modified subproblem MSP. And the added extreme ray cut eliminates this infeasible solution. On the other hand, if we can find some additional constraints for the master problem such that they are valid for the original problem P2 but not valid for the master problem, we can eliminate some of these infeasible solutions without solving the modified subproblem. So, we add two additional types of constraints to the master problem. Observe that master problem may provide a solution such that total capacity of the opened facilities is smaller than the total demand of the customers. Then, we add constraints (8.77) to the master problem. Furthermore, in the original problem constraints (3.12) ensure that a customer can be served from a facility if it is assigned to that facility. However, master problem may give a solution such that the variable y is zero for all i and k for a customer of class l at location j. In this case, this customer cannot be served from any facility. As a result, we add constraints (8.78).

$$\sum_{j=1}^{J}\sum_{l=1}^{L} d_{jl} \leq \sum_{i=1}^{I}\sum_{k=1}^{K} s_{ik}v_{ik} \tag{8.77}$$

$$\sum_{i=1}^{I}\sum_{k=1}^{K} y_{ikjl} \geq 1 \quad \text{j = 1,...,J; l = 1,...,L} \tag{8.78}$$

Obtaining the master problem and subproblems, we use the Table 7.1 in order to obtain an exact solution for our multi-source problem. The initial feasible solution is found using the *initial upper bound heuristic* which is developed for our Lagrangean heuristic. Although our Benders' decomposition algorithm finds an exact solution, it is not more efficient than some of the widely used commercial solvers (e.g. CPLEX). Therefore, we focus on some acceleration techniques which are proposed in the literature.

Since the master problem is a mixed-integer programming problem, overall computation time of the algorithm mainly depends on the solution time of the master problem. So, we decide to try the ϵ-method of Geoffrion and Graves [20], a method which suggests to find a feasible solution for the master problem instead of solving it to

optimality. Although this method decreases the average computation time of an iteration, the number of iterations required to find an exact solution for the original problem increases considerably. As a consequence, we are not able to obtain an improvement in the overall computation time through this method.

Magnanti and Wong [32] introduce the concept of pareto-optimality and present a linear program (7.11) for finding pareto-optimal cuts. At each iteration where the Benders' subproblem provides an extreme point cut, we solve the suggested linear program to find the strongest cut. The pareto-optimal solutions depend to the selection of the core points, different core points can give different pareto-optimal cuts. We change the core points at each iteration taking the convex combination of points which are provided by the master problem and at the same time which can give a feasible solution for the original problem.

To sum up, in our Benders' algorithm we solve the master problem to optimality at each iteration and find a pareto-optimal cut whenever the subproblem is not unbounded.

9. EXPERIMENTAL RESULTS

In this chapter, we present the results of our computational experiments performed by using the methods proposed in the previous sections. In the first and second sections we give the implementation details and present the computational results for the single-source and the multi-source problems respectively. As we know, both the single-source and multi-source CFLP with customer and facility differentiation have not been addressed in the literature before. Consequently, we are not able to find benchmark problems and create some test problems randomly in order to evaluate the performances of the methods we propose. We conduct all of our experiments on a PC with 2.40 GHz Intel(R) Core(TM)2 processor and 3.92 GBytes of RAM, under the Windows XP operating system. We encode new heuristics in Microsoft Visual Studio.Net 2005 and the Benders' decomposition algorithm in GAMS. The Benders' algorithm is adopted from the GAMS code provided by Kalvelagen [26] for a fixed charge transportation problem.

9.1. Single-Source Problem

9.1.1. Test Bed

The test problems are generated randomly in the following way. We assume that the customer serving cost depends on the distance between the customers and facilities. On the other hand, it is also expected that there should be differences in the costs of serving customers of different classes from facilities of different classes. Therefore, customer serving cost is calculated as,

$$c_{ikjl} = dist_{ij} * \rho_{kl} \tag{9.1}$$

where $dist_{ij}$ is the distance between facility i and customer j, ρ_{kl} is a random multiplier generated from normal distribution with mean 6 and standard deviation 4. These values are selected arbitrarily. When normal distribution is used, it may be possible

that negative values can exist. Since negative cost values are meaningless, the random number generation is repeated until positive values are obtained.

The fixed cost of opening a facility f_{ik} is obtained in a manner analogous to that process of determination of ρ_{kl}. It is randomly generated from a normal distribution with mean 500 and standard deviation 180. Again the values are selected arbitrarily.

Locations of customers and facilities are generated uniformly from range (0,10). Demands of customers are drawn in the same way but in the range (5,50) and rounded down to the greatest integer. The capacities of facilities are obtained uniformly in the range,

$$\left(\left\lfloor \frac{5JLq}{I} \right\rfloor, \left\lfloor \frac{50JLq}{I} \right\rfloor \right) \tag{9.2}$$

which is similar to the range proposed by Cortinhal and Captivo [15]. Here J is the number of customer locations, L is the number of customer classes, I is the number of potential facility locations, and q is a ratio. Facility capacities are also rounded to the greatest integer.

We generate 41 test problems with $q = 0.9, 1.0, 1.1, 1.2$. For each instance the number of candidate facility locations I, the number of facility classes K, the number of customers J, and the number of customer classes L are given in Table 9.1. We try to solve the test problems to optimality using CPLEX 10.1 solver through the instrumentality of GAMS 22.2 interface. Unfortunately, CPLEX is not able to solve all test problems to optimality in our computational environment because of memory limitations. In such cases, the problems are solved with small relative optimality criterion (relative tolerance) ϑ. We present optimal or near optimal objective values, the number of nodes, relative tolerance ϑ, linear programming (LP) relaxation values, relative percentage deviation of LP relaxation values from optimal objective values, and total execution time (CPU time) of solved test instances in Table 9.1. We use the formula $\left(100\frac{opt\ -\ LP\ relax}{opt}\%\right)$ to denote relative percentage deviations of LP relaxation values.

Table 9.1: Optimal and linear programming (LP) relaxation results of the single-source test problems

Problem No	I	K	J	L	opt.	nodes	CPU time	ϑ (%)	LP relax.	CPU time	dev. (%)
1	2	3	10	3	1628.32	10	0.17	0.00	1193.71	0.00	26.69
2	4	3	10	3	1471.80	20	0.17	0.00	1156.96	0.03	21.39
3	4	3	15	3	1634.33	0	0.03	0.00	1573.53	0.02	3.72
4	5	3	15	3	2216.81	100	0.23	0.00	2098.62	0.02	5.33
5	8	3	25	3	2601.80	652	1.06	0.00	2528.67	0.02	2.81
6	9	3	25	5	2586.41	376	1.69	0.00	2399.05	0.03	7.24
7	10	3	50	3	4409.81	6135	9.89	0.00	4183.70	0.05	5.13
8	10	4	50	5	5004.22	102203	435.70	2.80	4657.34	0.11	6.93
9	15	3	100	3	5010.65	16794	76.33	0.00	4674.28	0.13	6.71
10	25	4	100	5	6708.17	9800	274.93	0.06	6387.75	0.56	4.78
11	15	3	150	3	9152.76	20825	218.22	0.10	8915.96	0.17	2.59
12	25	4	150	5	9476.45	5200	273.10	0.04	9274.21	0.86	2.13
13	15	3	200	3	7753.82	55200	449.02	0.05	7327.80	0.27	5.49
14	25	4	200	5	11808.78	9376	469.81	0.05	11412.85	1.14	3.35
15	15	3	250	3	9543.55	24101	289.96	0.00	9064.91	0.31	5.02
16	25	4	250	5	12979.69	18669	1819.78	0.09	12089.60	1.42	6.86
17	15	3	300	3	9175.17	112	32.83	0.00	8343.88	0.31	9.06
18	25	4	300	5	12032.67	5080	745.61	0.04	11086.43	2.22	7.86
19	15	3	400	3	9698.17	69714	3365.71	0.00	8944.99	0.59	7.77
20	25	4	400	5	16885.40	13800	2326.83	0.04	15775.98	3.88	6.57
21	15	3	500	3	13537.65	4400	149.71	0.01	12606.51	0.80	6.88
22	25	4	500	5	27491.55	3255	1537.75	0.03	25339.85	2.84	7.83
23	15	3	600	3	16712.14	5000	272.10	0.01	15476.59	0.97	7.39
24	25	4	600	5	15436.26	3000	2287.08	0.04	14153.83	9.27	8.31

Continued on Next Page...

