SOLVING THE CAPACITATED MULTIFACILITY WEBER PROBLEM APPROXIMATELY

by

Burak Boyacı

B.S., Industrial Engineering, Boğaziçi University, 2006

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
2009

SOLVING THE CAPACITATED MULTIFACILITY WEBER PROBLEM APPROXIMATELY

APPROVED BY:

Prof. İ. Kuban Altınel (Thesis Supervisor)	....
Assoc. Prof. Necati Aras	....
Assoc. Prof. Temel Öncan	....

DATE OF APPROVAL: 02.06.2009

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis supervisor Prof. İ. Kuban Altınel for his guidance, encouragement and patience throughout this research. Apart from this thesis, he has always been the one whose suggestions I truly needed. It is an honour for me to have studied with him.

I also would like to thank Assoc. Prof. Necati Aras who was always available when I needed advice while working on my thesis.

Assoc. Prof. Temel Öncan deserves special thanks for taking time to examine this thesis and take part in my thesis jury.

I am grateful to all my friends but especially Hande, Hakan and my lab-mates Güray, Esra and Bilge for their friendship, help and company during my hardest times.

I am also grateful to my family, for their continuous support and endless love for me not only throughout this thesis but my whole life.

Finally I want to thank Belkıs for her everlasting support. She is the one who tolerated and heartened me during the hardest times while writing my thesis most.

I gratefully acknowledge the support of TÜBİTAK - Turkish Technological and Scientific Research Institute during my thesis period.

This research has been partly supported by the Boğaziçi University Research Fund Grant No. 08A304D.

ABSTRACT

SOLVING THE CAPACITATED MULTIFACILITY WEBER PROBLEM APPROXIMATELY

In this study, we consider the capacitated multifacility Weber problem which is concerned with locating m facilities in the plane, and allocating their limited capacities to n customers at minimum total cost. In this group of location-allocation problems, the only cost dealt with is the transportation cost that is proportional to the distance between the facility and the customer. The capacities of each facility and the demands and the locations of each customer are predetermined and given as parameters. This problem is an intractable non-convex optimization problem and difficult to solve. Therefore, using approximation strategies to compute efficient and accurate lower and upper bounds for the capacitated multifacility Weber problem can be a good approach.

We first concentrate on the alternating location allocation heuristics. Then we continue with the discretization strategies and the Lagrangean relaxations of the approximating models. Some specific lower bounding algorithms are also defined by using the special properties of some of the distance functions. In addition to them, the relaxation of the main model is investigated and a Lagrangean heuristic is devised. In this heuristic, either a linear relaxation or exact solution of the Lagrangean subproblem is found by using column generation and branch and price algorithms combined with concave minimization.

Although an exact solution methodology is not found, the approximation methods give accurate results. The tight bounds calculated by using these algorithms can be convenient in searching the exact solutions for this group of problems.

ÖZET

SINIRLI SIĞALI ÇOK TESİSLİ WEBER PROBLEMİ İÇİN YAKLAŞIK ÇÖZÜM YÖNTEMLERİ

Bu çalışmada enküçük maliyetle, m tane tesisin düzleme yerleştirilmesi ve sınırlı sığalarıyla n müşterinin istemlerinin karşılanmasını amaçlayan sınırlı sığalı çok tesisli Weber problemi üzerinde çalışıldı. Bu tür tesis yerleştirme atama problemlerindeki tek maliyet tesis ve müşteri arasındaki uzaklıkla doğru orantılı olan taşıma gideridir. Tesis sığaları ile müşteri yerleri ve istemleri önceden belirlenmiştir ve problem için veridir. Bu problem çözümü zor olan, bir dışbükey olmayan eniyileme problemidir ve bazı yaklaşık çözüm yöntemleriyle eniyi amaç fonksiyonu için üst ve alt sınırlar bulunup sürekli olarak iyileştirilmesi iyi bir yaklaşım olabilir.

Yaklaşık çözüm yöntemlerine değişmeli yerleştirme atama benzeri sezgiselleriyle başlandı ve asıl problemin kesikli uyarlaması ile onun Lagrange gevşetmesi ile devam edildi. Bazı alt sınır algoritmaları da bazı uzaklık normlarının özel niteliklerinden faydalanılarak tanımlandı. Bunlara ek olarak, asıl model gevşetilerek incelendi ve bir Lagrange sezgiseli tasarlandı. Bu sezgiselde, Lagrange alt problemin ya doğrusal gevşetmesi ya da tam sonucu sütun üretme ve dal maliyetlendir algoritmaları kullanılarak ve içbükey enküçükleme alt problemleri çözülerek bulundu.

Her ne kadar bu çözüm yöntemleriyle her zaman kesin çözüm bulunamasa da yaklaşık çözüm yöntemleri ümit verici sonuçlar verdi. Bu algoritmalar kullanılarak hesaplanan sıkı sınırlar, bu grup problemler için kesin çözüm arayan yöntemlerde de yararlı olabilirler.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$\mathbf{A}$	First part of the technology matrix
$\mathbf{a}_j = \begin{pmatrix} a_{j1} & a_{j2} \end{pmatrix}^T$	Coordinates of customer j
$\mathbf{b}$	First part of the resource vector
$\mathbf{B}$	Second part of the technology matrix
$\mathcal{B}$	Set of unsolved nodes in the branch and price algorithm
b_{jit}	Binary parameter to show if customer j is supplied in column t of facility i
$b_{ji}^{(t)}$	Binary variable to show if customer j is supplied in the column of facility i created at iteration t
b_{js}	Binary parameter to show if customer j exist in subset s or not
$\mathbf{b}_k = \begin{pmatrix} b_{k1} & b_{k2} \end{pmatrix}^T$	Coordinates of candidate location k
$\mathbf{c}$	Cost vector
C_i^*	Optimal objective value for subproblem RDCMLAP^i of facility i
c_{ij}	Unit shipment cost per unit amount per unit distance between facility i and customer j
$\tilde{c}_{ijk}$	Cost of assigning facility i located on candidate location k to customer j in RDCMLAP^i
c_{is}	Objective coefficient of subset s of facility i
c_{it}	Objective coefficient of column t of facility i
$c_i^{(t)}$	Cost of a column at iteration t
$\overline{c}_i^{(t)}$	Reduced cost of a column at iteration t
$\overline{c}_i^{(t)*}$	The minimum reduced cost calculated among the columns of facility i at iteration t
C_{LB}	Lower bound calculated in the most recent iteration
c_s	Optimal objective function value for the SFWP over subset s
$\overline{c}^{(t)*}$	The minimum reduced cost among all columns at iteration t
C_{UB}	Lower bound calculated in most recent iteration
$\mathbf{d}$	Second part of the resource vector

$d^{(t)}(i,j)$	Distance between facility i and customer j at iteration t
$d(\mathbf{x}_i, \mathbf{a}_j)$	Function to measure the distance between points $\mathbf{x}_i$ and $\mathbf{a}_j$
$\bar{d}(\mathbf{x}_i, \mathbf{a}_j)$	Given distance between points $\mathbf{x}_i$ and $\mathbf{a}_j$ in RCMFWP
$\mathcal{D}$	Initial feasible set in outer approximation
$\mathcal{D}^{(t)}$	Outer set in outer approximation at iteration t
e	Index of iterations
$\mathbf{G}$	Subgradient vector
$\mathcal{G}$	Set of cuts in outer approximation method
$g()$	A cut function used in outer approximation method
G_i	The i$^{\text{th}}$ entry of the subgradient vector G
G_j	The j$^{\text{th}}$ entry of the subgradient vector G
$\mathcal{H}$	Convex hull of customers
h_j	Demand of customer j
$\tilde{h}_j$	Remaining demand of customer j in RDCMLAP^i
i	Index of facilities
$\bar{i}$	Index of facilities
j	Index of customers
$\mathcal{J}$	Customer set
k	Index of candidate locations
$\mathcal{K}$	Candidate location set
k_i^*	Optimal candidate location for facility i in RDCMLAP^i
$l^{(t)}$	Cut function created in outer approximation method at iteration t
m	Number of facilities to be located
$\mathcal{M}_l = \{\mathcal{M}_l^0, \mathcal{M}_l^1\}$	Zero and one branched customer pair set of node l in BP algorithm
n	Number of customers
$\mathcal{N}$	Not branched customers set in BP algorithm
$\mathcal{O}$	One branched customer pair set whose elements are only branched once
P	Number of rejected iterations required to update radius in the region rejection type heuristics

$\mathcal{P}$	Hyperplane used in outer approximation method
$Q(\mathbf{x}_i, \tilde{R}^e)$	Total demand of customers inside circle with center of $\mathbf{x}_i$ and radius $\tilde{R}^e$
r	Number of candidate locations
r_i	Dynamic radius assigned to facility i
r_i	Additional variable used in concave minimization problem for facility i
R	Initial radius in region rejection type heuristics
$\tilde{R}^{(e)}$	Dynamic radius found at iteration e
s	Index of customer subsets
$\mathcal{S}$	Special set of customer group sets in the branch and price algorithm
$\mathcal{S}_k$	Customer group set k in a special set $\mathcal{S}$
$\mathcal{S}_{kl}$	Customer group l in customer group set k of special set $\mathcal{S}$
s_{klp}	Customer p in customer group l in customer group set k of special set $\mathcal{S}$
s_i	Capacity of facility i
$\mathcal{S}_s$	Subset s of customers set
t	Index of iterations
T	Step size in subgradient optimization
u_i	Dual variable value of supply constraints for facility i in RCM-FWP
$\mathcal{V}$	Set of vertices used in the outer approximation algorithm
v_j	Dual variable value of demand constraints for customer j in RCMFWP
$u_i^{(t)}$	Dual variable value of supply constraints for facility i in RCM-FWP at iteration t
$v_j^{(t)}$	Dual variable value of demand constraints for customer j in RCMFWP at iteration t
w	Dual variable value of total number of facilities constraint in RCMFWP
w_{ij}	Amount of goods to be shipped from facility i to customer j
$\overline{w}_{ij}$	Fixed amount of goods to be shipped from facility i to customer j

w_{ijk}	Amount of goods to be shipped from facility i located on candidate location k to customer j
$w^{(t)}$	Dual variable value of total number of facilities constraint in RCMFWP at iteration t
$\mathbf{x}$	Decision variable vector
$\mathbf{x}_i = \begin{pmatrix} x_{i1} & x_{i2} \end{pmatrix}^T$	Coordinates of facility i
$\overline{\mathbf{x}}_i = \begin{pmatrix} \overline{x}_{i1} & \overline{x}_{i2} \end{pmatrix}^T$	Fixed coordinates of facility i
$\tilde{\mathbf{x}}_i = \begin{pmatrix} \tilde{x}_{i1} & \tilde{x}_{i2} \end{pmatrix}^T$	Randomly assigned coordinates of facility i
x_{ik}	Binary decision variable to determine if facility i is located on candidate location k
$\mathbf{x}_i^{(t)} = \begin{pmatrix} x_{i1}^{(t)} & x_{i2}^{(t)} \end{pmatrix}^T$	Coordinates of facility i at iteration t
$\mathbf{x}_{it} = \begin{pmatrix} x_{i1t} & x_{i2t} \end{pmatrix}^T$	Coordinates of facility i at column t
$\mathbf{x}_s$	Optimal location for the SFWP over subset s
y_{ij}	Binary variable to determine if customer j is supplied by facility i or not
$\mathcal{Z}$	Zero branched customer pair set whose elements are only branched once
z_{is}	Binary variable to determine if subset s is completely served by facility i or not
z_{it}	Binary variable to determine if column t of facility i is selected or not
Z_l	Lower bound for node l
Z_{LB}	Current best lower bound so far
Z_p^*	Optimal objective value of location allocation problem for l_p distance
z_s	Binary variable to determine if subset s is selected or not
Z_{UB}	Current best upper bound so far
α	Radius update coefficient in region rejection type heuristics
$\alpha_i^{(t)\prime}(\mathbf{x}_i, j)$	Function used in subproblem derivations of the branch and price algorithm
$\overline{\alpha}_i^{(t)}(\mathbf{x}_i, j)$	Function used in subproblem derivations of the branch and price algorithm

$\underline{\alpha}_i^{(t)}(\mathbf{x}_i, j)$	Function used in subproblem derivations of the branch and price algorithm
β	Initial radius multiplier in RRH$'$ and DRRH$'$
$\beta_i^{(t)\prime}(\mathbf{x}_i, j_1, j_2)$	Function used in subproblem derivations of the branch and price algorithm
$\overline{\beta}_i^{(t)}(\mathbf{x}_i, j_1, j_2)$	Function used in subproblem derivations of the branch and price algorithm
$\underline{\beta}_i^{(t)}(\mathbf{x}_i, j_1, j_2)$	Function used in subproblem derivations of the branch and price algorithm
$\gamma_i^{(t)\prime}(\mathbf{x}_i, j)$	Function used in subproblem derivations of the branch and price algorithm
$\delta_i^{(t)\prime}(\mathbf{x}_i, j_1, j_2)$	Function used in subproblem derivations of the branch and price algorithm
$\overline{\delta}_i^{(t)}(\mathbf{x}_i, j_1, j_2)$	Function used in subproblem derivations of the branch and price algorithm
$\underline{\delta}_i^{(t)}(\mathbf{x}_i, j_1, j_2)$	Function used in subproblem derivations of the branch and price algorithm
θ	Lower limit coefficient in RRH$'$ and DRRH$'$
$\theta_i(\mathbf{x}_i, \mathbf{a}_j)$	Function used in subproblem derivations of the column generation and the branch and price algorithms
$\overline{\theta}_i^{(t)}(\mathbf{x}_i, \{\mathcal{S}_1, ..., \mathcal{S}_n\})$	Function used in subproblem derivations of the branch and price algorithm
$\underline{\theta}_i^{(t)}(\mathbf{x}_i, \{\mathcal{S}_1, ..., \mathcal{S}_n\})$	Function used in subproblem derivations of the branch and price algorithm
$\boldsymbol{\lambda}$	Lagrangean multiplier vector
λ_i	The ith entry of the Lagrangean multiplier vector
$\lambda_i^{(t)}$	The ith entry of the Lagrangean multiplier vector at iteration t
$\mu_i^{(t)\prime}(\mathbf{x}_i, \{\mathcal{P}_1, ..., \mathcal{P}_n\})$	Function used in subproblem derivations of the branch and price algorithm
μ_j	The jth entry of the Lagrangean multiplier vector
$\nu_i^{(t)\prime}(\mathbf{x}_i, \mathcal{P})$	Function used in subproblem derivations of the branch and price algorithm

ξ	Multiple acceptance limit used in creating columns in the column generation and the branch and price algorithms
π	Step size parameter in subgradient optimization algorithm
$\rho_i^{(t)}\left(\mathbf{x}_i, \left\{\mathcal{S}_1, ...\mathcal{S}_n\right\}, q\right)$	Function used in subproblem derivations of the branch and price algorithm
$\psi_i^{(t)}$	Constant part of the concave minimization problem in the column generation and the branch and price algorithms

ALA	Alternating location allocation
BP	Column generation with branch and price
BPA	Column generation with branch and price algorithm
BPH	Column generation with branch and price heuristic
BP^i	Subproblem for facility i in the branch and price algorithm
CALA	Capacitated alternating location problem
CG	Column generation
CGA	Column generation algorithm
CGH	Column generation heuristic
CG^i	Subproblem for facility i in column generation
CLAP	Continuous location allocation problem
CMFWP	Capacitated multifacility Weber problem
DAH	Discrete approximation heuristic
DALA	Discrete alternating location allocation heuristic
DC	Difference of convex functions
DCALA	Discrete capacitated alternating location allocation heuristic
DCMLAP	Discrete capacitated multifacility location allocation problem
DLAP	Discrete location allocation problem
DRRH	Discrete region rejection heuristic
DRRH$'$	Discrete region rejection heuristic with dynamic radius
DUMLAP	Discrete uncapacitated multifacility location allocation problem
LB	Lower bound
LDP	Lagrangean dual problem

LP	Linear programming
L_pCMFWP	l_p distance capacitated multifacility Weber problem
LR	Lagrangean relaxation
LSP	Lagrangean subproblem
L_1	Discrete approximation method by using l_1 distance
L_∞	Discrete approximation method by using l_∞ distance
MFWP	Multifacility Weber problem
MILP	Mixed integer linear programming
MP	Main problem
NP	Nondeterministic polynomial
RCMFWP	Relaxed capacitated multifacility Weber problem
RDAH	Relaxed discrete approximation heuristic
RDCMLAP	Relaxed discrete capacitated multifacility location allocation problem
RDCMLAP^i	Subproblem associated with each facility i for RDCMLAP
RDUMLAP	Relaxed discrete uncapacitated multifacility location allocation problem
RDUMLAP$_i$	Subproblem associated with each facility i for RDUMLAP
RCMFWP$_2$	Mixed integer binary programming formulation for RCMFWP
RCMFWP$_3$	Set partitioning problem formulation for RCMFWP
RCMFWP$_4$	Set covering problem formulation for RCMFWP
RCMFWP$_5$	Column generation master problem formulation for RCMFWP
RLT	Reformulation linearization technique
RL$_1$	Relaxed discrete approximation method by using l_1 distance
RL$_\infty$	Relaxed discrete approximation method by using l_∞ distance
RRH	Region rejection heuristic
RRH$'$	Region rejection heuristic with dynamic radius
RUDAH	Relaxed uncapacitated discrete approximation heuristic
SFWP	Single facility Weber problem
SO	Subgradient optimization
SPP	Set partitioning problem

TP	Transportation problem
UB	Upper bound
UDAH	Uncapacitated discrete approximation heuristic
UMFWP	Uncapacitated multifacility Weber problem
UMFWP$_2$	Set partitioning problem formulation for UMFWP

1. INTRODUCTION

Facility location allocation problems are used for deciding the locations of new facilities –such as warehouses, factories and retailers– and the allocation of customers to these new facilities. This decision is aimed to minimize the total cost composed of the summation of transportation cost, which is the product of the defined distance between customers and the supplying facility and the amount of product carried from the given facility to the customer, and the fixed cost of opening facilities. If the possible locations for the facilities is chosen from a set of candidate facility locations, problem becomes discrete location allocation problem (DLAP) whereas if the facilities can be located anywhere in the Euclidean space, the problem is named as continuous location allocation problem (CLAP).

In the (single facility) Weber problem (SFWP), the aim is to find the optimal location for a single facility in the Euclidean space which minimizes the total transportation cost. Similar to SFWP, in the multifacility Weber Problem (MFWP), facility location costs are not included in the objective, the only cost dealt with is the transportation cost. In addition to that, the number of facilities is predetermined, given as a parameter and facilities can be located anywhere in the continuous Euclidean space.

In the capacitated MFWP (CMFWP), each facility has a predetermined capacity to supply customers whereas in the (uncapacitated) MFWP (UMFWP) facilities do not have any capacity limitation so every customer is served by the closest facility. Both the UMFWP and the CMFWP are nonlinear nonconvex optimization problems and difficult to solve. These problems are NP-hard problems even if all customers are located on a straight line [1]. However, if the optimal locations for the facilities are given, the optimal allocation can be found by solving a transportation problem for the CMFWP and assigning every customer to the nearest facility for the uncapacitated case. Similarly, if the optimal allocations are given, both problems can be separated into SFWP's for every facility and their allocations over customers and solved easily.

Table 1.1. Distance functions most frequently used in the literature

Distance	**Formula**
Euclidean	$d\left(\mathbf{x}_i, \mathbf{a}_j\right) = \left[\left(x_{i1} - a_{j1}\right)^2 + \left(x_{i2} - a_{j2}\right)^2\right]^{1/2}$
rectilinear	$d\left(\mathbf{x}_i, \mathbf{a}_j\right) = \lvert x_{i1} - a_{j1}\rvert + \lvert x_{i2} - a_{j2}\rvert$
squared Euclidean	$d\left(\mathbf{x}_i, \mathbf{a}_j\right) = \left(x_{i1} - a_{j1}\right)^2 + \left(x_{i2} - a_{j2}\right)^2$
l_p	$d\left(\mathbf{x}_i, \mathbf{a}_j\right) = \left[\lvert x_{i1} - a_{j1}\rvert^p + \lvert x_{i2} - a_{j2}\rvert^p\right]^{1/p}$
l_p^q	$d\left(\mathbf{x}_i, \mathbf{a}_j\right) = \left[\lvert x_{i1} - a_{j1}\rvert^p + \lvert x_{i2} - a_{j2}\rvert^p\right]^{q/p}$

MFWPs can be categorized according to their distance functions. The most common distance functions investigated by the researchers are the Euclidean, rectilinear and squared Euclidean distances. Although not specifically investigated in the literature, there are also some results for some of the l_p distance functions. The most frequently used distance functions can be seen in Table 1.1.

CMFWPs can be grouped into two classes: single-source and multi-source problems. In the single-source CMFWP every customer can only be supplied by a single facility whereas in the multi-source CMFWP, customers are allowed to have more than one supplier.

In this study, we are aiming to propose approaches to solve the l_p-distance multi-source CMFWP (L$_p$CMFWP) for $1 \leq p \leq 2$ approximately. In Chapter 2, we will give general problem formulations for both the continuous and the discrete versions for the capacitated and the uncapacitated MFWPs. This part will be followed by a general literature survey on the MFWP in Chapter 3. In Chapter 4, a general overview about the Lagrangean relaxation and subgradient optimization will be given. These two will be used in several different parts of the thesis. Chapter 5 will include some alternating location allocation type heuristics to solve the CMFWP, which gives upper bounds for our problem. In Chapter 6, we will give information about the discretization strategies for the CMFWP. The main motivation in this chapter is to solve CMFWP approximately by solving approximating discrete capacitated multifacility location allocation problem. In Chapter 7, a LR scheme will be formulated. In addition, we will define some lower bounding algorithms for the Lagrangean relaxation of the CMFWP.

Computational results will be given in Chapter 8 and the thesis will be concluded by making comments and giving directions for the future work in the last chapter.

2. PROBLEM FORMULATION

In CMFWP, the aim is to find the locations of m capacitated facilities and their allocation for each of n customers in a Euclidean space which minimizes the total transportation cost. Each facility has a predetermined supply capacity and their total allocation to customers cannot exceed this amount. The mathematical programming formulation of the CMFWP can be stated as follows:

CMFWP:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} w_{ij} d\left(\mathbf{x}_i, \mathbf{a}_j\right) \tag{2.1}$$

$$\text{s.t.} \sum_{i=1}^{m} w_{ij} = h_j \qquad j = 1, ..., n \tag{2.2}$$

$$\sum_{j=1}^{n} w_{ij} = s_i \qquad i = 1, ..., m \tag{2.3}$$

$$w_{ij} \geq 0 \qquad i = 1, ..., m; j = 1, ..., n. \tag{2.4}$$

In this model m is the number of facilities to be located and n is the number of customers. h_j is the demand and $\mathbf{a}_j = \begin{pmatrix} a_{j1} & a_{j2} \end{pmatrix}^T$ is the coordinates of customer j. s_i represents the capacity of facility i. $\mathbf{x}_i = \begin{pmatrix} x_{i1} & x_{i2} \end{pmatrix}^T$ and w_{ij} are respectively the unknown coordinates of facility i and the amount shipped from facility i to customer j with the unit shipment cost per unit amount per unit distance c_{ij}. The formulation assumes that the problem is balanced; i.e. total demand equals to total supply. If the total demand exceeds the total supply, the problem is infeasible. On the other hand, if total supply exceeds total demand, a dummy customer with the demand of the excess supply and zero shipment cost per distance can make the problem balanced.

The objective (2.1) equals the total transportation cost. Constraints (2.2) and Constraints (2.3) ensure demand satisfaction of each customer and supply limitation of each facility respectively.

The only difference between the CMFWP and its uncapacitated counterpart is the capacity limitations over facilities. In the UMFWP capacities of the facilities are unlimited. Because of this, the CMFWP formulation without the facility supply limitation constraints, namely Constraints (2.3), can be used for the UMFWP. However, since the customers are assigned to the nearest facility in the uncapacitated case, the problem can be formulated using binary decision variables as follows:

UMFWP:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} y_{ij} h_j d\left(\mathbf{x}_i, \mathbf{a}_j\right) \tag{2.5}$$

$$\text{s.t.} \sum_{i=1}^{m} y_{ij} = 1 \qquad j = 1, ..., n \tag{2.6}$$

$$y_{ij} \in \{0, 1\} \qquad i = 1, ..., m; j = 1, ..., n. \tag{2.7}$$

Different than the capacitated case, instead of w_{ij}, binary decision variable y_{ij} is used. It is set to 1 if customer j fulfills all its demand from facility i, and 0 otherwise. Constraints (2.6) ensure that each customer is assigned to a facility.

One of the other ways to formulate the UMFWP is to use an equivalent set partitioning problem (SPP) formulation (UMFWP$_2$). In this formulation, every possible subset of the customer set $\mathcal{N}$ are created which will be the sets in our SPP model. A SFWP model is solved for every subset to find the cost of assigning a single facility for the elements of this subset, or cost of selecting the given set for the SPP. The mathematical formulation of the UMFWP$_2$ is given by Krau [2] as follows:

UMFWP$_2$:

$$\min \sum_{s:\mathcal{S}_s \subset \mathcal{N}} c_s z_s \tag{2.8}$$

$$\text{s.t.} \sum_{s:\mathcal{S}_s \subset \mathcal{N}} b_{js} z_s = 1 \qquad j = 1, ..., n \tag{2.9}$$

$$\sum_{s:\mathcal{S}_s \subset \mathcal{N}} z_s = m \tag{2.10}$$

$$z_s \in \{0, 1\} \tag{2.11}$$

where,

$$b_{js} = \begin{cases} 1 \text{ if } j \in \mathcal{S}_s \\ 0 \text{ otherwise} \end{cases} \quad \text{and } c_s = \min_{x_s} \left\{ \sum_{j=1}^{n} b_{js} d\left(\mathbf{x}_s, \mathbf{a}_j\right) \right\} \tag{2.12}$$

and $d\left(\mathbf{x}_s, \mathbf{a}_j\right)$ is any defined distance function for $(\mathbf{x}_s, \mathbf{a}_j)$ pair.

In the above UMFWP$_2$ formulation, $\mathcal{S}_s$ is the subset s of the customer set $\mathcal{N}$, c_s is the cost associated with this subset, z_s is the binary decision variable, equals to 1 if subset $\mathcal{S}_s$ is selected or the customers in the subset $\mathcal{S}_s$ are supplied by the same facility, 0 otherwise. Constraints (2.9) guarantee that every customer is assigned to a facility. Constraint (2.10) ensures that the number of facilities which can be opened equals to m. Equations (2.12) shows that for every subset $\mathcal{S}_s$, a SFWP must be solved to determine c_s. Since the SFWP can be solved easily to optimality by using the Weiszfeld procedure [3], the related costs c_s can be calculated efficiently. However, the main problem for this formulation is the exponential number of variables related to the number of customers. There are $2^n - 1$ decision variables in this model.

In the discrete capacitated multifacility location allocation problem (DCMLAP), the facilities have limited capacities and serve customers by using this limited supply in the Euclidean space, as it is the case in CMFWP. Both the locations of the facilities and their allocations to customers which minimizes the total transportation cost are found. The only difference is the possible set of locations for the facilities.

In CMFWP, the facilities can be placed anywhere in the Euclidean space, whereas in the DCMLAP, the set of candidate locations are predetermined and facilities can only be placed on these sides. Besides, even though the CMFWP does not have a known linear model, its discrete version, discrete capacitated multifacility location allocation problem (DCMLAP), can be modeled and solved as a difficult mixed integer linear programming (MILP) problem.

When $\mathcal{K} = \{\mathbf{b}_1, ..., \mathbf{b}_k, ..., \mathbf{b}_r\}$ is the set of candidate locations for the facilities and $d(\mathbf{b}, \mathbf{a})$ is the distance between customer location $\mathbf{a}$ and candidate location $\mathbf{b}$, the mathematical model of the DCMLAP can be stated as:

DCMLAP:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{r} w_{ijk} c_{ij} d\left(\mathbf{b}_k, \mathbf{a}_j\right) \tag{2.13}$$

$$\text{s.t.} \sum_{j=1}^{n} w_{ijk} = s_i x_{ik} \qquad i = 1, ..., m; k = 1, ..., r \tag{2.14}$$

$$\sum_{i=1}^{m} \sum_{k=1}^{r} w_{ijk} = h_j \qquad j = 1, ..., n \tag{2.15}$$

$$\sum_{k=1}^{r} x_{ik} = 1 \qquad i = 1, ..., m \tag{2.16}$$

$$w_{ijk} \geq 0 \qquad i = 1, ..., m; j = 1, ..., n; k = 1, ..., r \tag{2.17}$$

$$x_{ik} \in \{0, 1\} \qquad i = 1, ..., m; k = 1, ..., r. \tag{2.18}$$

Different than its continuous counterpart CMFWP, for the DCMLAP model given above, binary decision variables x_{ik} are used to determine facility locations $\mathbf{x}_i$; it is set to 1 if facility i is located on candidate location k, or 0 otherwise. Constraints (2.14) ensure that only opened facilities can serve the customers. Constraints (2.15) guarantee that the demand of each customer is satisfied and Constraints (2.16) force each facility to be located only on one of the candidate locations.