Problem No	I	K	J	L	opt.	nodes	CPU time	ϑ (%)	LP relax.	CPU time	dev. (%)
25	15	3	700	3	19788.97	3204	160.05	0.00	17723.74	0.72	10.44
26	25	4	700	5	23326.17	1200	1731.41	0.01	20755.69	3.91	11.02
27	20	3	500	3	14531.97	24400	2862.19	0.04	13415.09	1.09	7.69
28	30	4	500	5	19806.38	7829	7416.47	0.07	18353.18	4.91	7.34
29	20	3	550	3	13383.94	1411	180.68	0.01	12695.95	2.20	5.14
30	30	4	550	5	25843.96	12325	15025.43	0.04	22356.19	3.61	13.50
31	20	3	600	3	15277.37	2000	526.89	0.02	13924.15	1.53	8.86
32	30	4	650	5	32099.20	8220	9750.00	0.20	29897.32	4.84	6.86
33	20	3	700	3	20079.48	3800	363.34	0.02	19089.51	1.30	4.93
34	30	4	750	5	24415.84	600	2963.76	0.02	22153.07	6.50	9.27
35	20	3	800	3	16312.98	2644	1780.88	0.01	15401.29	4.08	5.59
36	30	4	850	5	21993.04	4162	8123.53	0.02	21058.83	17.92	4.25
37	20	3	900	3	17904.23	1070	932.75	0.02	15422.24	1.45	13.86
38	30	4	950	5	23417.27	6400	24416.18	0.03	20587.50	21.27	12.08
39	20	3	1000	3	28176.56	18359	4651.92	0.07	26747.68	2.28	5.07
40	30	4	1050	5	30540.42	5421	14255.25	0.03	27748.67	9.09	9.14
41	40	4	1400	5	48044.25	4499	68605.67	0.22	42577.95	15.55	11.38
Average							**4360.59**			**3.13**	**7.91**

Since the relative tolerances are very small except problem 8, it can be said that the solutions found are optimal values or very close to them. The test problems are solved in 4360.59 seconds on the average and LP relaxation gives bounds with average deviation of 7.91% from optimal values. The results in Table 9.1 are used as a reference in the rest of the study.

9.1.2. Performances of the Proposed Methods

In this section, we compare experimentally our Lagrangean heuristic results with the results provided by CPLEX both in terms of solution quality and execution time efficiency on a number of test instances. The initial value of step length parameter μ of the subgradient algorithm is initialized to 2 for classical subgradient method and volume algorithm. Since, the step length parameter should be in the interval $(0, 1)$ for the deflected subgradient method, it is initialized to 1 if deflected method is used. When there is no improvement in the lower bound after a sequence of 20 iterations, μ is halved if we use classical and deflected subgradient methods. In the volume algorithm, the step length parameter is updated using a different procedure which is explained clearly in Section 5.2. All three algorithms stop when μ gets below 0.005. Camerini et al. [11] suggest to initialize τ, a coefficient used to calculate deflection parameter, to 1.5 in the deflected subgradient method. Therefore, we use this value in all our experiments. Barahona and Anbil [3] propose two methods to determine the value of coefficient β, which is used to take the convex combination of Lagrangean subproblem solutions. One method is to fix it to a value for a number of iterations and then decrease. The other method is to find the values by solving a non-linear programming problem. The details of this suggestions can be found in Section 5.2. We prefer to use the first method instead of solving the given problem. However, the constraint of the suggested problem is taken into consideration during the initialization and update procedure. The value of β is initialized to 0.1, and after every 100 iterations it is checked whether the lower bound has increased by at least 1 %, if not β is divided by 2. When β becomes less than 10^{-5}, it is kept constant.

Table 9.2: Results for the single-source test problems: Lagrangean heuristic with classical subgradient optimization

		Classical Subgradient Optimization				
Problem No	**opt.**	**LB**	**UB**	**LB dev. (%)**	**UB dev. (%)**	**CPU time**
1	1628.32	1298.77	1650.38	20.24	1.35	0
2	1471.80	1196.43	1676.14	18.71	13.88	0
3	1634.33	1612.17	1634.33	1.36	0.00	1
4	2216.81	2141.74	2549.72	3.39	15.02	0
5	2601.80	2612.95	2610.04	-0.43	0.32	1
6	2586.41	2549.55	2664.02	1.43	3.00	2
7	4409.81	4394.49	4438.31	0.35	0.65	3
8	5004.22	4827.44	5123.52	3.53	2.38	10
9	5010.65	5016.57	5106.05	-0.12	1.90	19
10	6708.17	6643.57	6786.04	0.96	1.16	117
11	9152.76	9109.65	9302.92	0.47	1.64	36
12	9476.45	9450.47	9787.41	0.27	3.28	202
13	7753.82	7724.94	8021.65	0.37	3.45	58
14	11808.78	11774.16	11902.90	0.29	0.80	306
15	9543.55	9500.87	9596.75	0.45	0.56	84
16	12979.69	12943.32	13338.55	0.28	2.76	603
17	9175.17	9156.60	9192.99	0.20	0.19	125
18	12032.67	12008.54	12208.46	0.20	1.46	752
19	9698.17	9661.70	10052.34	0.38	3.65	200
20	16885.40	16854.78	17493.82	0.18	3.60	1540
21	13537.65	13522.66	13560.65	0.11	0.17	328
22	27491.55	27472.44	28023.36	0.07	1.93	2118
23	16712.14	16632.49	16800.46	0.48	0.53	483
24	15436.26	15416.26	15979.20	0.13	3.52	4393
25	19788.97	19789.99	19789.94	-0.01	0.00	722
26	23326.17	23321.52	23348.65	0.02	0.10	5707
27	14531.97	14389.13	14859.84	0.98	2.26	499
28	19806.38	19715.11	20346.30	0.46	2.73	2957
29	13383.94	13310.17	13566.87	0.55	1.37	551
30	25843.96	25755.70	26323.84	0.34	1.86	3349

Continued on Next Page...