It is important to note that DCMLAP does not guarantee the optimal solution for the CMFWP if candidate facility locations set does not include optimal facility sites. In other words, if optimal facility locations for the CMFWP are included in the set of candidate facility locations, DCMLAP will find an optimal solution for the CMFWP. We can also state that, as the size of the candidate location set goes to infinity, the optimal value of DCMLAP converges to the one of CMFWP.

3. LITERATURE SURVEY

SFWP in the Euclidean space was first proposed by Pierre de Fermat. He composed the problem with three equal demanding customers and it is solved in the same decade [4]. Then the problem was expanded in three dimensions: different demands for each customer, increased number of customers and increased number of facilities.

Cooper introduced the MFWP first [5]. He also proved that the objective function of the given problem is neither concave nor convex [6]. In addition to these, he tried to solve the problem by a converging Alternating Location Allocation (ALA) heuristic, which is still one of the widely used heuristics to solve the MFWP [7]. A similar heuristic for the capacitated version, Capacitated Alternating Location Allocation (CALA) heuristic, was also proposed by Cooper. The only difference from ALA is that, instead of assigning the customers to the closest facilities in the allocation phase, a transportation problem is solved [8]. In addition to heuristic methods, Cooper also defined an exact method, which enumerates all basic feasible solutions which is based on the fact that the optimal solution lies at an extreme point of the feasible region [8]. Since the number of extreme points can be very large, this method is applicable for only small sized problems.

Ostresh devised an exact method called TWAIN to solve the uncapacitated two center location-allocation problem in particular [9]. His method first partitions the customers into every possible two groups and solves a single facility location problem for every group. However, since the convex hulls of each facility-customer group cannot coincide at the optimality, he partitioned customers according to Thiessen polygons. This method decreases possible partition groups from Stirling number of the second kind, $S(n,2) = 2^{n-1} - 1$ to $C(n,2) = \frac{n(n-1)}{2}$ and less if collinearity exist in customer locations. However, the method is impractical for the UMFWP with more than two facilities.

Rosing [10] devised a method using Ostresh's approach, which works in two stages.

In the first stage, every possible convex hull and their costs are found. Costs are calculated by solving a Weber problem for each convex hull which also finds optimal facility locations for every convex hull. Then a set partitioning problem is solved which includes every (or some intelligently selected) convex hulls and their associated costs as columns.

Bongartz et al. [11] conducted a solution procedure which is generalized for l_p distance location allocation problems. Their algorithm finds stationary points and descent directions by using second-order information for the locations and first-order information for the allocations after relaxing the binary restrictions of the allocation variables.

Bischoff and Klamroth [12] proposed two branch and bound schemes for the multi connection location problem and UMFWP, one of which is branching on discrete assignment variables whereas the other is branching on continuous location variables. These are promising methods for both problems and neither of them dominates the other.

Chen at al. [13] models UMFWP as a difference of convex function (DC) programming problem and convert it into a concave minimization problem that is solved by outer approximation method in which number of dimensions increases as the number of facilities increases. As a result problems with 20 facilities or less are solved optimally.

Krau [2] reported an exact method that first models the UMFWP as a set covering problem and solves them by using column generation and DC programming subproblems. He also proposes a branch and price strategy to get binary optimal solution. As a branching strategy, he selects two customers and creates child nodes by using them. In the left child's feasible column set, the selected customer pair either exist together or does not exist at all, whereas in the right child's column set, at most one of the customers exists. UMFWP instances with at most 100 facilities are solved optimally by his algorithm and from the experimental results, he claims that, his method works better for instances with small facility-to-customer ratio .

Righini and Zaniboni [14] extends Krau's early work [2] and propose a method which first models the UMFWP as a set partitioning model and solves the problem again by column generation method. The difference between the work of Krau and that of Righini and Zaniboni is that, in the latter one, subproblems are solved by SFWP with limited distance instead of DC programming. They use two lower bounds to decrease the CPU seconds. Similar to Krau [2] they apply a branch and price technique which groups customers and create nodes accordingly. Last but not least, in order to eliminate degeneracy, they added two different stabilization methods namely box stabilization [15] and interior point method [16]. They solve problems optimally with at most 2000 customers and 1300 facilities. Their claim about the algorithm is that, it works better for instances with high facility-to-customer ratio.

Sherali and Tunçbilek transformed squared Euclidean distance CMFWP problem into convex quadratic function with transportation constraints [17]. They developed a branch and bound scheme which uses four upper bounding schemes one of which is derived by using reformulation linearization technique (RLT) [18]. RLT linearizes the original nonlinear problem unfortunately by increasing the number of variables enormously. Although RLT needs more computational effort, it results in tighter bounds for the original problem. This lower bounding algorithm is used together with branch and bound algorithm that implicitly enumerates the vertices of the feasible region. In their branching scheme, the allocation space is partitioned according to the values of the variables, either zero or positive.

Sherali et al. extend the same procedure for the Euclidean distance CMFWP [19]. They again used RLT to find a lower bound for the subproblems arising at each node. In addition to the Euclidean distance, they generalized the method for the l_p distance as well.

In addition to exact methods given above, there are some heuristics which approximates the CMFWP. Aras et al. devised heuristics for both the rectilinear [20] and the Euclidean distance [21] CMFWP's. In the rectilinear distance case, they proposed a new formulation which is more efficient than Sherali et al.'s [19] RLT based one. This is based on Wendell and Hurter's dominance results. They showed that Weber problem has an optimal solution located within the convex hull of the set of customers for every distance norm and located on the intersection point of vertical and horizontal lines drawn through customer locations for rectilinear norm [22], Hansen et al. generalized these two results for the MFWP [23]. Aras et al. exploited these results to formulate the MFWP with the rectilinear distance function as equivalent MILP problem [20] and developed an efficient discrete approximation heuristic (DAH) when the l_p distance is used instead [21]. In addition to these, it is worth noting that in both papers, Cooper's alternating location allocation heuristic [7] is used to improve the results at the final stage.

4. LAGRANGEAN RELAXATION AND SUBGRADIENT OPTIMIZATION

This chapter aims to give a general idea about the Lagrangean relaxation and the subgradient optimization. These two techniques are used together in several different parts of this thesis. Lagrangean relaxation is one of the best relaxation technique for the problems with complicated constraints. First we will describe this technique and give an example that is somewhat similar to the problems considered in this research. Then, we will give the general scheme for the subgradient optimization to solve the Lagrangean dual problem.

As stated in the previous sections, it is computationally difficult to solve CMFWP even for small instances. This is a nonlinear, nonconvex optimization problem. In order to find good solution for these types of problems, lower and upper bounds for the same problem can be calculated. However, accuracy and efficiency are the key issues in calculating the lower and upper bounds.

Upper bounds (UB) (for minimization problems) can be calculated by using heuristics. There are various types of heuristics reported in the literature. On the other hand, lower bounds (LB) can be found by relaxing the original problem and solving it. The most commonly used relaxation strategies are the linear programming relaxation and the Lagrangean relaxation.

Lagrangean relaxation (LR) has been successfully applied to many difficult optimization problems ever since Held and Karp's work on the traveling salesman problem [24]. It is used to find bounds on the optimal value of the MILP problems. In LR one or some of the constraints are added to the objective function by multiplying this function with a so called Lagrangean multiplier. Usually complicating constraints are selected to be relaxed. The main issue in LR is the decision of constant values multiplied (i.e. Lagrangean multipliers) by the constraints and the choice of constraints to be relaxed. Now let us show this technique on a generic model. Assume that our main problem (MP) is formulated as:

MP:

$$\begin{aligned} \min\ & \mathbf{cx} && (4.1)\\ \text{s.t.}\ & \mathbf{Ax} \geq \mathbf{b} && (4.2)\\ & \mathbf{Bx} \geq \mathbf{d} && (4.3)\\ & \mathbf{x} \geq 0. && (4.4) \end{aligned}$$

If we decide to relax Constraints (4.3) our Lagrangean subproblem (LSP) can be reformulated as,

LSP:

$$\begin{aligned} \min\ & \mathbf{cx} + \boldsymbol{\lambda}(\mathbf{d} - \mathbf{Bx}) && (4.5)\\ \text{s.t.}\ & \mathbf{Ax} \geq \mathbf{b} && (4.6)\\ & \mathbf{x} \geq 0 && (4.7)\\ & \boldsymbol{\lambda} \geq 0. && (4.8) \end{aligned}$$

As stated above, the objective function value of the LSP is a LB for the main problem, and our aim is to make it as tight as possible. In other words, we are aiming to maximize the LSP by choosing $\boldsymbol{\lambda}$ vector wisely. This new problem is named as the Lagrangean dual problem (LDP) and has the following form:

LDP:

$$\max_{\boldsymbol{\lambda} \geq 0} \left\{ \begin{aligned} \min\ & \mathbf{cx} + \boldsymbol{\lambda}(\mathbf{d} - \mathbf{Bx}) \\ \text{s.t.}\ & \mathbf{Ax} \geq \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned} \right\}. \qquad (4.9)$$

There are several ways for determining the values of these multipliers. One of them is to solve LDP using the subgradient optimization (SO). SO is an iterative procedure and it creates multipliers for the LRP in every step with given initial values. It attempts to maximize the lower bound obtained from the LR by calculating multipliers wisely. SO computes tight lower bounds efficiently for the most of the MILP problems and it is widely used in the literature as a result [25].

The generic subgradient algorithm for the minimization problem (4.1) - (4.4) is given above. Here $\boldsymbol{\lambda}$ and $\mathbf{G}$ are respectively the Lagrangean multiplier and subgradient vectors, C_{UB} and C_{LB} denote the recent bounds and Z_{UB} and Z_{LB} denote the current best upper and lower bounds. π is the step size parameter.

```
1. Decide initial $\boldsymbol{\lambda}$ and $\pi$ (where $0 < \pi \leq 2$).
2. Calculate $C_{\text{UB}}$ by using a problem specific heuristic.
3. Solve the LRP by using the current $\boldsymbol{\lambda}$ and find $C_{\text{LB}}$
4. $Z_{\text{UB}} \leftarrow \min\{Z_{\text{UB}}, C_{\text{UB}}\}$ and $Z_{LB} \leftarrow \max\{Z_{\text{LB}}, C_{\text{LB}}\}$
5. Calculate $\mathbf{G}$ as the infeasibility of the relaxed constraints.
6. Calculate step size $T$ by using $Z_{\text{UB}}$ and $C_{\text{LB}}$: $T = \frac{\pi(Z_{\text{UB}} - C_{\text{LB}})}{|\mathbf{G}|^2}$
7. Update $\boldsymbol{\lambda}$ using the step size calculated above: $\boldsymbol{\lambda} = \max\{0, \boldsymbol{\lambda} + T\mathbf{G}\}$
8. Update $\pi$ if necessary.
9. Go to Step 3 until termination criteria are satisfied.
```

Figure 4.1. General subgradient optimization algorithm

5. ALTERNATING LOCATION ALLOCATION HEURISTICS

Alternating location allocation heuristic [7] and its capacitated version [8] were first proposed by Cooper and they are still two powerful heuristics to solve the MFWP. One of their weak points is the dependency on the starting points of the heuristics. In this chapter, first the CALA heuristic will be defined and then some more heuristics built on CALA will be explained.

5.1. Capacitated Location Allocation Heuristic

As mentioned above, CALA heuristic was proposed by Cooper, who defined the CMFWP in 1971 [8]. The heuristic is simply composed of the sequential location and transportation problems (TPs). Even though the CMFWP is an NP-complete problem, the location and transportation problems are easy to solve.

In TP the aim is to minimize the total transportation cost by only deciding the allocations between facilities and the capacitated customers. Different than CMFWP, in TP, locations of the facilities are known. The mathematical model for the TP is as follows:

TP:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} w_{ij} d\left(\overline{\mathbf{x}}_i, \mathbf{a}_j\right) \tag{5.1}$$

$$\text{s.t.} \sum_{i=1}^{m} w_{ij} = h_j \qquad j = 1, ..., n \tag{5.2}$$

$$\sum_{j=1}^{n} w_{ij} = s_i \qquad i = 1, ..., m \tag{5.3}$$

$$w_{ij} \geq 0 \qquad i = 1, ..., m; j = 1, ..., n. \tag{5.4}$$

As given before in the CMFWP, facility locations are given as decision variables whereas in TP $\overline{\mathbf{x}}_i$'s are predetermined. Because of this difference, TP is an easy to solve linear programming problem. Similar to CMFWP, w_{ij} is the decision variable for the allocation amount sent from facility i to customer j. In addition to that, Constraints (5.2) ensure demand satisfaction of each customer and Constraints (5.3) limit the amount of goods that can be sent by the facilities.

The second part of the heuristic is composed of finding the allocations between the facilities and the customers by solving a SFWP for every facility. In the SFWP, the aim is to locate a single facility in a Euclidean space to minimize total transportation cost. Allocations are predetermined and given as parameters to the problem. Different than its multifacility counterpart, the SFWP can be solved easily by the Weiszfeld procedure [3] or its generalizations [26]. Weiszfeld procedure is an iterative algorithm which has fast asymptotic convergence. For $\overline{w}_{ij}$ is the predetermined allocations from facility i to customer j, the overall Weiszfeld procedure can be formally stated as in Figure 5.1.

```
1. $t \leftarrow 0$
2. Initialize the facility location by setting $x_{i1}^0 = \frac{\sum_{j=1}^{n} \overline{w}_{ij} a_{i1}}{\sum_{j=1}^{n} \overline{w}_{ij}}$ and $x_{i2}^0 = \frac{\sum_{j=1}^{n} \overline{w}_{ij} a_{i2}}{\sum_{j=1}^{n} \overline{w}_{ij}}$
3. Calculate the distance $d_{ij}^{(t)} = d\left(\mathbf{x}_i^{(t)}, \mathbf{a}_j\right)$
4. Update the location of facility $i$ as $x_{i1}^{(t+1)} = \frac{\sum_{j=1}^{n} \frac{\overline{w}_{ij} x_{i1}^{(t)}}{d_{ij}^{(t)}}}{\sum_{j=1}^{n} \frac{\overline{w}_{ij}}{d_{ij}^{(t)}}}$ and $x_{i2}^{(t+1)} = \frac{\sum_{j=1}^{n} \frac{\overline{w}_{ij} x_{i2}^{(t)}}{d_{ij}^{(t)}}}{\sum_{j=1}^{n} \frac{\overline{w}_{ij}}{d_{ij}^{(t)}}}$
5. $t \leftarrow t + 1$
6. Go to Step 3 until the termination criteria are satisfied
```

Figure 5.1. Weiszfeld procedure

Now we can define the CALA heuristic. CALA heuristic is initialized at given initial facility locations. This locations are used to calculate distances that are the main ingredients of the costs of TP. Then the TP is solved. The optimal allocations found from the TP are treated as given during the location phase of the problem. A SFWP is solved with them by using the Weiszfeld Procedure for every facility. Then, the location and allocation phases are repeated alternately until the stopping criteria are not satisfied. The overall CALA heuristic procedure is given in Figure 5.2.

```
1. Define initial facility locations $x_i$ for $i = 1, ..., m$
2. Set facility locations $\overline{\mathbf{x}}_i$ for $i = 1, ..., m$ as parameters and solve the TP to find
   the allocations $w_{ij}$ for $i = 1, ..., m; j = 1, ..., n$
3. Set allocations $\overline{w}_{ij}$ for $i = 1, ..., m; j = 1, ..., n$ as parameters and solve a SFWP
   for every facility to find the locations $\mathbf{x}_i$ for $i = 1, ..., m$
4. Go to Step 2 until termination criteria are not satisfied
```

Figure 5.2. Capacitated alternating location allocation heuristic

It is worth to note here that, performance of CALA heuristic heavily relies on the initial facility locations. For this reason we define some initialization methods.