		Classical Subgradient Optimization				
Problem No	**opt.**	**LB**	**UB**	**LB dev. (%)**	**UB dev. (%)**	**CPU time**
31	15277.37	15250.32	15384.29	0.18	0.70	680
32	32099.20	32026.12	32429.82	0.23	1.03	6074
33	20079.48	20021.46	20127.82	0.29	0.24	932
34	24415.84	24403.64	24521.88	0.05	0.43	7569
35	16312.98	16141.96	16525.90	1.05	1.31	1744
36	21993.04	21941.99	22083.27	0.23	0.41	9416
37	17904.23	17899.96	18018.52	0.02	0.64	1642
38	23417.27	23389.69	23681.27	0.12	1.13	10367
39	28176.56	28066.48	28286.97	0.39	0.39	1647
40	30540.42	30517.82	30656.21	0.07	0.38	14813
41	48044.25	47900.42	49044.23	0.30	2.08	40742
Average				**1.56**	**2.06**	**2946.15**

During the interpretation of all results we use LB and UB to denote lower and upper bounds respectively, and opt to denote the optimal or near optimal objective values of the test problems. The percentage deviations of lower and upper bounds from the optimal values are calculated as $\left(100\frac{opt\ -\ LB}{opt}\%\right)$ and $\left(100\frac{opt\ -\ UB}{opt}\%\right)$ respectively. Table 9.2 presents the results of Lagrangean heuristic with classical subgradient optimization. Notice that, for some problem instances the obtained lower bound is greater than the optimal value. This is because, in our Lagrangean heuristic we find a feasible solution for our subproblems instead of solving them to optimality or solving their LP relaxations. These problem instances are discarded when average deviations of lower bounds are calculated. On the average, lower bound deviation is around 1.6% which is a clear indication of the lower bound qualities. Except the problems 1 and 2, lower bound deviation is less than 3.53% for all test problems. In addition, the average deviations of upper bounds is around 2.1%. Futhermore, discarding the problems 2 and 4 upper bounds deviate less than 3.65%.

The results of the deflected subgradient optimization method are given in Table 9.3. The average deviations obtained do not differ much from the ones obtained by using the classical subgradient method. Discarding the same problems as in the previous paragraph, lower bound and upper bound deviations are less than 3.51% and 4.34%

respectively for all test problems. It can be said that both classical and deflected subgradient optimization methods provide good bounds. Although classical method solves the test problems in 2946.15 seconds on the average, deflected one solves in 2359.71 seconds. However, since bounds given by the classical method are slightly tighter we cannot say that one method is better than the other for our single-source problem.

Table 9.3: Results for the single-source test problems: Lagrangean heuristic with deflected subgradient optimization

		Deflected Subgradient Optimization				
Problem No	**opt.**	**LB**	**UB**	**LB dev. (%)**	**UB dev. (%)**	**CPU time**
1	1628.32	1300.00	1650.38	20.16	1.35	0
2	1471.80	1191.46	1676.14	19.05	13.88	0
3	1634.33	1614.42	1634.33	1.22	0.00	0
4	2216.81	2139.83	2556.35	3.47	15.32	1
5	2601.80	2613.12	2614.35	-0.43	0.48	0
6	2586.41	2548.89	2667.76	1.45	3.14	2
7	4409.81	4393.02	4448.84	0.38	0.89	3
8	5004.22	4828.65	5118.74	3.51	2.29	8
9	5010.65	5014.96	5135.93	-0.09	2.50	17
10	6708.17	6644.60	6786.59	0.95	1.17	83
11	9152.76	9105.76	9307.51	0.51	1.69	29
12	9476.45	9449.69	9800.05	0.28	3.41	201
13	7753.82	7721.92	8013.73	0.41	3.35	53
14	11808.78	11774.83	11899.06	0.29	0.76	367
15	9543.55	9500.69	9591.55	0.45	0.50	91
16	12979.69	12942.12	13421.90	0.29	3.41	559
17	9175.17	9156.50	9190.24	0.20	0.16	109
18	12032.67	12008.77	12208.68	0.20	1.46	957
19	9698.17	9660.77	10049.60	0.39	3.62	218
20	16885.40	16853.55	17617.43	0.19	4.34	1721
21	13537.65	13523.18	13559.11	0.11	0.16	386
22	27491.55	27470.77	28010.64	0.08	1.89	2203
23	16712.14	16631.63	16799.06	0.48	0.52	454

Continued on Next Page...

		Deflected Subgradient Optimization				
Problem No	**opt.**	**LB**	**UB**	**LB dev. (%)**	**UB dev. (%)**	**CPU time**
24	15436.26	15416.16	15980.61	0.13	3.53	3698
25	19788.97	19790.09	19794.03	-0.01	0.03	649
26	23326.17	23321.51	23347.59	0.02	0.09	4808
27	14531.97	14389.09	14800.81	0.98	1.85	430
28	19806.38	19714.97	20272.57	0.46	2.35	2533
29	13383.94	13310.45	13570.75	0.55	1.40	611
30	25843.96	25754.33	26390.35	0.35	2.11	3267
31	15277.37	15251.01	15407.74	0.17	0.85	705
32	32099.20	32024.53	32494.28	0.23	1.23	4348
33	20079.48	20021.13	20143.45	0.29	0.32	1045
34	24415.84	24403.77	24515.17	0.05	0.41	6603
35	16312.98	16140.61	16521.17	1.06	1.28	1086
36	21993.04	21943.07	22084.23	0.23	0.41	6821
37	17904.23	17900.25	17990.64	0.02	0.48	1700
38	23417.27	23390.51	23796.79	0.11	1.62	7679
39	28176.56	28067.17	28301.90	0.39	0.44	1640
40	30540.42	30518.14	30680.35	0.07	0.46	12283
41	48044.25	47898.09	49144.02	0.30	2.29	29380
Average				**1.57**	**2.13**	**2359.71**

On the other hand, volume algorithm does not provide bounds as good as ones given by deflected and classical subgradient methods. From Table 9.4, it is observed that average deviations of lower bounds and upper bounds from the optimal values are 1.67% and 2.52% respectively. In addition, this method solves the test problems on the average in 5062.29 seconds which is much larger than the time required for the other two methods.

Our Lagrangean heuristics with three different subgradient methods give good bounds in reasonable execution time. Furthermore, the obtained lower bounds are tighter than the lower bounds given by LP relaxations for all test problems. However, it can be noticed that total execution time of CPLEX is better than the execution time of the Lagrangean heuristic with volume algorithm. On the other hand, although

the heuristics using the classical and deflected subgradient optimization methods solve some of the test problems in an execution time larger than that of CPLEX, on the average both methods require less time than CPLEX in order to solve all test problems.

The bounds given by deflected subgradient method are as good as the bounds given by classical method. For some problem instances, classical subgradient optimization requires less time than deflected subgradient optimization. However, deflected subgradient method solves all test problems in less average execution time than other methods. Therefore, taking the average execution time into consideration, Lagrangean heuristic with deflected subgradient optimization method seems more preferable than our other heuristics.

Table 9.4: Results for the single-source test problems: Lagrangean heuristic with volume algorithm

		Volume Algorithm				
Problem No	**opt.**	**LB**	**UB**	**LB dev. (%)**	**UB dev. (%)**	**CPU time**
1	1628.32	1286.06	1650.38	21.02	1.35	0
2	1471.80	1185.03	1583.23	19.48	7.57	0
3	1634.33	1611.83	1634.33	1.38	0.00	0
4	2216.81	2137.01	2538.18	3.60	14.50	1
5	2601.80	2618.07	2616.77	-0.63	0.58	1
6	2586.41	2541.65	2706.71	1.73	4.65	3
7	4409.81	4376.67	4464.64	0.75	1.24	4
8	5004.22	4814.31	5133.76	3.80	2.59	14
9	5010.65	4995.54	5232.76	0.30	4.43	30
10	6708.17	6636.74	6793.56	1.06	1.27	200
11	9152.76	9083.53	9313.28	0.76	1.75	61
12	9476.45	9433.85	9804.27	0.45	3.46	348
13	7753.82	7689.37	8242.33	0.83	6.30	115
14	11808.78	11760.18	11989.93	0.41	1.53	695
15	9543.55	9487.89	9595.02	0.58	0.54	178
16	12979.69	12907.46	13512.31	0.56	4.10	1033
17	9175.17	9152.59	9191.77	0.25	0.18	234

Continued on Next Page...