5.2. Region Rejection Heuristic

In the region rejection heuristic (RRH), which is due to Luis et al. [27], the aim is to select initial locations of the facilities well separated. This is accomplished by checking whether or not there is an already placed facility that remains within the circle of a given radius centered at the newly initialized one. If this is the case, a new location is randomly selected for that facility. Initialization is completed when all the facilities are located within the convex hull of the customers, according to this procedure. In other words, while the initial facility locations are assigned, if the randomly selected location is not away from one of the previously placed facilities by a certain threshold distance, the selected location is renewed or the radius is decreased by some amount. Then the usual ALA steps are implemented. For the predetermined radius R, radius decreasing factor α and the iteration limit P, the general scheme of RRH can be defined as in Figure 5.3.

```
1. Initialize $R$, $P$ and find convex hull $\mathcal{H}$ of the customer locations
2. Let the facilities be in arbitrary order and set $i \leftarrow 1$
3. If $i = m + 1$ then go to Step 7, else $t \leftarrow 1$
4. Repeat
   (a) Select a point $\tilde{\mathbf{x}}_i \in \mathcal{H}$ randomly
   (b) $t \leftarrow t + 1$
   (c) If $t > P$ then $R \leftarrow \alpha R$ and $t \leftarrow 1$
   (d) Until $\min_{\bar{i}<i} \{d(\mathbf{x}_{\bar{i}}, \tilde{\mathbf{x}}_i)\} \geq R$
5. $\mathbf{x}_i \leftarrow \tilde{\mathbf{x}}_i$
6. $i \leftarrow i + 1$ and go to Step 2
7. Run CALA heuristic starting with these initial facility locations
```

Figure 5.3. Region rejection heuristic

5.3. Discrete Region Rejection Heuristic

Discrete region rejection heuristic (DRRH) is developed as an enhancement of the region rejection heuristic. In DRRH, instead of selecting a random initialized location within the convex hull of the customers, a random customer is selected from the set of customers and its coordinates are used as an initial location. The motivation behind this is that the optimal facility locations usually overlaps with the customer locations as observed for the UMFWP by Hansen et al. [23] and by Aras et al. [21] for the CMFWP. The overall procedure for the DDRH is provided in Figure 5.4.

5.4. Region Rejection Heuristic with Dynamic Radius

In RRH, the size of the region is set to a fixed value, all restricted regions have the same radius. However, this selection can be improper since some circles can have more customer demands. For this reason RRH heuristic was improved with the dynamic radius concept [27]. The radius of the facility is found by using an iterative procedure. First, a value Q, which is the total demand of the customers remaining within the circle centered of facility i with dynamic radius R_i are determined. Second, R_i is updated

```
1. Initialize $R$ and $P$
2. Let the facilities be in arbitrary order and set $i \leftarrow 1$
3. If $i = m + 1$ then go to Step 7, else $t \leftarrow 1$
4. Repeat
    (a) Select a random customer, say customer $j$ and set $\tilde{\mathbf{x}}_i \leftarrow \mathbf{a}_j$
    (b) $t \leftarrow t + 1$
    (c) If $t > P$ then $R \leftarrow \alpha R$ and $t \leftarrow 1$
    (d) Until $\min_{\bar{i}<i} \{d(\mathbf{x}_{\bar{i}}, \tilde{\mathbf{x}}_i)\} \geq R$
5. $\mathbf{x}_i \leftarrow \tilde{\mathbf{x}}_i$
6. $i \leftarrow i + 1$ and go to Step 2
7. Run CALA heuristic starting with these initial facility locations
```

Figure 5.4. Discrete region rejection heuristic

to satisfy the inequalities $\theta h_i \leq Q \leq h_i$, and the circle is created with center x_i and radius R_i. All the other parts of the heuristic is the same as the simple RRH heuristic. In addition to parameters defined in RRH, for R_i^e is the radius of facility i at iteration e, $Q(\mathbf{x}_i, R_i)$ is the total demand of customers inside the circle with center $\mathbf{x}_i$ and radius R_i, and E is the iteration limit in radius adjustment, the overall procedure of the region rejection heuristic with dynamic radius (RRH$'$) can be seen in Figure 5.5.

5.5. Discrete Region Rejection Heuristic with Dynamic Radius

Similar to the modification done between RRH and RRH$'$, discrete region rejection heuristic with dynamic radius (DRRH$'$) includes notion of capacity of the facility and demands of the customers around it. In addition to selection of random locations over the customers, a dynamic radius is assigned for every located facilities. This value is the radius of the customers whose demands' summation is close to the capacity of the facility. This radius is either found by multiplying it with a calculated parameter or by bisection method. For the same parameters given in RRH$'$, the overall procedure for the DRRH$'$ is listed as Figure 5.6.

```
1. Initialize $E$, $R$, $P$ and find convex hull $\mathcal{H}$ of the customer locations
2. Let the facilities be in arbitrary order and set $i \leftarrow 1$
3. If $i = m + 1$ then go to Step 14, else $t \leftarrow 1$
4. Repeat (location selection)
    (a) Select a point $\tilde{\mathbf{x}}_i \in \mathcal{H}$ randomly
    (b) If $t > P$ then go to Step 13
    (c) $t \leftarrow t + 1$
    (d) Until $\min_{\bar{i}<i} \{d(\mathbf{x}_{\bar{i}}, \tilde{\mathbf{x}}_i)\} - r_{\bar{i}} \leq 0$
5. $\mathbf{x}_i \leftarrow \tilde{\mathbf{x}}_i$
6. Initialize $\tilde{R}^{(0)} = R$ and set $e \leftarrow 1$
7. Repeat (radius adjustment)
    (a) $\beta \leftarrow \sqrt{\frac{b}{Q(\mathbf{x}_i, \tilde{R}^{(e)})}}$ and $\tilde{R}^{(e)} \leftarrow \tilde{R}^{(e-1)} * \beta$
    (b) $e \leftarrow e + 1$
    (c) Until $\theta h_i \leq Q\left(\mathbf{x}_i, \tilde{R}^{(e)}\right) \leq h_i$ or $e > E$
8. If $\theta h_i \leq Q\left(\mathbf{x}_i, \tilde{R}^{(e)}\right) \leq h_i$ then $R_i \leftarrow \tilde{R}^{(e)}$ and go to Step 13 else go to Step 10
9. If $Q\left(\mathbf{x}_i, \tilde{R}^{(e)}\right) > h_i$ then $R_l \leftarrow \tilde{R}^{(e)}$ and $R_u \leftarrow \tilde{R}^{(e-1)}$ else $R_l \leftarrow \tilde{R}^{(e-1)}$ and
   $R_u \leftarrow \tilde{R}^{(e)}$
10. Repeat (bisection)
    (a) $R_m = \frac{R_l + R_u}{2}$
    (b) If $Q(\mathbf{x}_i, R_m) > h_i$ then $R_u \leftarrow R_m$ else $R_l \leftarrow R_m$
    (c) Until $\theta h_i \leq Q(\mathbf{x}_i, R_m) \leq h_i$
11. $R_i \leftarrow R_m$
12. $i \leftarrow i + 1$ and go to Step 2
13. Locate all of the unassigned facilities to random locations
14. Run CALA heuristic starting with these initial facility locations
```

Figure 5.5. Region rejection heuristic with dynamic radius

```
1. Initialize $E$, $R$ and $P$
2. Let the facilities be in arbitrary order and set $i \leftarrow 1$
3. If $i = m + 1$ then go to Step 14, else $t \leftarrow 1$
4. Repeat (location selection)
   (a) Select a random customer, say customer $j$ and set $\tilde{\mathbf{x}}_i \leftarrow \mathbf{a}_j$
   (b) If $t > P$ then go to Step 13
   (c) $t \leftarrow t + 1$
   (d) Until $\min_{\bar{i}<i} \{d(\mathbf{x}_{\bar{i}}, \tilde{\mathbf{x}}_i)\} - r_{\bar{i}} \leq 0$
5. $\mathbf{x}_i \leftarrow \tilde{\mathbf{x}}_i$
6. Initialize $\tilde{R}^{(0)} = R$ and set $e \leftarrow 1$
7. Repeat (radius adjustment)
   (a) $\beta \leftarrow \sqrt{\frac{b}{Q\left(\mathbf{x}_i, \tilde{R}^{(e)}\right)}}$ and $\tilde{R}^{(e)} \leftarrow \tilde{R}^{(e-1)} * \beta$
   (b) $e \leftarrow e + 1$
   (c) Until $\theta h_i \leq Q\left(\mathbf{x}_i, \tilde{R}^{(e)}\right) \leq h_i$ or $e > E$
8. If $\theta h_i \leq Q\left(\mathbf{x}_i, \tilde{R}^{(e)}\right) \leq h_i$ then $R_i \leftarrow \tilde{R}^{(e)}$ and go to Step 13 else go to Step 10
9. If $Q\left(\mathbf{x}_i, \tilde{R}^{(e)}\right) > h_i$ then $R_l \leftarrow \tilde{R}^{(e)}$ and $R_u \leftarrow \tilde{R}^{(e-1)}$ else $R_l \leftarrow \tilde{R}^{(e-1)}$ and
   $R_u \leftarrow \tilde{R}^{(e)}$
10. Repeat (bisection)
   (a) $R_m \leftarrow \frac{R_l + R_u}{2}$
   (b) If $Q(\mathbf{x}_i, R_m) > h_i$ then $R_u \leftarrow R_m$ else $R_l \leftarrow R_m$
   (c) Until $\theta h_i \leq Q(\mathbf{x}_i, R_m) \leq h_i$
11. $R_i \leftarrow R_m$
12. $i \leftarrow i + 1$ and go to Step 2
13. Locate all of the unassigned facilities to random customer locations
14. Run CALA heuristic starting with these initial facility locations
```

Figure 5.6. Discrete region rejection heuristic with dynamic radius

6. DISCRETIZATION STRATEGIES

As mentioned in Chapter 2, even though the CMFWP is a nonlinear programming problem, its discrete approximation DCMLAP can be formulated as the MILP problem given with Equations (2.13) - (2.18). In this chapter, our aim is to define solution approaches based on the discrete approximation DCMLAP. We will first define the discrete approximation heuristic which is simply finding the initial locations by solving a DCMLAP formulation and improving it by a single CALA run. Then the Lagrangean relaxation for the discrete approximation heuristic will be introduced. It is especially meaningful when the number of binary decision variables in the DCMLAP is large. These two strategies can be used to compute upper bounds on the optimal value of the CMFWP. In the last section of this chapter, we will propose two discrete approximation strategies. They are based on the relation of l_1, l_∞ and l_p norms and result two approximating MILP's whose optimal values give lower bounds for the CMFWP.

6.1. Discrete Approximation Heuristic

As mentioned in the previous chapter, CALA depends very much on the initial locations of the facilities. As a result, a DCMLAP model is solved first to improve CALA heuristic solution. This overall heuristic, which is named as the Discrete Approximation Heuristic (DAH), is due to Aras et al. [21]. In DAH, the candidate facility location set is formed by the customer locations, $\mathcal{K} = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$, and a DCMLAP is solved to find initial facility locations before running a CALA heuristic. Recall that DCMLAP is formulated in Chapter 2 using the Equations (2.13) - (2.18). A formal definition of DAH is given in Figure 6.1.

```
1. Set candidate location set $\mathcal{K} \leftarrow \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$
2. Solve the DCMLAP problem defined on $\mathcal{K}$
3. Run CALA after initializing at the locations obtained by solving DCMLAP.
```

Figure 6.1. Discrete approximation heuristic

6.2. Relaxed Discrete Approximation Heuristic

Depending on the size of the candidate point set $\mathcal{K}$, DCMLAP can be very large and very difficult to solve exactly. However, Lagrangean relaxation and subgradient optimization can be used to compute a good solution, which can be made even better if CALA is also incorporated. As stated before, the relaxed constraints in LR is an important issue and we are aiming both tight bounds and easier solution process by this relaxation. All possible relaxations are formulated but finally relaxing the demand constraints (2.15) is seen as the best strategy because of two reasons. First, choosing the demand constraints makes the Lagrangean subproblems separable over facilities. Second, computational experiments show that this relaxation generates tight lower bounds. The Lagrangean subproblem obtained by relaxing the demand constraints can be given as:

RDCMLAP:

$$\min \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{r} w_{ijk}\left[c_{ij}d\left(\mathbf{b}_k,\mathbf{a}_j\right)+\mu_j\right]-\sum_{j=1}^{n}\mu_j h_j \tag{6.1}$$

$$\text{s.t. Constraints (2.14), (2.16) - (2.18)} \tag{6.2}$$

$$w_{ijk} \le h_j \qquad i=1,...,m; j=1,...,n; k=1,...,r. \tag{6.3}$$

In the RDCMLAP, μ_j represents the Lagrangean multipliers. All the constraints, except the relaxed (2.15), of the DCMLAP are added to the new model. In addition, Constraints (6.3) which limit the demand satisfied are added to the model. In fact, these newly added constraints are redundant in the main model but when the demand constraints (2.15) are relaxed they increase the optimal value of RDCMLAP, which improves the lower bound.

As can be observed, the second summation of the objective function (6.1) is constant and does not affect the optimal solution. As a result, RDCMLAP can be decomposed into m problems RDCMLAP^i each for one facility i.