		Volume Algorithm				
Problem No	**opt.**	**LB**	**UB**	**LB dev. (%)**	**UB dev. (%)**	**CPU time**
18	12032.67	11993.10	12202.45	0.33	1.41	1750
19	9698.17	9639.87	10011.78	0.60	3.23	474
20	16885.40	16757.18	17952.08	0.76	6.32	3206
21	13537.65	13520.24	13561.90	0.13	0.18	743
22	27491.55	27431.13	28683.26	0.22	4.33	4043
23	16712.14	16613.18	16841.81	0.59	0.78	1258
24	15436.26	15367.80	16350.07	0.44	5.92	10515
25	19788.97	19789.41	19810.08	0.00	0.11	1548
26	23326.17	23316.60	23397.82	0.04	0.31	7310
27	14531.97	14371.95	14777.17	1.10	1.69	888
28	19806.38	19686.90	20480.13	0.60	3.40	5188
29	13383.94	13287.05	13590.11	0.72	1.54	2026
30	25843.96	25722.42	26755.22	0.47	3.53	6173
31	15277.37	15238.11	15422.22	0.26	0.95	1730
32	32099.20	31991.77	32716.52	0.33	1.92	8203
33	20079.48	20013.02	20205.22	0.33	0.63	1738
34	24415.84	24392.83	24593.24	0.09	0.73	11556
35	16312.98	16104.68	16513.36	1.28	1.23	3419
36	21993.04	21936.22	22097.48	0.26	0.47	19210
37	17904.23	17884.65	18186.27	0.11	1.58	2962
38	23417.27	23362.67	23703.20	0.23	1.22	30675
39	28176.56	28052.07	28315.78	0.44	0.49	3549
40	30540.42	30495.62	31189.60	0.15	2.13	23723
41	48044.25	47793.60	49635.31	0.52	3.31	52750
Average				**1.67**	**2.52**	**5062.29**

9.2. Multi-Source Problem

9.2.1. Test Bed

The test problems are generated randomly in a way analogous to that of single-source problem. Customer serving costs c_{ikjl}, fixed costs of opening a facility f_{ik}, demands of customers d_{jl}, capacities of facilities s_{ik}, locations of customers and facilities

are obtained using the same distributions and formulations proposed for the single-source problem. The parameter q representing the ratio of the total supply to total demand is taken $q = 0.8, 0.9, 1.0, 1.1, 1.2$. In the multi-source problem, we include fixed costs of serving a customer g_{ikjl}. These costs are generated from normal distribution with mean 5 and standard deviation 10 and again negative numbers are discarded. Mean and standard deviation are selected arbitrarily. We generate 40 test problems.

A strong formulation to our multi-source problem is obtained by modifying the constraints (3.9) as follows:

$$\sum_{j=1}^{J}\sum_{l=1}^{L} d_{jl}x_{ikjl} \leq s_{ik}v_{ik} \quad \text{i = 1,...,I, k = 1,...,K} \tag{9.3}$$

This strong formulation has the same number of constraints with the formulation P2. Therefore, tighter LP relaxation bounds are obtained without increasing the number of constraints. The solutions found using the weak formulation P2 and strong formulation are given in Table 9.5 and Table 9.6 respectively. In Table 9.6, the test problems are solved with small relative tolerances ϑ so the obtained solutions are very close to the optimal solutions. Therefore, these solutions are taken as reference when calculating the deviations of upper and lower bounds given by our Lagrangean heuristics. We use both time and relative tolerance limitations when solving the test problems using the weak formulation. For each problems, the time limitation for CPLEX is set to a time larger than the execution time of our Lagrangean heuristic with classical subgradient optimization. The time limitation is three hours for 34^{th} and 36^{th} problems, five hours for 38^{th} and 40^{th} problems, two hours for the rest. The relative tolerance limitation is taken 0.1 % for all test problems. Because of these limitations, the given objective values are not generally close to the optimal values and they represent an upper bound. The notation used in Table 9.6 is same as the notation in Table 9.1. In Table 9.5 a similar notation is used but in this case instead of relative tolerances, deviations of upper bounds from optimal solutions are indicated. In addition the deviations of lower bounds obtained by LP relaxation are calculated using the near optimal values given in Table 9.6. N/A indicates not available and is used when a solution is not found.

Table 9.5: Optimal and linear programming (LP) relaxation results of the multi-source test problems: weak formulation

Problem No	I	K	J	L	UB	nodes	CPU time	UB dev. (%)	LP relax.	CPU time	LB dev. (%)
1	2	3	10	3	2593.30	14	1.47	0.00	2150.43	0.02	17.08
2	4	3	10	3	2536.55	716	1.83	0.00	1391.03	0.06	45.16
3	4	3	15	3	2474.96	176	1.73	0.00	1889.13	0.05	23.67
4	5	3	15	3	2653.38	5342	11.08	0.03	1578.98	0.06	40.47
5	8	3	25	3	3258.91	88585	423.25	0.00	2358.45	0.14	27.63
6	9	3	25	5	4791.05	525001	7200.11	0.00	3918.43	0.27	18.21
7	10	3	50	3	6138.90	23791	7200.13	0.06	4056.33	0.36	33.88
8	10	4	50	5	7358.29	16474	7200.19	4.60	5024.93	1.44	28.57
9	15	3	100	3	8921.66	27801	7200.77	0.09	7898.81	1.66	11.39
10	25	4	100	5	11223.06	1226	7201.03	10.11	8174.01	17.52	19.80
11	15	3	150	3	9630.25	7200	7200.39	2.77	7598.02	4.59	18.92
12	25	4	150	5	N/A	N/A	7200.00	N/A	13863.82	37.42	18.66
13	15	3	200	3	13657.31	6955	7200.55	1.27	12033.95	5.44	10.77
14	25	4	200	5	19389.58	430	7203.14	4.60	17470.56	37.69	5.75
15	15	3	250	3	14188.15	3726	7200.80	6.76	11659.17	10.70	12.27
16	25	4	250	5	N/A	N/A	7200.00	N/A	19520.74	64.14	12.45
17	15	3	300	3	17999.92	3200	7200.81	5.04	16025.65	9.05	6.48
18	25	4	300	5	35330.98	260	7204.34	42.98	23816.71	80.48	3.62
19	15	3	400	3	16072.29	1482	7201.70	1.16	14457.08	18.05	9.00
20	25	4	400	5	41155.53	80	7205.41	52.30	24937.26	181.86	7.72
21	15	3	500	3	25519.09	9997	7201.89	0.27	24486.94	19.19	3.79
22	25	4	500	5	56970.33	233	7205.41	37.86	39146.21	166.91	5.27
23	15	3	600	3	24251.62	4516	7203.17	0.25	23580.95	31.88	2.52

Continued on Next Page...

Problem No	I	K	J	L	UB	nodes	CPU time	UB dev. (%)	LP relax.	CPU time	LB dev. (%)
24	25	4	600	5	N/A	N/A	7200.00	N/A	40989.97	466.52	5.96
25	15	3	700	3	N/A	N/A	7200.00	N/A	33632.79	44.30	7.21
26	25	4	700	5	N/A	N/A	7200.00	N/A	48957.67	555.78	3.51
27	20	3	500	3	N/A	N/A	7200.00	N/A	27481.53	32.17	6.74
28	30	4	500	5	N/A	N/A	7200.00	N/A	35890.82	359.53	4.51
29	20	3	550	3	N/A	N/A	7200.00	N/A	26942.74	55.91	10.95
30	30	4	550	5	N/A	N/A	7200.00	N/A	36702.09	698.53	10.47
31	20	3	600	3	N/A	N/A	7200.00	N/A	29491.17	56.88	9.13
32	30	4	650	5	N/A	N/A	7200.00	N/A	45668.04	720.14	6.21
33	20	3	700	3	N/A	N/A	7200.00	N/A	24299.40	97.69	7.32
34	30	4	750	5	71250.35	52	10811.08	33.82	51252.91	623.67	3.74
35	20	3	800	3	N/A	N/A	7200.00	N/A	30441.29	91.47	10.52
36	30	4	850	5	N/A	N/A	10800.00	24.92	58814.66	844.97	3.21
37	20	3	900	3	52044.46	368	7207.62	22.89	41148.62	134.99	2.84
38	30	4	950	5	N/A	N/A	18000.00	N/A	49314.69	1536.04	N/A
39	20	3	1000	3	N/A	N/A	7200.00	N/A	36737.97	134.44	3.82
40	30	4	1050	5	N/A	N/A	18000.00	N/A	64618.37	1718.61	N/A
Average							**7032.20**	**10.49**		**221.51**	**12.61**

Table 9.6: Optimal and linear programming (LP) relaxation results of the multi-source test problems: strong formulation

Problem No	I	K	J	L	opt.	nodes	CPU time	ϑ (%)	LP relax.	CPU time	dev. (%)
1	2	3	10	3	2593.30	3	0.28	0.00	2533.35	0.02	2.31
2	4	3	10	3	2536.55	297	0.59	0.00	2450.55	0.03	3.39
3	4	3	15	3	2474.88	176	0.98	0.00	2402.42	0.03	2.93
4	5	3	15	3	2652.60	144	0.50	0.00	2562.63	0.03	3.39
5	8	3	25	3	3258.91	1795	9.75	0.00	3195.34	0.11	1.95
6	9	3	25	5	4791.05	5057	31.80	0.00	4749.98	0.14	0.86
7	10	3	50	3	6135.25	5074	42.97	0.00	5969.77	0.14	2.70
8	10	4	50	5	7034.60	100663	1584.16	0.07	6913.55	2.70	1.72
9	15	3	100	3	8913.91	1780492	111576.06	0.00	8824.62	3.92	1.00
10	25	4	100	5	10192.34	39000	10727.27	0.09	10161.48	33.48	0.30
11	15	3	150	3	9370.70	35937	2105.72	0.05	9292.93	6.69	0.83
12	25	4	150	5	17045.17	20200	15757.73	0.12	16926.85	78.98	0.69
13	15	3	200	3	13485.98	25000	2697.44	0.04	13464.29	8.41	0.16
14	25	4	200	5	18536.19	14800	21518.85	0.05	18428.85	86.22	0.58
15	15	3	250	3	13289.75	9600	2925.16	0.04	13190.47	15.06	0.75
16	25	4	250	5	22297.18	11800	10630.58	0.06	22228.30	178.28	0.31
17	15	3	300	3	17135.98	15775	4802.70	0.03	17043.83	17.39	0.54
18	25	4	300	5	24710.56	16372	59460.75	0.06	24592.78	222.38	0.48
19	15	3	400	3	15887.65	2400	1951.00	0.03	15823.64	22.05	0.40
20	25	4	400	5	27023.27	19650	140178.93	0.67	26682.77	576.28	1.26
21	15	3	500	3	25451.22	79649	39110.81	0.04	25280.50	48.53	0.67
22	25	4	500	5	41323.99	12200	50022.53	0.03	41198.85	408.86	0.30
23	15	3	600	3	24191.27	33580	12934.28	0.03	23953.63	51.59	0.98

Continued on Next Page...

Problem No	I	K	J	L	opt.	nodes	CPU time	ϑ (%)	LP relax.	CPU time	dev. (%)
24	25	4	600	5	43589.24	24494	232000.83	0.12	43145.02	1105.83	1.02
25	15	3	700	3	36247.71	2600	3113.41	0.02	36159.36	161.70	0.24
26	25	4	700	5	50738.04	13483	118946.67	1.70	49781.37	924.09	1.89
27	20	3	500	3	29468.30	38240	42186.30	0.06	29349.59	114.73	0.40
28	30	4	500	5	37584.04	9709	33962.47	0.13	37439.39	679.86	0.38
29	20	3	550	3	30256.85	3800	5514.94	0.03	30153.69	204.94	0.34
30	30	4	550	5	40992.14	400	7972.52	0.13	40936.40	1151.59	0.14
31	20	3	600	3	32453.02	8289	8645.38	0.04	32417.96	192.07	0.11
32	30	4	650	5	48693.83	4882	84452.97	0.14	48377.77	2464.05	0.65
33	20	3	700	3	26218.92	23875	71172.38	0.03	26150.79	282.13	0.26
34	30	4	750	5	53242.96	2200	52796.47	1.00	52504.52	2144.24	1.39
35	20	3	800	3	34018.61	21646	30031.19	0.50	33802.40	228.50	0.64
36	30	4	850	5	60762.75	0	6411.55	0.90	60272.64	1381.75	0.81
37	20	3	900	3	42349.64	2680	10640.83	0.23	42213.56	227.96	0.32
38	30	4	950	5	N/A	N/A	N/A	N/A	52859.64	4089.42	N/A
39	20	3	1000	3	38197.48	10041	10804.73	3.80	38103.70	275.95	0.25
40	30	4	1050	5	N/A	N/A	N/A	N/A	67904.79	4896.78	N/A
Average							**31755.88**			**557.17**	**0.98**

From Tables 9.5 and 9.6, we can see that the good lower bounds are given by strong formulation and they are strictly tighter than the ones given by weak formulation. Furthermore, CPLEX requires less time when solving the problems with strong formulation. Because of the time limitation used in solving weak formulations a comparison is not possible for all problems. However looking at the first eight and $11^{th}, 13^{th}, 15^{th}, 17^{th}, 19^{th}$ problems, we can see the decrease in execution time clearly. It is seen from Table 9.6 that although strong formulation is used, CPLEX is not able to find a solution for 38^{th} and 40^{th} problems.

9.2.2. Performances of the Proposed Methods

We propose two different solution procedures for our multi-source problem. We obtain bounds for optimal solutions using a Lagrangean relaxation based solution procedure which is similar to the method proposed for the single-source problem. In addition, we try to obtain optimal solutions by an exact solution method called Benders' decomposition.

In our Lagrangean heuristic, we again use the classical subgradient optimization, deflected subgradient optimization and volume algorithm. We will not mention the initialization and parameter update procedures of the subgradient methods since the same procedures used for the single-source problem are utilized for the multi-source problem.

Considering the results in Table 9.7, the average deviations of lower and upper bounds from the optimal solutions are respectively 0.79% and 1.44%, which are clear indications of the qualities of bounds obtained using classical subgradient optimization method. In addition, lower bounds do not deviate more than 2.68% for all test problems. The maximum deviation of upper bounds is 3%.

From Table 9.8, it is observed that although deflected subgradient optimization method provides good bounds, in general the upper bounds are not as tight as the upper bounds given by the classical subgradient method. On the average the method provides

a lower bound deviation of 0.8% and upper bound deviation of 1.59%. Futhermore, the maximum deviations are 2.83% and 3.77% for lower bounds and upper bounds respectively.