RDCMLAP^i:

$$\min \sum_{j=1}^{n} \sum_{k=1}^{r} w_{ijk} \left[c_{ij} d\left(\mathbf{b}_k, \mathbf{a}_j\right) + \mu_j \right] \tag{6.4}$$

$$\text{s.t.} \sum_{j=1}^{n} w_{ijk} = s_i x_{ik} \qquad k = 1, ..., r \tag{6.5}$$

$$\sum_{k=1}^{r} x_{ik} = 1 \tag{6.6}$$

$$w_{ijk} \geq 0 \qquad j = 1, ..., n; k = 1, ..., r \tag{6.7}$$

$$x_{ik} \in \{0, 1\} \qquad k = 1, ..., r \tag{6.8}$$

$$w_{ijk} \leq h_j \qquad j = 1, ..., n; k = 1, ..., r \tag{6.9}$$

The above RDCMLAP^i model can be solved easily by inspection. First, customers are sorted in nondecreasing order with respect to the cost of sending one unit of good from facility i to customer j, when the facility is located at candidate location k. Second, all the capacity of the facility is distributed to the customers by using the sorted list and the cost of assigning the facility to the given candidate location is found. This sorting and cost calculation processes are done for every candidate location. The candidate location with the minimum reduced cost is the optimal solution for facility i. The overall inspection algorithm for the RDCMLAP^i is given in Figure 6.2.

For the upper bounding algorithm the discrete CALA (DCALA) heuristic is used. DCALA is the discrete version of CALA and the location phase is accomplished solving 1-median problems instead of SFWP using Weiszfeld procedure. DCALA heuristic is stated as algorithm given in Figure 6.3.

After defining all the required algorithms, the SO algorithm for the RDAH can be formally listed as Figure 6.4.

```
1. Calculate $\tilde{c}_{ijk} \leftarrow c_{ij} d(\mathbf{b}_k, \mathbf{a}_j) + \mu_j$ for $j = 1, ..., n; k = 1, ..., r$
2. For each candidate location $k$
   (a) Sort the customers in nondecreasing order with respect to $\tilde{c}_{ijk}$. Let the
       customers in this order be given as $j(1), j(2), ..., j(n)$
   (b) $\tilde{h}_i \leftarrow h_i$, current objective $C \leftarrow 0$ and $l \leftarrow 1$
   (c) Repeat
         i. If $d_{j(l)} < \tilde{h}_i$ then $C \leftarrow C + \tilde{c}_{ijk} d_{j(l)}$ else $C \leftarrow C + \tilde{c}_{ijk} \tilde{h}_i$
        ii. $\tilde{h}_i \leftarrow \max \left\{ \tilde{h}_i - d_{j(l)}, 0 \right\}$
       iii. Until $\tilde{h}_i = 0$
   (d) If $C_i^* > C$ then $C_i^* \leftarrow C$ and $k_i^* \leftarrow k$
3. $k_i^*$ is the optimal candidate location and $C_i^*$ is the optimal objective value for the
   subproblem of facility $i$
```

Figure 6.2. Inspection method for RDCMLAP^i

```
1. Define initial facility locations from the candidate location set $\mathcal{K}$
2. Set facility locations $\overline{x}_i$ for $i = 1, ..., m$ as parameters and solve the TP to find
   the allocations $w_{ij}$ for $i = 1, ..., m; j = 1, ..., n$
3. Set allocations $\overline{w}_{ij}$ for $i = 1, ..., m; j = 1, ..., n$ as parameters
4. For each facility $i$ do
   (a) $k_i^* \leftarrow \arg\min_{\mathbf{b}_k \in \mathcal{K}} \left\{ \sum_j c_{ij} \overline{w}_{ij} d(\mathbf{b}_k, \mathbf{a}_j) \right\}$
   (b) $x_i \leftarrow b_{k_i^*}$
5. Go to step 2 until termination criteria are satisfied
```

Figure 6.3. Discrete capacitated alternating location allocation heuristic

```
1. Decide $\mu_j$, $\pi$ (where $0 \leq \pi \leq 2$) and set $Z_{\mathrm{UB}} \leftarrow \infty, Z_{\mathrm{LB}} \leftarrow 0$
2. Run DCALA algorithm, $C_{\mathrm{UB}}$ is the objective
3. $Z_{\mathrm{UB}} \leftarrow \min(Z_{\mathrm{UB}}, C_{\mathrm{UB}})$
4. Calculate the objective by solving CALA. Keep the solution if it is the best so far
5. For each facility $i$ do, find the optimal candidate location index $k_i^*$, by solving subproblems RCMLAP$^i$
6. Calculate $C_{\mathrm{LB}}$ and set $Z_{\mathrm{LB}} \leftarrow \max(Z_{\mathrm{LB}}, C_{\mathrm{LB}})$
7. $G_j \leftarrow \sum_{i=m}^{n} w_{ijk_i^*} - d_j$          $j = 1, ..., n$
8. $T \leftarrow \frac{\pi(Z_{\mathrm{UB}} - C_{\mathrm{LB}})}{\sum_{j=n}^{m} G_j^2}$
9. $\mu_j \leftarrow \mu_j + TG_j$          $j = 1, ..., n$
10. Update $\pi$ if needed
11. Go to Step 2 until termination criteria are satisfied
```

Figure 6.4. Relaxed discrete approximation heuristic

6.3. Discrete Approximations Using l_1 and l_∞ Norms

l_1 (rectilinear) and l_∞ (Tchebycheff) norms are two common distance functions, which have two important properties for the MFWP. The first property is that, the locations for the optimal solutions are an element of a finite set. This finite set contains the intersection points of the vertical and horizontal lines drawn through customer locations if it is l_1 norm and the intersection points of the 45° projection of the vertical and horizontal lines drawn through customer locations if it is l_∞ norm [28]. In addition to that, Wendell and Hurter showed that facility locations lie inside the convex hull of the customers [22]. As a result, since we have a set of candidate locations which contains the optimal locations for the CMFWP, optimal solution for it can be found by solving a DCMLAP problem with the appropriate candidate location set. The mathematical model for the DCMLAP is given by Equations (2.13) - (2.18).

Second, these two distance norms can be used as lower bounds for the other l_p norms [19]. More specifically if we let Z_p^* denotes the optimal value of any given LAP in l_p distance with $1 \leq p \leq \infty$,

$$Z_p^* \geq Z_\infty^* \tag{6.10}$$

$$2^{(p-1)/p} Z_p^* \geq Z_1^* \tag{6.11}$$

By using the inequalities, lower bound for the CMFWP with l_p distance can be computed solving MILP's based on the two properties mentioned in the previous paragraph. Last but not least, since the LR of DCMLAP produces a lower bound for the optimal value of the original model, RDCMLAP can also be used as a lower bound generated for the CMFWP. Then the overall procedure given in Figure 6.4 can be used without Step 4, namely the step where CALA heuristic is run for this purpose.

7. LAGRANGEAN HEURISTICS

As stated before, both the capacitated and the uncapacitated MFWP are NP-hard problems [1]. However, we know that, UMFWP has an equivalent pure binary linear programming set partitioning model with exponential number of decision variables. Even though it is a difficult problem with an exponential number of decision variables, relaxing some of the constraints and turning the CMFWP into an uncapacitated problem can be a good idea. In this section, we will relax the CMFWP by Lagrangean relaxation and solve it using different methods.

It is important to note that, we need either an optimal solution or a lower bound on the optimal value of the relaxed CMFWP. For this reason a set partitioning model of this relaxed problem is created, which is similar but more complicated than the usual UMFWP$_2$. This set partitioning model turns into an equivalent set covering model and a column generation scheme is created to solve the linear relaxation of this model optimally. In the next section, this column generation algorithm is combined with a branch and price scheme, which finds the (binary) optimal solution of the set covering problem, which gives a tighter lower bound. Finally, three more algorithms will be defined which are not valid lower bounds but can be used to make valid lower bounding algorithms faster.

7.1. Lagrangean Relaxation and Capacitated Multifacility Weber Problem

As mentioned above, in this part our aim is to relax the CMFWP model given with Equations (2.1) - (2.4) and deal with a problem which can be modeled as a set partitioning problem. In order to do this, we relax the capacity constraints (2.3) of the facilities to obtain the following LR of the CMFWP (RCMFWP) is obtained:

RCMFWP:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \left[c_{ij} d\left(\mathbf{x}_i, \mathbf{a}_j\right) + \lambda_i \right] - \sum_{i=1}^{m} \lambda_i s_i \tag{7.1}$$

$$\text{s.t.} \sum_{i=1}^{m} w_{ij} = h_j \qquad j = 1, ..., n \tag{7.2}$$

$$w_{ij} \geq 0 \qquad i = 1, ..., m; j = 1, ..., n. \tag{7.3}$$

In this formulation λ_i stands for the weights of the relaxed constraints and must be chosen wisely to have better lower bounds so these values are updated by using subgradient optimization (SO). In SO, an iterative procedure is used. Initial values of the multipliers ($\boldsymbol{\lambda}$) are set to a certain value and updated according to the subgradients ($\mathbf{G}$) and the step length T. The subgradient vector and the step length are calculated by using upper and lower bounding algorithms. Even though the upper bound can be found by using any heuristic solving the CMFWP, calculating a tight lower bound is not as easy as it seems. Lower bounds used for the CMFWP can either be the optimal value of the RCMFWP or a lower bound of it. The overall SO algorithm for solving the CMFWP is given in Figure 7.1.

```
1. Decide $\lambda_i$, $\pi$ (where $0 \leq \pi \leq 2$) and set $Z_{\text{UB}} \leftarrow \infty$
2. Run selected UB method and set $C_{\text{UB}}$ as the objective value
3. $Z_{\text{UB}} \leftarrow \min\left(Z_{\text{UB}}, C_{\text{UB}}\right)$
4. Run selected LB method and set $C_{\text{LB}}$ as the objective value
5. $Z_{\text{LB}} \leftarrow \max\left(Z_{\text{LB}}, C_{\text{LB}}\right)$
6. $G_i \leftarrow s_i - \sum_{j=1}^{n} y_{ij} h_j$        $i = 1, ..., m$
7. $T \leftarrow \frac{\pi(Z_{\text{UB}} - C_{\text{LB}})}{\sum_{i=1}^{m} G_i^2}$
8. $\lambda_i \leftarrow \max\left(0, \lambda_i + TG_i\right)$        $i = 1, ..., m$
9. Update $\pi$ if needed
10. Go to Step 2 until termination criteria are satisfied
```

Figure 7.1. Subgradient optimization procedure for the CMFWP

Before defining some lower bounding algorithms to use in the SO steps given before, let us rewrite our RCMFWP model. As it can be seen from the above model, the objective function (7.1) has a constant part which is the weighted sum of the facility capacities. For $\overline{d}_i(\mathbf{x}_i, \mathbf{a}_j) = c_{ij} d(\mathbf{x}_i, \mathbf{a}_j) + \lambda_i$ the objective function (7.1) of the relaxed model without the constant part can be rewritten as

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \overline{d}_i(\mathbf{x}_i, \mathbf{a}_j). \tag{7.4}$$

Before continuing, please note that for $d(\mathbf{x}_i, \mathbf{a}_j)$ is any defined distance function, c_{ij} and λ_i are nonnegative constants, $\overline{d}_i(\mathbf{x}_i, \mathbf{a}_j)$ always has nonnegative values. Since $\overline{d}_i(\mathbf{x}_i, \mathbf{a}_j)$ is always nonnegative and there is no capacity restrictions over facilities, the customers satisfy their demands from the one and only one facility, probably the one with the minimum function value of $\overline{d}_i(\mathbf{x}_i, \mathbf{a}_j)$. In other words, RCMFWP can be modeled using binary decision variables instead of the continuous w_{ij}. For y_{ij} is the binary decision variable which determines whether customer j is supplied by facility i or not, RCMFWP can be reformulated as

RCMFWP$_2$:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} y_{ij} h_j \overline{d}(\mathbf{x}_i, \mathbf{a}_j) - \sum_{i=1}^{m} \lambda_i s_i \tag{7.5}$$

$$\text{s.t.} \sum_{i=1}^{m} y_{ij} = 1 \qquad j = 1, ..., n \tag{7.6}$$

$$y_{ij} \in \{0, 1\} \qquad i = 1, ..., m; j = 1, ..., n. \tag{7.7}$$

The formulation given above is an extension of the UMFWP model given by equations (2.5) - (2.7) if the constant part is disregarded. The only difference between the ordinary UMFWP and the problem given with the above formulation is that, in RCMFWP$_2$ the distance function is facility specific and has a different value for every facility. In the remaining part of the chapter, we will define some algorithms to solve

RCMFWP$_2$ which will be used to compute lower bounds in procedure given in Figure 7.1. Please note that, for the upper bounding algorithms, any heuristic giving a feasible solution for the CMFWP can be used. In other words any strategies defined in Chapter 5 and Chapter 6 are suitable as an upper bounding procedure.