Table 9.7: Results for the multi-source test problems: Lagrangean heuristic with classical subgradient optimization

		Classical Subgradient Optimization				
Problem No	**opt.**	**LB**	**UB**	**LB dev. (%)**	**UB dev. (%)**	**CPU time**
1	2593.30	2544.54	2593.30	1.88	0.00	0
2	2536.55	2468.49	2548.49	2.68	0.47	0
3	2474.88	2416.05	2484.14	2.38	0.37	1
4	2652.60	2592.20	2710.33	2.28	2.18	0
5	3258.91	3218.05	3342.53	1.25	2.57	0
6	4791.05	4771.10	4808.60	0.42	0.37	1
7	6135.25	6024.99	6156.51	1.80	0.35	3
8	7034.60	6929.95	7173.18	1.49	1.97	12
9	8913.91	8851.10	9049.43	0.70	1.52	18
10	10192.34	10187.41	10493.95	0.05	2.96	94
11	9370.70	9317.95	9461.73	0.56	0.97	44
12	17045.17	16966.03	17288.90	0.46	1.43	180
13	13485.98	13479.26	13649.50	0.05	1.21	82
14	18536.19	18442.01	18775.81	0.51	1.29	464
15	13289.75	13214.22	13398.81	0.57	0.82	108
16	22297.18	22253.71	22808.04	0.19	2.29	546
17	17135.98	17053.36	17358.33	0.48	1.30	184
18	24710.56	24601.58	25151.40	0.44	1.78	1073
19	15887.65	15838.35	16098.88	0.31	1.33	226
20	27023.27	26689.48	27626.95	1.24	2.23	1642
21	25451.22	25284.10	25833.60	0.66	1.50	458
22	41323.99	41208.77	41835.85	0.28	1.24	2386
23	24191.27	23955.94	24385.45	0.97	0.80	552
24	43589.24	43148.02	44895.83	1.01	3.00	3497
25	36247.71	36175.86	36512.15	0.20	0.73	1083
26	50738.04	49776.25	51827.18	1.90	2.15	6075

Continued on Next Page...

		Classical Subgradient Optimization				
Problem No	**opt.**	**LB**	**UB**	**LB dev. (%)**	**UB dev. (%)**	**CPU time**
27	29468.30	29368.07	29976.43	0.34	1.72	538
28	37584.04	37453.67	38235.14	0.35	1.73	2989
29	30256.85	30174.29	30692.74	0.27	1.44	707
30	40992.14	40969.73	41355.43	0.05	0.89	4029
31	32453.02	32451.81	32706.51	0.00	0.78	1104
32	48693.83	48383.84	49982.07	0.64	2.65	6393
33	26218.92	26159.51	26668.13	0.23	1.71	1205
34	53242.96	52501.34	54596.56	1.39	2.54	8622
35	34018.61	33831.34	34362.81	0.55	1.01	1589
36	60762.75	60269.95	61449.49	0.81	1.13	8936
37	42349.64	42218.91	42865.79	0.31	1.22	2816
38	N/A	52877.13	53218.19	N/A	N/A	16310
39	38197.48	38103.85	38600.05	0.25	1.05	2312
40	N/A	67909.63	70621.09	N/A	N/A	17070
Average				**0.79**	**1.44**	**2333.73**

Table 9.9 reveals that volume algorithm is not as successful as the other two methods both in terms of solution quality and execution time, which is also the case for the single-source problem. The average lower bound deviation is 0.99% and the maximum lower bound deviation is 2.76%. The upper bounds deviate from the optimal results on the average 2.36% whereas the maximum upper bound deviation is 5.6%.

Our Lagrangean heuristic with each subgradient method gives good bounds for all problems in reasonable execution time. The average execution time is much more smaller than the time required by CPLEX in all cases. In addition, on the average our Lagrangean heuristics with classical and deflected subgradient optimizations provide tighter bounds than the ones given in Table 9.5 although their execution times are less than the time limits of CPLEX for all test problems. When average execution time is considered, it can also be said that volume algorithm also provides better bounds than CPLEX in less average time. Furthermore, it should be noted that CPLEX is not able to find a solution for nearly one third of the problems with weak formulation in given

time limits. Among the three methods the volume algorithm is the least successful method. When deflected subgradient method is used, a bit less time is required than the classical subgradient method in order to solve all test problems. Average execution times are 2333.73 and 1919.35 for the classical and deflected methods respectively. On the other hand, the classical subgradient method gives tighter bounds.

Table 9.8: Results for the multi-source test problems: Lagrangean heuristic with deflected subgradient optimization

		Deflected Subgradient Optimization				
Problem No	**opt.**	**LB**	**UB**	**LB dev. (%)**	**UB dev. (%)**	**CPU time**
1	2593.30	2546.25	2593.30	1.81	0.00	0
2	2536.55	2464.81	2544.80	2.83	0.33	0
3	2474.88	2412.55	2489.02	2.52	0.57	1
4	2652.60	2595.04	2715.83	2.17	2.38	0
5	3258.91	3215.72	3356.81	1.33	3.00	1
6	4791.05	4767.99	4808.60	0.48	0.37	1
7	6135.25	6030.90	6146.80	1.70	0.19	3
8	7034.60	6936.04	7198.99	1.40	2.34	7
9	8913.91	8844.44	9055.21	0.78	1.59	11
10	10192.34	10187.79	10512.42	0.04	3.14	88
11	9370.70	9316.90	9466.98	0.57	1.03	33
12	17045.17	16965.26	17291.63	0.47	1.45	173
13	13485.98	13479.42	13687.48	0.05	1.49	52
14	18536.19	18438.90	18838.94	0.52	1.63	421
15	13289.75	13211.27	13393.82	0.59	0.78	85
16	22297.18	22251.37	22798.89	0.21	2.25	551
17	17135.98	17055.66	17319.71	0.47	1.07	124
18	24710.56	24600.86	25371.06	0.44	2.67	926
19	15887.65	15840.43	16019.48	0.30	0.83	178
20	27023.27	26686.34	27703.80	1.25	2.52	1655
21	25451.22	25282.69	26000.06	0.66	2.16	283
22	41323.99	41208.55	41864.69	0.28	1.31	2148
23	24191.27	23953.58	24419.82	0.98	0.94	448

Continued on Next Page...

		Deflected Subgradient Optimization				
Problem No	**opt.**	**LB**	**UB**	**LB dev. (%)**	**UB dev. (%)**	**CPU time**
24	43589.24	43145.69	45232.41	1.02	3.77	3014
25	36247.71	36175.39	36544.40	0.20	0.82	623
26	50738.04	49772.92	51957.18	1.90	2.40	4184
27	29468.30	29370.92	29945.47	0.33	1.62	555
28	37584.04	37449.83	38399.14	0.36	2.17	2127
29	30256.85	30172.32	30619.38	0.28	1.20	549
30	40992.14	40969.51	41389.85	0.06	0.97	3057
31	32453.02	32445.38	32818.96	0.02	1.13	886
32	48693.83	48383.20	50022.52	0.64	2.73	4895
33	26218.92	26157.73	26632.26	0.23	1.58	920
34	53242.96	52498.44	54267.49	1.40	1.92	7238
35	34018.61	33829.38	34599.86	0.56	1.71	1167
36	60762.75	60268.44	62022.21	0.81	2.07	8130
37	42349.64	42216.05	42900.10	0.32	1.30	1630
38	N/A	52872.39	53207.04	N/A	N/A	12865
39	38197.48	38106.47	38582.15	0.24	1.01	2302
40	N/A	67903.73	71132.97	N/A	N/A	15443
Average				**0.80**	**1.59**	**1919.35**

In the proposed Benders' decomposition algorithm, the subproblems and the master problem are solved to optimality at each iteration using the solver CPLEX 10.1. At each iteration whenever we obtain an extreme point cut, we solve the linear program suggested by Magnanti and Wong in order to find a pareto-optimal cut. We test this algorithm on first ten test problems and limit the execution time to two hours on the average. The obtained upper and lower bounds are provided in Table 9.10. Although we are able to obtain optimal solutions for the first four problems, comparing the results with Table 9.5 it is clear that the algorithm is not competitive with CPLEX taking the execution time into consideration. Because a mixed-integer programming problem (master problem) is solved to optimality at each iteration, the overall computation time is large. The algorithm provides very loose bounds for 8^{th} and 8^{th} problems and it is not able to find bounds for the 10^{th} problem.

Table 9.9: Results for the multi-source test problems: Lagrangean heuristic with volume algorithm

		Volume Algorithm				
Problem No	**opt.**	**LB**	**UB**	**LB dev. (%)**	**UB dev. (%)**	**CPU time**
1	2593.30	2543.75	2593.30	1.91	0.00	0
2	2536.55	2466.44	2546.04	2.76	0.37	0
3	2474.88	2408.38	2488.14	2.69	0.54	0
4	2652.60	2588.63	2716.97	2.41	2.43	0
5	3258.91	3202.88	3359.78	1.72	3.10	1
6	4791.05	4764.12	4809.72	0.56	0.39	3
7	6135.25	5999.43	6152.45	2.21	0.28	5
8	7034.60	6892.00	7315.51	2.03	3.99	16
9	8913.91	8826.40	9106.34	0.98	2.16	27
10	10192.34	10174.79	10499.51	0.17	3.01	161
11	9370.70	9308.79	9489.50	0.66	1.27	58
12	17045.17	16944.74	17548.61	0.59	2.95	376
13	13485.98	13462.72	13765.47	0.17	2.07	101
14	18536.19	18389.22	18931.78	0.79	2.13	609
15	13289.75	13193.20	13404.40	0.73	0.86	184
16	22297.18	22203.34	23027.60	0.42	3.28	1168
17	17135.98	17032.04	17399.70	0.61	1.54	276
18	24710.56	24553.76	25416.36	0.63	2.86	1399
19	15887.65	15793.55	16368.39	0.59	3.03	406
20	27023.27	26637.15	27577.53	1.43	2.05	3494
21	25451.22	25200.34	26307.87	0.99	3.37	619
22	41323.99	41151.54	42179.42	0.42	2.07	4112
23	24191.27	23906.42	24528.95	1.18	1.40	1177
24	43589.24	42952.64	45479.82	1.46	4.34	6259
25	36247.71	36159.31	36471.34	0.24	0.62	1273
26	50738.04	49632.22	52096.39	2.18	2.68	8321
27	29468.30	29298.53	30529.56	0.58	3.60	923
28	37584.04	37386.06	38699.38	0.53	2.97	5187
29	30256.85	30132.46	30816.26	0.41	1.85	1178
30	40992.14	40900.14	43249.78	0.22	5.51	5794
31	32453.02	32424.15	33001.74	0.09	1.69	1580

Continued on Next Page...

		Volume Algorithm				
Problem No	**opt.**	**LB**	**UB**	**LB dev. (%)**	**UB dev. (%)**	**CPU time**
32	48693.83	48288.33	50518.21	0.83	3.75	10516
33	26218.92	26091.69	27303.28	0.49	4.14	2023
34	53242.96	52402.59	54117.53	1.58	1.64	14811
35	34018.61	33769.36	35922.63	0.73	5.60	2098
36	60762.75	60185.84	61979.38	0.95	2.00	18645
37	42349.64	42171.84	43164.55	0.42	1.92	3719
38	N/A	52799.13	54195.77	N/A	N/A	26005
39	38197.48	38034.98	39080.93	0.43	2.31	3455
40	N/A	67733.92	71789.35	N/A	N/A	24201
Average				**0.99**	**2.36**	**3754.50**

Table 9.10: Results for the multi-source test problems: Benders' decomposition

		Benders' Decomposition				
Problem No	**opt.**	**LB**	**UB**	**LB dev. (%)**	**UB dev. (%)**	**CPU time**
1	2593	2593	2593	0.00	0.00	8.53
2	2537	2537	2537	0.00	0.00	82.81
3	2475	2475	2475	0.00	0.00	416.97
4	2653	2653	2653	0.00	0.00	335.27
5	3259	2952	3518	9.42	7.94	7899.31
6	4791	4432	5399	7.50	12.69	8087.61
7	6135	5699	6703	7.10	9.26	7269.42
8	7035	5456	13582	22.44	93.07	7713.58
9	8914	6773	16043	24.02	79.98	7264.64
10	10192	N/A	N/A	N/A	N/A	7200.00
Average				**7.83**	**22.55**	**4627.814**

10. CONCLUSION AND FUTURE WORK

Throughout this study, we are concerned with the CFLP. We assume that each market consists of customers with different profiles and consider the situation where additional costs occur when the customers are not served by the right type of facility. We present mathematical formulations for the single-source and multi-source versions of this problem and then attempt to solve them.

CFLPs are difficult combinatorial optimization problems. As a consequence of this fact, many heuristic solutions are proposed in the literature in order to obtain near optimal solutions. A successful example of these heuristic methods is Lagrangean relaxation which is widely used to provide bounds for the CFLPs. Therefore, at the initial phases of our study, we make use of the Lagrangean relaxation with classical subgradient optimization for our two problem types. By comparing our results with the optimal values of our test problems, we notice that lower and upper bounds are fairly close to the optimal values for both problems. However, although our Lagrangean heuristic requires much less execution time than CPLEX in order to provide bounds for the multi-source problem, this case is not valid for the single-source problem. Consequently, we decide to use subgradient optimization methods different than the classical method in order to see their effects on both execution time and solution quality. Deflected subgradient optimization provides bounds that are closer to the bounds of the classical subgradient method in shorter average computation time. However, the decrease in the average computation time is not so significant. This situation is valid for both single-source and multi-source problems. On the other hand, volume algorithm provides an improvement neither in bounds nor in average computation time.

In addition to the heuristic methods, there exist algorithms that give optimal solutions. We adopt Benders' decomposition method to the multi-source problem in order to solve it optimally. We are able to obtain optimal solutions for small sized test problems, but the execution time considerably increases when the problem size grows. Although we attempt to decrease the overall computation time by employing some

modifications such as finding pareto-optimal cuts and providing additional constraints for the master problem, we do not obtain a significant improvement.

Our Benders' decomposition and Lagrangean heuristic can be combined in order to develop an efficient exact solution method. The feasible solutions given by Lagrangean upper bound heuristic can be used to find extreme point cuts for the master problem. Therefore, we can develop a Benders' decomposition algorithm composed of two phases. In the initial phase, the cuts can be added to the master problem by only solving the Benders' subproblem. There is no need to solve the master problem since we already have a sample of feasible solutions. In the second phase, the iterative procedure of our Benders' algorithm between the master problem and subproblem is executed. It may be possible that the LP relaxation of the master problem obtained in the initial phase is very close to the optimal solution if enough cuts are added. In this case the solution given by LP relaxation can be made feasible for the original problem using a heuristic method. Therefore instead of solving the master problem, its LP relaxation can be solved in the second phase. To obtain a feasible solution a tabu search procedure can be utilized so that the solutions obtained before are added to the tabu list.

What is more, although our Benders' algorithm is very inefficient, its applicability to our multi-source problem is promising for similar exact solution methods. Consequently, Dantzig-Wolfe decomposition and column generation methods can be utilized to find exact solutions for the multi-source problem.

When the LP relaxation of the strong formulation proposed for the multi-source problem is solved, very tight lower bounds are obtained. This formulation can be made much stronger through generalizing the strong valid inequalities proposed for UFLPs and other CFLPs.