7.2. Set Partitioning

The ordinary UMFWP can be modeled as a set partitioning problem (2.8) - (2.12) by creating all possible subsets of customer set and solving a Weber problem for each subset to find their objective coefficients. However, in our case every facility can have different distance function because of the facility specific unit shipment cost c_{ij} per unit good per unit distance and multipliers λ_i. In order to solve our case, we have to create all possible subsets of the customer set for every facility and solve them. For $\overline{d}_i(\mathbf{x}_i, \mathbf{a}_j)$ is any defined distance function for $(\mathbf{x}_i, \mathbf{a}_j)$ pair for facility i and,

$$b_{js} = \begin{cases} 1 \text{ if } j \in \mathcal{S}_s \\ 0 \text{ otherwise} \end{cases} \quad \text{and } c_{is} = \min_{\mathbf{x}_i} \left\{ \sum_{j=1}^{n} b_{js} \overline{d}(\mathbf{x}_i, \mathbf{a}_j) \right\} \tag{7.8}$$

the set partitioning formulation for RCMFWP$_2$ can be given as

RCMFWP$_3$:

$$\min \sum_{i=1}^{m} \sum_{s:\mathcal{S}_s \subset \mathcal{N}} c_{is} z_{is} \tag{7.9}$$

$$\text{s.t.} \sum_{i=1}^{m} \sum_{s:\mathcal{S}_s \subset \mathcal{N}} b_{js} z_{is} = 1 \qquad j = 1, ..., n \tag{7.10}$$

$$\sum_{s:\mathcal{S}_s \subset \mathcal{N}} z_{is} = 1 \qquad i = 1, .., m \tag{7.11}$$

$$\sum_{i=1}^{m} \sum_{\mathcal{S}_s \subset \mathcal{N}} z_{is} = m \tag{7.12}$$

$$z_{is} \in \{0, 1\} \qquad i = 1, ..., m; s : \mathcal{S}_s \subset \mathcal{N}. \tag{7.13}$$

For every $(\mathbf{x}_i, \mathbf{a}_j)$ pair, since $\overline{d}_i(\mathbf{x}_i, \mathbf{a}_j) \geq 0$, every objective coefficient of subset s of facility i c_{is} is nonnegative. Because of that customers are not assigned to more than one facility in an optimal solution even though it is restricted with greater than or equal to instead of equal to in Constraints (7.10). Second, relaxing Constraints (7.11) with less than or equal to would not change the optimal value either, since the empty subsets are also created for the model and assigning an empty set to a facility is equal to not opening the facility. The former and latter facts ensure that relaxing the constraint sets (7.10) and (7.11) with greater than or equal to and less than or equal to respectively, would not make any difference at the optimality and the above model can be formulated with the new constraints which makes the model equivalent to a set covering problem (SCP). The RCMFWP can be modeled by using the SCP formulation RCMFWP$_4$ as follows:

RCMFWP$_4$:

$$\min \sum_{i=1}^{m} \sum_{s:\mathcal{S}_s \subset \mathcal{N}} c_{is} z_{is} \tag{7.14}$$

$$\text{s.t.} \sum_{i=1}^{m} \sum_{s:\mathcal{S}_s \subset \mathcal{N}} b_{js} z_{is} \geq 1 \qquad j = 1, ..., n \tag{7.15}$$

$$\sum_{s:\mathcal{S}_s \subset \mathcal{N}} z_{is} \leq 1 \qquad i = 1, .., m \tag{7.16}$$

$$\sum_{i=1}^{m} \sum_{s:\mathcal{S}_s \subset \mathcal{N}} z_{is} \leq m \tag{7.17}$$

$$z_{is} \in \{0, 1\} \qquad i = 1, .., m; s : S_s \subset \mathcal{N}. \tag{7.18}$$

Since the objective of this model is used as a lower bound in the main problem, the linear relaxation of the set covering version of RCMFWP is also a valid lower bound for the main problem. The linear relaxation of the RCMFWP$_4$ is obtained by replacing Constraints (7.18) with the nonnegativity restrictions

$$z_{is} \geq 0 \qquad i = 1, .., m; s : \mathcal{S}_s \subset \mathcal{N}. \tag{7.19}$$

Even though the relaxed model given above does not contain any binary variables, even the medium sized problems cannot be solved easily, because of the very large number of decision variables and SFWPs that must be solved: We need to solve $2^n m$ SFWP to create the set covering model with the same number of decision variables. In order to deal with this problem, we use the column generation approach explained in the next section.

7.3. Column Generation

One of the most successful approaches to solve large integer programming problems optimally is to adopt the column generation developed by Dantzig and Wolfe in 1960 [29] for solving large liner programming problems. The main idea in column generation is choosing entering variable to the basis by generating and solving a specific subproblem in every step to calculate and find the nonbasic variable which has the minimum reduced cost instead of introducing every possible columns to the model from the beginning and solving it.

The set covering problem mentioned above can also be modeled and solved by the column generation approach. It should be recalled that, the column generation approach for the UMFWP was originally proposed and used by Krau [2]. We adopt this method for our case and our derivations are heavily drawn on his work.

Assume that not all of the columns, but a set of them that can give a feasible solution in RCMFWP$_4$ are used and a new model is generated (RCMFWP$_5$). The linear programming relaxation of the problem that is used in our column generation approach is given below. Here T_i is the number of columns generated for facility i, c_{it} is the objective coefficient for column t of facility i, b_{jit} equals to 1 if customer j is served in column t of facility i, 0 otherwise, u_j, v_i and w are related dual values for the related constraints.

RCMFWP$_5$:

$$\min \sum_{i=1}^{m} \sum_{t=1}^{T_i} c_{it} z_{it} \tag{7.20}$$

$$\text{s.t.} \sum_{i=1}^{m} \sum_{t=1}^{T_i} b_{jit} z_{it} \geq 1 \qquad j = 1, ..., n \qquad : u_j \tag{7.21}$$

$$\sum_{t=1}^{i_t} z_{it} \leq 1 \qquad i = 1, .., m \qquad : v_i \tag{7.22}$$

$$\sum_{i=1}^{m} \sum_{t=1}^{T_i} z_{it} \leq m \qquad : w \tag{7.23}$$

$$z_{it} \geq 0 \qquad i = 1, .., m; t = 1, ..., T_i. \tag{7.24}$$

In order to find the nonbasic variable with the minimum reduced cost, we must derive a subproblem related to this master problem. At every iteration (t), this subproblem must be solved for every facility to find the column with the minimum reduced cost. For $u_j^{(t)}$, $v_i^{(t)}$ and $w^{(t)}$ are the dual values of the related Constraints (7.21), (7.22) and (7.23) at iteration t respectively, $c_i^{(t)}$ is the cost and $\overline{c}_i^{(t)}$ is the reduced cost related to any column created at iteration t for facility i, the following calculations should be done to accomplish the pricing operation. The reduced cost is defined as

$$\overline{c}_i^{(t)} = c_i^{(t)} - \sum_{j=1}^{n} b_{ji}^{(t)} u_j^{(t)} + v_i^{(t)} - w^{(t)} \tag{7.25}$$

$$\text{where } b_{ji}^{(t)} = \begin{cases} 1, & \text{if customer } j \text{ is served by facility } i \text{ in} \\ & \text{column created at iteration } t \text{ for facility } i \\ 0, & \text{otherwise} \end{cases}$$

Then, for $\overline{c}_i^{(t)*}$ is the minimum reduced cost calculated among the columns of facility i and $\overline{c}^{(t)*}$ is the minimum reduced cost among all columns at iteration t, we can state that,

$$\overline{c}^{(t)*} = \min_{1\le i\le m} \left\{ \overline{c}_i^{(t)*} \right\} \tag{7.26}$$

$$\overline{c}_i^{(t)*} = \min_{b_{ji}^{(t)}} \left\{ c_i^{(t)} - \sum_{j=1}^{n} b_{ji}^{(t)} u_j^{(t)} + v_i^{(t)} - w^{(t)} \right\}. \tag{7.27}$$

The cost $c_i^{(t)}$ of the column of facility i at iteration t can be calculated using the expression

$$c_i^{(t)} = \min_{\mathbf{x}_i, b_{ji}^{(t)}} \left\{ \sum_{j=1}^{n} b_{ji}^{(t)} h_j \left[c_{ij} d\left(\mathbf{x}_i, \mathbf{a}_j\right) + \lambda_i^{(t)} \right] \right\}. \tag{7.28}$$

In order to increase the readability we continue our derivation without the iteration symbol superscript (t). After Combining (7.27) with (7.28) the minimum reduced cost $\overline{c}_i^*$ becomes

$$\overline{c}_i^* = \min_{b_{ji}} \left\{ \min_{x_i} \left\{ \sum_{j=1}^{n} b_{ji} h_j \left[c_{ij} d\left(\mathbf{x}_i, \mathbf{a}_j\right) + \lambda_i \right] \right\} - \sum_{j=1}^{n} b_{ji} u_j + v_i - w \right\}. \tag{7.29}$$

For $\gamma_i\left(\mathbf{x}_i, j\right) = h_j \left[c_{ij} d\left(\mathbf{x}_i, \mathbf{a}_j\right) + \lambda_i \right]$ (7.29) reduces to

$$\overline{c}_i^* = \min_{b_{ji}} \left\{ \min_{\mathbf{x}_i} \left\{ \sum_{j=1}^{n} b_{ji} \gamma_i\left(\mathbf{x}_i, j\right) \right\} - \sum_{j=1}^{n} b_{ji} u_j + v_i - w \right\}, \tag{7.30}$$

which turns into

$$\overline{c}_i^* = \min_{\mathbf{x}_i, b_{ji}} \left\{ \sum_{j=1}^{n} b_{ji} \left[\gamma_i\left(\mathbf{x}_i, j\right) - u_j \right] \right\} + v_i - w \tag{7.31}$$

after some simple algebraic manipulations. Since $b_{ji} \in \{0, 1\}$ and it is aimed to find the minimum over $\mathbf{x}_i$ and b_{ji},

$$b_{ji} = \begin{cases} 1, \text{ if } \left[\gamma_i\left(\mathbf{x}_i, j\right) - u_j\right] \leq 0 \\ 0, \text{ otherwise} \end{cases} \tag{7.32}$$

is obtained. In other words,

$$\min_{\mathbf{x}_i, b_{ji}} \left\{ \sum_{j=1}^{n} b_{ji} \left[\gamma_i\left(\mathbf{x}_i, j\right) - u_j\right] \right\} = \min_{\mathbf{x}_i} \left\{ \sum_{j=1}^{n} \min\left\{\gamma_i\left(\mathbf{x}_i, j\right) - u_j, 0\right\} \right\}. \tag{7.33}$$

After performing the change given in (7.33) on (7.31)

$$\overline{c}_i^* = \min_{\mathbf{x}_i} \left\{ \sum_{j=1}^{n} \min\left\{\gamma_i\left(\mathbf{x}_i, j\right) - u_j, 0\right\} \right\} + v_i - w. \tag{7.34}$$

Since,

$$\min\left\{\gamma_i\left(\mathbf{x}_i, j\right) - u_j, 0\right\} = \left[\gamma_i\left(\mathbf{x}_i, j\right) - u_j\right] - \max\left\{\gamma_i\left(\mathbf{x}_i, j\right) - u_j, 0\right\}, \tag{7.35}$$

(7.34) can be rewritten as,

$$\overline{c}_i^* = \min_{\mathbf{x}_i} \left\{ \sum_{j=1}^{n} \left[\gamma_i\left(\mathbf{x}_i, j\right) - u_j\right] - \sum_{j=1}^{n} \max\left\{\gamma_i\left(\mathbf{x}_i, j\right) - u_j, 0\right\} \right\} + v_i - w, \tag{7.36}$$

or equivalently as for a particular iteration t as,

$$\begin{aligned} \overline{c}_i^{(t)*} = \min_{\mathbf{x}_i} \left\{ \sum_{j=1}^{n} \gamma_i\left(\mathbf{x}_i, j\right) - \sum_{j=1}^{n} \max\left\{\gamma_i\left(\mathbf{x}_i, j\right) - u_j^{(t)}, 0\right\} \right\} - \sum_{j=1}^{n} u_j^{(t)} \\ + v_i^{(t)} - w^{(t)} \end{aligned} \tag{7.37}$$

because the dual variables $u_j^{(t)}$, $v_i^{(t)}$ and $w^{(t)}$ are independent of the location variables

$\mathbf{x}_i$ for $i = 1, ..., m$.

In (7.37), the last summation is the constant part of the inner minimization, and the first two summations are the two convex functions. The remaining parts are the constants. The first summation is convex since it is the summation of distance functions which are convex, and the second summation is the summation of the maximum functions which are also convex. In the literature, these types of problems where the objective function and the constraints can be expressed as the difference of two convex functions are known as the d.c. programming (DC) problems.

There are several methods to solve DC programming problems. One of them is to convert the problem into a concave minimization problem, which requires the introduction of auxiliary variables [30]. The model after converting the problem into a concave minimization problem becomes

CG^i:

$$\min F(\mathbf{x}_i, r_i) = r_i - \sum_{j=1}^{n} \max\left\{\gamma_i(\mathbf{x}_i, j) - u_j^{(t)}, 0\right\} - \sum_{j=1}^{n} u_j^{(t)} + v_i^{(t)} - w^{(t)} \tag{7.38}$$

$$\text{s.t.} \sum_{j=1}^{n} \gamma_i(\mathbf{x}_i, j) - r_i \leq 0 \tag{7.39}$$

$$r_i \geq 0 \tag{7.40}$$

$$\mathbf{x}_i \in \mathcal{H} \tag{7.41}$$

where $\mathcal{H}$ is the convex hull of the customer locations. Here r_i is the auxiliary variable. The concave minimization subproblem CG^i for every facility i is solved by outer approximation method. A detailed explanation of the outer approximation method is provided in Section 7.5.

After solving the subproblem CG^i for facility i, suppose $\mathbf{x}_i^*$, r_i^* are optimal values for problem related to facility i and $i^* = \arg\min_i \{F(\mathbf{x}_i^*, r_i^*)\}$. If $\overline{c}^*$ which is $\overline{c}_{i^*}^*$ is nonnegative then the model is optimal, meaning that the remaining columns have positive reduced costs and thus no need to add any more columns. However, if $\overline{c}^* < 0$ then attach the column $b_{ji^*}^{(t)*}$ after setting

$$b_{ji^*}^{(t)*} = \begin{cases} 1, \text{ if } \gamma_i\left(\mathbf{x}_i, j\right) - u_j < 0 \\ 0, \text{ otherwise} \end{cases} \qquad \text{for } j = 1, ..., n. \tag{7.42}$$

The basic column generation scheme for solving the RCMFWP$_5$, which is denoted as CG$_1$, is given in Figure 7.2.

```
1. Set $t \leftarrow 0$
2. Initialize the RCMFWP$_5$ model with a feasible solution
3. Repeat
   (a) $t \leftarrow t + 1$ and $\overline{c}^* \leftarrow +\infty$
   (b) Solve the main problem and find the dual values $u_i^{(t)}$, $v_j^{(t)}$ and $s^{(t)}$
   (c) For each facility $i$
       i. Solve the CG$^i$ problem for facility $i$.
       ii. If $\overline{c}_i^* \leq \overline{c}^*$, then $\overline{c}^* \leftarrow \overline{c}_i^* \leq \overline{c}^*$
   (d) If $\overline{c}^* < 0$, then insert column for facility $i$ which is found by Equation (7.42)
   (e) Until $\overline{c}^* \geq 0$
```

Figure 7.2. Basic column generation algorithm for the RCMFWP$_5$

As mentioned above, these concave minimization problems are solved by outer approximation [31] which has asymptotic convergence. In other words, the values that are found by concave minimization are not optimal but they are ϵ close to an optimal solution, or the solution reaches to optimality in an infinite number of iterations. In order to eliminate problems related to asymptotic convergence, a ξ value is decided and columns are created using (7.43) instead of (7.42):

$$b_{ji^*}^{(t)*} = \begin{cases} 1, \text{ if } \gamma_i\left(\mathbf{x}_i, j\right) - u_j \leq -\xi \\ 0 \text{ or } 1, \text{ if } -\xi < \gamma_i\left(\mathbf{x}_i, j\right) - u_j \leq \xi \\ 0, \text{ otherwise} \end{cases} \quad \text{for } j = 1, ..., n. \tag{7.43}$$

With the above modification, in column generation, there is a possibility to create more than one column with negative reduced cost for a single concave minimization problem which increases the convergence rate of the problem and decreases the running time.