REFERENCES

1. Agar, M.C. and S. Salhi, "Lagrangean heuristics applied to a variety of large capacitated plant location problems", *Journal of the Operational Research Society*, No. 49, pp. 1072-1084, 1998.

2. Ahuja, R.K., J.B. Orlin, S. Pallottino, M.P. Scaparra and M.G. Scutella, "A multi-exchange heuristic for the single-source capacitated facility location problem", *Management Science*, No. 50-6, pp. 749-760, 2004.

3. Barahona, F. and R. Anbil, *The Volume Algorithm: producing primal solutions with a subgradient method*, Research Report, RC 21103(94395), IBM Watson Research Center, 1997.

4. Bazaraa, M.S., H.D. Sherali and C.M. Shetty, *Nonlinear Programming: Theory and Algorithms*, John Wiley and Sons, New York, 1993.

5. Beasley, J.E., "Lagrangean heuristics for location problems", *European Journal of Operations Research*, No. 65, pp. 383-399, 1993.

6. Reeves, C.R., *Modern heuristic techniques for combinatorial problems*, John Wiley and Sons, New York, 1993.

7. Benders, J.F., "Partitioning producers for solving mixed-variables programming problems", *Numerische Mathematik*, No. 4, pp. 238-252, 1962.

8. Bestsekas, D.P., *Constrained optimization and Lagrangean multiplier methods*, Academic Press, New York, 1982.

9. Bramel, J. and D. Simchi-Levi, *The Logic of Logistics: Theory, Algorithms, and Applications for Logistics Management*, Springer Series in Operations Research, 1999.

10. Burachik, R.S., R.N. Gasimov, N.A. Ismayilova and C.Y. Kaya, "On a modified subgradient algorithm for dual problems via sharp augmented Lagragian", *Journal of Global Optimization*, No. 34, pp. 55-78, 2006.

11. Camerini, P.M., L. Fratta and F. Maffioli, "On improving relaxation methods by modified gradient techniques", *Mathematical Programming Study*, No. 3, pp. 26-34, 1975.

12. Cornuéjols, G., R. Sridharan and J.M. Thizy, "A Comparison of Heuristics and Relaxations for the Capacitated Plant Location Problem", *European Journal of Operations Research*, No. 50, pp. 280-297, 1991.

13. Cornuéjols, G., G.L. Nemhauser and L.A. Wolsey, *The Uncapacitated Facility Location Problem*, in P.B. Mirchandani and R. L. Francis (eds), Discrete Location Theory, Wiley, New York, pp. 119-171, 1990.

14. Correia, I. and M.E. Captivo, "A Lagrangean heuristic for a modular capacitated location problem", *Annals of Operations Research*, No. 122, pp. 141-161, 2003.

15. Cortinhal, M.J. and M.E. Captivo, "Upper and lower bounds for the single source capacitated location problem", *European Journal of Operational Research*, No. 151, pp. 333-351, 2003.

16. Fisher, M.L., "The Lagrangean relaxation method for solving integer programming problems", *Management Science*, No. 27-1, pp. 1-18, 1981.

17. Gasimov, R.N., "Augmented Lagrangean duality and nondifferentiable optimization methods in nonconvex programming", *Journal of Global Optimization*, No. 24, pp. 187-203, 2002.

18. Gasimov, R.N., O. Ustun and A. M. Rubinov, "The modified subgradient algorithm based on feasible values", *Journal of Global Optimization*, Article in Press.

19. Geoffrion, A.M., "Lagrangean relaxation and its uses in integer programming",

Mathematical Programming, No. 2, pp. 82-114, 1974.

20. Geoffrion, A.M. and G.W. Graves, "Multicommodity distribution system design by Benders' decomposition", *Management Science*, No. 5-20, pp. 822-844, 1974.

21. Görtz, S. and A. Klose, *Analysis of Some Greedy Algorithms for the Single-Sink Fixed-Charge Transportation Problem*, Working Paper, No. 2006/6.

22. Guta, B., *Subgradient optimization methods in integer programming with an application to a radiation therapy problem*, PhD Thesis, Kaiserslautern University, 2003.

23. Held, M. and R.M. Karp, "The traveling salesman problem and minimum spanning trees", *Operations Research*, No. 18, pp. 1138-1162, 1970.

24. Hindi, K.S. and K. Pieńkosz, "Efficient solution of large scale, single-source, capacitated plant location problems", *Journal of the Operational Research Society*, No. 50, pp. 268-274, 1999.

25. Ioffe, A., "Necessary and sufficient conditions for a local minimum. 3: Second-order conditions and augmented duality", *SIAM J. Control and Optimization*, No. 17, pp. 266-288, 1979.

26. Kalvelagen, E., *Benders decomposition with gams*, http://www.gams.com/ erwin/benders/benders.pdf

27. Karp, R.M., *Reducibility among combinatorial problems*, Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, eds, Plenum Press, New York, 1972.

28. Kellerer, H., U. Pferschy and D. Pisinger, *Knapsack problems*, Springer, Berlin, 2004.

29. Klose, A. and A. Drexl, "Facility location models for distribution system design",

European Journal of Operational Research, Article in Press.

30. Klose, A., "An LP-based heuristic for two stage capacitated facility location problems", *Journal of the Operational Research Society*, No. 50, pp. 157-166, 1999.

31. Li, H.L., "An approximate method for local optima for nonlinear mixed integer programming problems", *Computers and Operations Research*, No. 19-5, pp. 435-444, 1992.

32. Magnanti, T.L. and R.T. Wong, "Accelerating Benders' decomposition: Algorithmic enhancement and model selection criteria", *Operations Research*, No. 29-3, pp. 464-484, 1981.

33. Mazzola, J.B. and A.W. Neebe, "Lagrangean-relaxation-based solution procedures for a multiproduct capacitated facility location problem with choice of facility type", *European Journal of Operations Research*, No. 115, pp. 285-299, 1999.

34. Mirchandani, P.B. and R.L. Francis, *Discrete Location Theory*, Wiley, New York, 1990.

35. ReVelle, C.S. and H.A. Eiselt, "Location Analysis: A synthesis and survey", *European Journal of Operations Research*, No. 165, pp. 1-19, 2005.

36. ReVelle, C.S. and G. Laporte, "The plant location problem: new models and research prospects", *Operations Research*, No. 44-6, pp. 864-874, 1996.

37. Rockafellar, R.T., "Augmented Lagrange multiplier functions and duality", *J. Control and Optim.*, No. 12, pp. 268-285, 1974.

38. Rockafellar, R.T., "Lagrange multipliers and optimality", *SIAM Rev.*, No. 35, pp. 183-238, 1993.

39. Rockafellar, R.T. and R.J B. Wets, *Variational Analysis*, Springer, Berlin, 1998.

40. Ross, G.T. and R.M. Soland, "A branch and bound algorithm for the generalized assignment problem", *Mathematical Programming*, No. 8, pp. 91-103, 1975.

41. Sherali, H.D. and O. Ulular, "A primal-dual conjugate subgradient algorithm for specially structured linear and convex programming problems", *Applied Mathematics and Optimization*, No. 20, pp. 193-221, 1989.

42. Saraç, T. and A. Sipahioğlu, *0-1 Sırt çantası probleminin çözümünde genişletilmiş subgradient yönteminin kullanımı*, YA/EM'2004 - Yöneylem Araştırması/Endüstri Mühendisliği - XXIV Ulusal Kongresi, 2004.

43. Wentges, P., "Accelerating Benders' decomposition for the capacitated facility location problem", *Mathematical Methods of Operations Research*, No. 44, pp. 267-290, 1996.