One of the other ways to increase the number of columns created in every step is to add the every columns with negative reduced cost instead of adding the one with the most negative value. Theoretically, every column which has negative reduced cost must improve the master problem. So in our implementation, every generated column with negative reduced cost is added to the master problem which makes a slight decrease in the total running time.

On top of all this, the procedure given in Figure 7.3 which is named as column generation heuristic (CGH) is used in intermediate steps. This heuristic, which is run for every facility separately, starts from an initial location and creates the columns with negative reduced cost by updating its customer subset with the customers which decreases the reduced cost and finds an optimal location for the updated customer subset after solving a SFWP by using the Weiszfeld procedure [3]. Although CGH does not guarantee the most negative column, applying this approach does not cause suboptimal solutions for the relaxed problem since the optimality of the model is checked by solving a concave minimization problem for each facility. The solution is declared optimal and column generation approach is terminated when all the columns created for every facility by concave minimization have nonnegative reduced cost. In order to increase the number of columns created, the heuristic is modified as adding every column with negative reduced cost created in Step 3 and this modification makes slight improvements in some cases.

```
1. Set $\mathcal{J}_s^{\text{prev}} \leftarrow \mathcal{J}_s, \mathcal{J}_s \leftarrow \emptyset$
2. For all $j = 1, ..., n$ if $\gamma_i(\mathbf{x}_i, j) - u_j^{(t)} < 0$ then $\mathcal{J}_s \leftarrow \mathcal{J}_s \cup \{j\}$
3. Find $c_i^{(t)} = \min_{\mathbf{x}_i} \left\{ \sum_{j \in \mathcal{J}_s} \gamma_i(\mathbf{x}_i, j) \right\}$
4. If $\mathcal{J}_s^{\text{prev}} = \mathcal{J}_s$ then go to Step 5, else go to Step 1
5. Add newly created column containing $\mathcal{J}_s$ and facility $i$ and solve model
   RCMFWP5. If the objective function is not improved, then terminate, else go to
   Step 6
6. Update $u^{(t)}$, $t \leftarrow t + 1$
```

Figure 7.3. Column generation heuristic

The column generation algorithm used in computations (CG_2) is given in Figure 7.4. It makes calls to CGH given in Figure 7.3.

As mentioned before, by the column generation approach, the linear relaxation of the set covering model can be solved, which is a lower bound for the relaxed problem and can be used as a valid lower bound for the main problem. However, the exact value of the relaxed problem can be found by combining a branch and price procedure combined with the column generation approach.

7.4. Branch and Price

In this work, a branching rule that is similar to Ryan and Foster's [32] is applied. When a node is needed to be branched, first a branching pair is formed containing two customers which does not exist in zero or one branched list of the current node. In addition to the inherited zero and one branched sets from the father node, the left child adds the new branching pair to its zero branched list whereas the right child adds the new branching pair to its one branched list. For every customer pairs in zero branched list, the customers either exist together or both do not exist in the feasible columns. Similar to that, for every customer pairs in one branched list, either one of the customers exist or both do not exist in the set of feasible columns.

```
1. Set $t \leftarrow 0$
2. Initialize the RCMFWP$_5$ model with a feasible solution
3. Repeat
    (a) $t \leftarrow t + 1$ and $\overline{c}^* \leftarrow +\infty$
    (b) Solve the main problem and find the dual values $u_i^{(t)}$, $v_j^{(t)}$ and $s^{(t)}$
    (c) For each facility $i$
        i. Solve the subproblem CG$^i$ for facility $i$
        ii. If $\overline{c}_i^* \leq \overline{c}^*$, then $\overline{c}^* \leftarrow \overline{c}_i^* \leq \overline{c}^*$ and $i^* \leftarrow i$
        iii. If $\overline{c}_i^* \leq 0$, then
            A. Insert column for facility $i$ which is found by as in (7.43)
            B. Solve the main problem and find the dual values $u_i^{(t)}$, $v_j^{(t)}$ and $s^{(t)}$
            C. Run CGH for facility $i$ from the locations found in subproblem CG$^i$
    (d) Until $\overline{c}^* \geq 0$
```

Figure 7.4. Column generation algorithm for the RCMFWP$_5$

After branching the set of customer pairs have to handle by four types of sets: $\mathcal{Z}$, $\mathcal{O}$, $\mathcal{N}$ and $\mathcal{S}$. $\mathcal{Z}$ is the set of customer pairs which contains zero branched customer pairs whose customers are branched only in this pair. Similarly $\mathcal{O}$ includes the customer pairs branched in one branched list and both customers of the pair only exist in this branched pair. $\mathcal{N}$ is the set of customers which are not branched yet and $\mathcal{S}$ is the special set of customer group sets which contains customer groups, simple set of customers. A customer group set contains feasible customer groups of connected customer pairs. Two customer pairs are connected if either one of the customers are common in both pairs or common in their connected customer pairs. An example would clearify this set construction.

If there are 20 customers in the problem and current nodes zero and one branched set contains the following customer pairs:

$$\text{zero branched set} = \{\{1,2\},\{2,3\},\{4,5\},\{6,7\},\{8,9\}\}$$

$$\text{one branched set} = \{\{9,10\},\{11,12\},\{12,13\},\{14,15\},\{16,17\}\}$$

$\{1, 2\}$ and $\{2, 3\}$ are connected. Similar to that $\{8, 9\}$ and $\{9, 10\}$, and $\{11, 12\}$ and $\{12, 13\}$ are also connected groups. Furthermore, $\mathcal{N}$, $\mathcal{Z}$, $\mathcal{O}$ and $\mathcal{S}$ include the following set of customer pairs and sets:

$$\begin{aligned}
\mathcal{N} &= \{18, 19, 20\} \\
\mathcal{Z} &= \{\{4, 5\}, \{6, 7\}\} \\
\mathcal{O} &= \{\{14, 15\}, \{16, 17\}\} \\
\mathcal{S} &= \{\{\emptyset, \{1, 2, 3\}\}, \{\emptyset, \{10\}, \{8, 9\}\}, \{\emptyset, \{11\}, \{12\}, \{13\}, \{11, 13\}\}\}
\end{aligned}$$

In the branch and price approach, every nodes' subproblem is different than the others because of their pricing. Before finding the pricing function let us define sets given above properly. For $\mathcal{J}$ is the set of all customers, $\mathcal{N} = \left\{n_1, n_2, ..., n_{|\mathcal{N}|}\right\}$ where $n_1, n_2, ...,$ and $n_{|\mathcal{N}|}$ are all in $\mathcal{J}$, $\mathcal{Z} = \left\{\{z_{11}, z_{12}\}, ..., \left\{z_{|\mathcal{Z}|1}, z_{|\mathcal{Z}|2}\right\}\right\}$ where $z_{11}, z_{12}, ..., z_{|\mathcal{Z}|1}$ and $z_{|\mathcal{Z}|2}$ are all in $\mathcal{J}$, $\mathcal{O} = \left\{\{o_{11}, o_{12}\}, ..., \left\{o_{|\mathcal{O}|1}, o_{|\mathcal{O}|2}\right\}\right\}$ where $o_{11}, o_{12}, ..., o_{|\mathcal{O}|1}$ and $o_{|\mathcal{O}|2}$ are all in $\mathcal{J}$, $\mathcal{S} = \left\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{|\mathcal{S}|}\right\}$ where $\mathcal{S}_k = \left\{\mathcal{S}_{k1}, \mathcal{S}_{k2}, ..., \mathcal{S}_{k|\mathcal{S}_1|}\right\}$ for $k = 1, ..., |\mathcal{S}|$ and $\mathcal{S}_{kl} = \left\{s_{kl1}, s_{kl2}, ..., s_{kl|\mathcal{S}_{11}|}\right\}$ for $k = 1, ..., |\mathcal{S}|$ and $l = 1, ..., |\mathcal{S}_k|$ and s_{klp} is an element of $\mathcal{J}$ for $k = 1, ..., |\mathcal{S}|$, $l = 1, ..., |\mathcal{S}_k|$ and $p = 1, ..., |\mathcal{S}_{kl}|$.

As it is defined before, $\mathcal{N}$ is the set of customers which are not branched before, $\mathcal{Z}$ and $\mathcal{O}$ sets are set of customer pairs which are zero and one branched and their elements (customers) only exist in one branch and $\mathcal{S}$ is the special set for the sets of customer subsets, $\mathcal{S}_k$ is one of the set of customer subsets, $\mathcal{S}_{kl}$ is one of the customer subsets of $\mathcal{S}_k$ and s_{klp} is the p^{th} customer in the customer subset $\mathcal{S}_{kl}$. The aim of this division is that, only customers of one and only one customer group from each customer group set can exist in the valid column for this node.

Now we can start to find the pricing of the subproblem for the column of facility i at iteration (t). Similar to the column generation procedure

$$\bar{c}_i^{(t)} = c_i^{(t)} - \sum_{j=1}^{n} b_{ji}^{(t)} u_j^{(t)} + v_i^{(t)} - w^{(t)}, \tag{7.44}$$

and

$$\overline{c}_i^{(t)*} = \min\left\{ c_i^{(t)} - \sum_{j=1}^{n} b_{ji}^{(t)} u_j^{(t)} + v_i^{(t)} \right\} - w^{(t)}. \tag{7.45}$$

Different than the pure column generation explained previously

$$\begin{aligned} c_i^{(t)} = \min_{\mathbf{x}_i} & \left\{ \sum_{j=1}^{|\mathcal{N}|} b_{n_j i}^{(t)} \gamma_i \left(\mathbf{x}_i, n_j\right) \right. \\ & + \sum_{j=1}^{|\mathcal{Z}|} b_{z_{j1} i}^{(t)} \left[\gamma_i \left(\mathbf{x}_i, z_{j1}\right) + \gamma_i \left(\mathbf{x}_i, z_{j2}\right)\right] \\ & + \sum_{\substack{j=1 \\ b_{o_{j1} i}^{(t)} + b_{o_{j2} i}^{(t)} \leq 1}}^{|\mathcal{O}|} \left[b_{o_{j1} i}^{(t)} \gamma_i \left(\mathbf{x}_i, o_{j1}\right) + b_{o_{j2} i}^{(t)} \gamma_i \left(\mathbf{x}_i, o_{j2}\right) \right] \\ & \left. + \sum_{l=1}^{|\mathcal{S}|} \sum_{\substack{k=1 \\ \sum_{k=1}^{|\mathcal{S}_l|} b_{s_{kl} i}^{(t)} = 1}}^{|\mathcal{S}_l|} b_{s_{kl} i}^{(t)} \left[\sum_{j=1}^{|\mathcal{S}_{kl}|} \gamma_i \left(\mathbf{x}_i, s_{jkl}\right) \right] \right\}. \end{aligned} \tag{7.46}$$

First part under summation, which is for elements of $\mathcal{N}$, is similar to the column generation. Second summation is for customer pairs in $\mathcal{Z}$ and since the customers of each pair of this set either exist together or do not exist at all, their decision variables are equal to each other. Third summation includes customer pairs in $\mathcal{O}$ where the customers in the pairs of this set cannot exist together, either one of them exists in the generated columns or both do not exist. This property is shown as a condition over the summation. Last summation is for the elements of $\mathcal{S}$. Only customers of one of the customer group in each customer group set can exist and it is added as a condition over the summation again. Again the subscript (t) is dropped to increase the readability. After inserting cost function (7.46) to (7.45), $\overline{c}_i^*$ becomes

$$\begin{aligned}
\overline{c}_i^* =& \min_{\mathbf{x}_i} \Bigg\{ \sum_{j=1}^{|\mathcal{N}|} b_{n_j i} \gamma_i \left(\mathbf{x}_i, n_j\right) + \sum_{j=1}^{|\mathcal{Z}|} b_{z_{j1} i} \left[\gamma_i \left(\mathbf{x}_i, z_{j1}\right) + \gamma_i \left(\mathbf{x}_i, z_{j2}\right)\right] \\
&+ \sum_{\substack{j=1 \\ b_{o_{j1} i} + b_{o_{j2} i} \le 1}}^{|\mathcal{O}|} \left[b_{o_{j1} i} \gamma_i \left(\mathbf{x}_i, o_{j1}\right) + b_{o_{j2} i} \gamma_i \left(\mathbf{x}_i, o_{j2}\right)\right] \\
&+ \sum_{l=1}^{|\mathcal{S}|} \sum_{\substack{k=1 \\ \sum_{k=1}^{|\mathcal{S}_l|} b_{s_{kl} i} = 1}}^{|\mathcal{S}_l|} b_{s_{kl} i} \left[\sum_{j=1}^{|\mathcal{S}_{kl}|} \gamma_i \left(\mathbf{x}_i, s_{jkl}\right)\right] \Bigg\} - \sum_{j=1}^{n} b_{ji} u_j + v_i - w.
\end{aligned} \tag{7.47}$$

If we group the equations with the same decision variables, we will end up with the expression

$$\begin{aligned}
\overline{c}_i^* =& \min_{\mathbf{x}_i} \Bigg\{ \sum_{j=1}^{|\mathcal{N}|} \left[b_{n_j i} \gamma_i \left(\mathbf{x}_i, n_j\right) - u_{n_j}\right] \\
&+ \sum_{j=1}^{|\mathcal{Z}|} \left\{ b_{z_{j1} i} \left[\gamma_i \left(\mathbf{x}_i, z_{j1}\right) + \gamma_i \left(\mathbf{x}_i, z_{j2}\right)\right] - \left[u_{z_{j1}} + u_{z_{j2}}\right] \right\} \\
&+ \sum_{\substack{j=1 \\ b_{o_{j1} i} + b_{o_{j2} i} \le 1}}^{|\mathcal{O}|} \left\{ b_{o_{j1} i} \left[\gamma_i \left(\mathbf{x}_i, z_{j1}\right) - u_{o_{j1}}\right] + b_{o_{j2} i} \left[\gamma_i \left(\mathbf{x}_i, z_{j2}\right) - u_{o_{j2}}\right] \right\} \\
&+ \sum_{l=1}^{|\mathcal{S}|} \sum_{\substack{k=1 \\ \sum_{k=1}^{|\mathcal{S}_l|} b_{s_{kl} it} = 1}}^{|\mathcal{S}_l|} b_{s_{kl} i} \left[\sum_{j=1}^{|\mathcal{S}_{kl}|} \gamma_i \left(\mathbf{x}_i, s_{jkl}\right) - u_{s_{jkl}}\right] \Bigg\} + v_i - w.
\end{aligned} \tag{7.48}$$

We can rewrite (7.48) as,

$$\begin{aligned}\overline{c}_i^* =&\min_{\mathbf{x}_i}\left\{\sum_{j=1}^{|\mathcal{N}|}\min\left\{\gamma_i\left(\mathbf{x}_i,n_j\right)-u_{n_j},0\right\}\right.\\&+\sum_{j=1}^{|\mathcal{Z}|}\min\left\{\gamma_i\left(\mathbf{x}_i,z_{j1}\right)+\gamma_i\left(\mathbf{x}_i,z_{j2}\right)-\left(u_{z_{j1}}+u_{z_{j2}}\right),0\right\}\\&+\sum_{j=1}^{|\mathcal{O}|}\min\left\{\gamma_i\left(\mathbf{x}_i,o_{j1}\right)-u_{o_{j1}},\gamma_i\left(\mathbf{x}_i,o_{j2}\right)-u_{o_{j2}},0\right\}\\&+\sum_{l=1}^{|\mathcal{S}|}\min_k\left\{\sum_{j=1}^{|\mathcal{S}_{kl}|}\gamma_i\left(\mathbf{x}_i,s_{jkl}\right)-u_{s_{jkl}}\right\}+v_i-w,\end{aligned}\tag{7.49}$$

from which

$$\begin{aligned}\overline{c}_i^* =&\min_{\mathbf{x}_i}\left\{\sum_{j=1}^{|\mathcal{N}|}\alpha_i'\left(\mathbf{x}_i,n_j\right)+\sum_{j=1}^{|\mathcal{Z}|}\beta_i'\left(\mathbf{x}_i,z_{j1},z_{j1}\right)\right.\\&\left.+\sum_{j=1}^{|\mathcal{O}|}\delta_i'\left(\mathbf{x}_i,o_{j1},o_{j2}\right)+\sum_{l=1}^{|\mathcal{S}|}\mu_i'\left(\mathbf{x}_i,\mathcal{S}_l\right)\right\}+v_i-w\end{aligned}\tag{7.50}$$

follows for

$$\begin{aligned}\gamma_i^{(t)\prime}\left(\mathbf{x}_i,j\right) =&h_j\left[c_{ij}d\left(\mathbf{x}_i,a_j\right)+\lambda_i\right]-u_j^{(t)},\\\alpha_i^{(t)\prime}\left(\mathbf{x}_i,j\right) =&\min\left\{\gamma_i^{(t)\prime}\left(\mathbf{x}_i,j\right),0\right\},\\\beta_i^{(t)\prime}\left(\mathbf{x}_i,j_1,j_2\right) =&\min\left\{\gamma_i^{(t)\prime}\left(\mathbf{x}_i,j_1\right)+\gamma_i^{(t)\prime}\left(\mathbf{x}_i,j_2\right),0\right\},\\\delta_i^{(t)\prime}\left(\mathbf{x}_i,j_1,j_2\right) =&\min\left\{\gamma_i^{(t)\prime}\left(\mathbf{x}_i,j_1\right),\gamma_i^{(t)\prime}\left(\mathbf{x}_i,j_2\right),0\right\},\\\nu_i^{(t)\prime}\left(\mathbf{x}_i,\mathcal{P}\right) =&\sum_{j\in\mathcal{P}}\gamma_i^{(t)\prime}\left(\mathbf{x}_i,j\right),\\\mu_i^{(t)\prime}\left(\mathbf{x}_i,\left\{\mathcal{P}_1,...,\mathcal{P}_n\right\}\right) =&\min_l\left\{\nu_i^{(t)\prime}\left(\mathbf{x}_i,\mathcal{P}_l\right)\right\}.\end{aligned}$$

For,

$$\begin{aligned}
\overline{\alpha}_i^{(t)}\left(\mathbf{x}_i, j\right) =& \max\left\{\gamma_i^{(t)\prime}\left(\mathbf{x}_i, j\right), 0\right\} \\
\overline{\beta}_i^{(t)}\left(\mathbf{x}_i, j_1, j_2\right) =& \max\left\{\gamma_i^{(t)\prime}\left(\mathbf{x}_i, j_1\right) + \gamma_i^{(t)\prime}\left(\mathbf{x}_i, j_2\right), 0\right\} \\
\overline{\delta}_i^{(t)}\left(\mathbf{x}_i, j_1, j_2\right) =& \max\left\{\gamma_i^{(t)\prime}\left(\mathbf{x}_i, j_1\right), \gamma_i^{(t)\prime}\left(\mathbf{x}_i, j_2\right)\right\} + \max\left\{\gamma_i^{(t)\prime}\left(\mathbf{x}_i, j_1\right), 0\right\} \\
&+ \max\left\{\gamma_i^{(t)\prime}\left(\mathbf{x}_i, j_2\right), 0\right\} \\
\underline{\alpha}_i^{(t)}\left(\mathbf{x}_i, j\right) =& \gamma_i\left(\mathbf{x}_i, j\right) \\
\underline{\beta}_i^{(t)}\left(\mathbf{x}_i, j_1, j_2\right) =& \gamma_i\left(\mathbf{x}_i, j_1\right) + \gamma_i\left(\mathbf{x}_i, j_2\right) \\
\underline{\delta}_i^{(t)}\left(\mathbf{x}_i, j_1, j_2\right) =& \gamma_i\left(\mathbf{x}_i, j_1\right) + \gamma_i\left(\mathbf{x}_i, j_2\right) + \max\left\{\gamma_i^{(t)\prime}\left(\mathbf{x}_i, j_1\right), \gamma_i^{(t)\prime}\left(\mathbf{x}_i, j_2\right), 0\right\}
\end{aligned}$$

Although $\underline{\alpha}_i^{(t)}$ and $\underline{\beta}_i^{(t)}$ remain unchanged through the iterations, the superscript (t) is still used for the sake of completeness. Then for $\mathcal{S} = \{\mathcal{S}_1, ...\mathcal{S}_n\}$, $\mathcal{S}_k = \{\mathcal{S}_{k1}, ..., \mathcal{S}_{kl}, ..., \mathcal{S}_{|\mathcal{S}_k|}\}$, and $\mathcal{S}_{kl} = \left\{s_{1kl}, ..., s_{pkl}, ..., s_{|\mathcal{S}_{kl}|}\right\}$ the variables

$$\begin{aligned}
\rho_i^{(t)}\left(\mathbf{x}_i, \{\mathcal{S}_1, ...\mathcal{S}_n\}, q\right) &= \sum_{\substack{\mathcal{Q}\subseteq\mathcal{S} \\ |\mathcal{Q}|=q}} \max_{\mathcal{S}\in\mathcal{Q}} \left\{\nu_i^{(t)\prime}\left(\mathbf{x}_i, \mathcal{S}\right)\right\}, \\
\overline{\theta}_i^{(t)}\left(\mathbf{x}_i, \{\mathcal{S}_1, ..., \mathcal{S}_n\}\right) &= \sum_{\substack{q=2 \\ q\, even}}^{n} \rho_i^{(t)}\left(\mathbf{x}_i, \{\mathcal{S}_1, ..., \mathcal{S}_n\}, q\right),
\end{aligned}$$

and

$$\underline{\theta}_i^{(t)}\left(\mathbf{x}_i, \{\mathcal{S}_1, ..., \mathcal{S}_n\}\right) = \sum_{q=1}^{n}\sum_{l=1}^{|\mathcal{S}_n|}\sum_{p=1}^{|\mathcal{S}_{kl}|} \gamma_i\left(\mathbf{x}_i, s_{pkl}\right) + \sum_{\substack{q=3 \\ q\, odd}}^{n} \rho_i^{(t)}\left(\mathbf{x}_i, \{\mathcal{S}_1, ..., \mathcal{S}_n\}, q\right)$$

of (7.50) can be written as

$$\alpha_i^{(t)\prime}\left(\mathbf{x}_i, j\right) = \underline{\alpha}_i^{(t)}\left(\mathbf{x}_i, j\right) - \overline{\alpha}_i^{(t)}\left(\mathbf{x}_i, j\right) - u_j^{(t)}, \tag{7.51}$$

$$\beta_i^{(t)\prime}\left(\mathbf{x}_i, j_1, j_2\right) = \underline{\beta}_i^{(t)}\left(\mathbf{x}_i, j_1, j_2\right) - \overline{\beta}_i^{(t)}\left(\mathbf{x}_i, j_1, j_2\right) - \left(u_{j_1}^{(t)} + u_{j_2}^{(t)}\right), \tag{7.52}$$

$$\delta_i^{(t)\prime}\left(\mathbf{x}_i, j_1, j_2\right) = \underline{\delta}_i^{(t)}\left(\mathbf{x}_i, j_1, j_2\right) - \overline{\delta}_i^{(t)}\left(\mathbf{x}_i, j_1, j_2\right) - \left(u_{j_1}^{(t)} + u_{j_2}^{(t)}\right), \tag{7.53}$$

and

$$\mu_i^{(t)\prime}\left(\mathbf{x}_i,\{\mathcal{S}_1,...\mathcal{S}_n\}\right) = \underline{\theta}_i^{(t)}\left(\mathbf{x}_i,\{\mathcal{S}_1,...,\mathcal{S}_n\}\right) - \overline{\theta}_i^{(t)}\left(\mathbf{x}_i,\{\mathcal{S}_1,...,\mathcal{S}_n\}\right) - \sum_{k=1}^{n}\sum_{l=1}^{|\mathcal{S}_n|}\sum_{p=1}^{|\mathcal{S}_{kl}|} u_{s_{pkl}}^{(t)}. \tag{7.54}$$

After replacing the variables in (7.50) with their definitions (7.51) - (7.54), we have

$$\begin{aligned}\overline{c}_i^{(t)*} = \min_{\mathbf{x}_i} \Bigg\{ & \sum_{j=1}^{|\mathcal{N}|}\left[\underline{\alpha}_i^{(t)}\left(\mathbf{x}_i,n_j\right) - \overline{\alpha}_i^{(t)}\left(\mathbf{x}_i,n_j\right) - u_{n_j}^{(t)}\right] \\ & + \sum_{j=1}^{|\mathcal{Z}|}\left[\underline{\beta}_i^{(t)}\left(\mathbf{x}_i,z_{j1},z_{j2}\right) - \overline{\beta}_i^{(t)}\left(\mathbf{x}_i,z_{j1},z_{j2}\right) - \left(u_{z_{j1}}^{(t)} + u_{z_{j2}}^{(t)}\right)\right] \\ & + \sum_{j=1}^{|\mathcal{O}|}\left[\underline{\delta}_i^{(t)}\left(\mathbf{x}_i,o_{j1},o_{j2}\right) - \overline{\delta}_i^{(t)}\left(\mathbf{x}_i,o_{j1},o_{j2}\right) - \left(u_{o_{z_{j1}}}^{(t)} + u_{o_{z_{j2}}}^{(t)}\right)\right] \\ & + \sum_{k=1}^{|\mathcal{S}|}\left[\underline{\theta}_i^{(t)}\left(\mathbf{x}_i,\mathcal{S}_l\right) - \overline{\theta}_i^{(t)}\left(\mathbf{x}_i,\mathcal{S}_l\right)\right] - \sum_{k=1}^{|\mathcal{S}|}\sum_{l=1}^{|\mathcal{S}_k|}\sum_{j=1}^{|\mathcal{S}_{kl}|} u_{s_{jkl}}^{(t)} \Bigg\} + v_i^{(t)} - w^{(t)}.\end{aligned} \tag{7.55}$$

Regrouping of the variables results in

$$\begin{aligned}\overline{c}_i^{(t)*} = \min_{x_i} \Bigg\{ & \Bigg[\sum_{j=1}^{|\mathcal{N}|}\underline{\alpha}_i^{(t)}\left(\mathbf{x}_i,n_j\right) + \sum_{j=1}^{|\mathcal{Z}|}\underline{\beta}_i^{(t)}\left(\mathbf{x}_i,z_{j1},z_{j2}\right) + \sum_{j=1}^{|\mathcal{O}|}\underline{\delta}_i^{(t)}\left(\mathbf{x}_i,o_{j1},o_{j2}\right) \\ & + \sum_{l=1}^{|\mathcal{S}|}\underline{\theta}_i^{(t)}\left(\mathbf{x}_i,\mathcal{S}_l\right)\Bigg] - \Bigg[\sum_{j=1}^{|\mathcal{N}|}\overline{\alpha}_i^{(t)}\left(\mathbf{x}_i,n_j\right) + \sum_{j=1}^{|\mathcal{Z}|}\overline{\beta}_i^{(t)}\left(\mathbf{x}_i,z_{j1},z_{j2}\right) \\ & + \sum_{j=1}^{|\mathcal{O}|}\overline{\delta}_i^{(t)}\left(\mathbf{x}_i,o_{j1},o_{j2}\right) + \sum_{l=1}^{|\mathcal{S}|}\overline{\theta}_i^{(t)}\left(\mathbf{x}_i,\mathcal{S}_l\right)\Bigg] + \Bigg[\sum_{j=1}^{|\mathcal{N}|}u_{n_j}^{(t)} + \sum_{j=1}^{|\mathcal{Z}|}\left(u_{z_{j1}}^{(t)},u_{z_{j2}}^{(t)}\right) \\ & + \sum_{j=1}^{|\mathcal{O}|}\left(u_{o_{j1}}^{(t)},u_{o_{j2}}^{(t)}\right) + \sum_{l=1}^{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}_k|}\sum_{j=1}^{|\mathcal{S}_{kl}|} u_{s_{jkl}}^{(t)}\Bigg]\Bigg\} + v_i^{(t)} - w^{(t)}.\end{aligned} \tag{7.56}$$

Similar to the column generation, the last summation is the constant part of the minimization, and the first two summations are two convex functions, since they are the difference of convex functions, the pricing problem becomes a DC programming problem.

As it is previously done in the column generation, this problem can be rewritten as a concave minimization problem. For,

$$\psi_i^{(t)} = -\left[\sum_{j=1}^{|\mathcal{N}|} u_{n_j}^{(t)} + \sum_{j=1}^{|\mathcal{Z}|}\left(u_{z_{j1}}^{(t)}, u_{z_{j2}}^{(t)}\right) + \sum_{j=1}^{|\mathcal{O}|}\left(u_{o_{j1}}^{(t)}, u_{o_{j2}}^{(t)}\right) + \sum_{l=1}^{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}_k|}\sum_{j=1}^{|\mathcal{S}_{kl}|} u_{s_{jkl}}^{(t)}\right] + v_i^{(t)} - w^{(t)}$$

the concave minimization submodel (BP^i) becomes:

BP^i:

$$\begin{aligned} \min F(\mathbf{x}_i, r_i) = & r_i - \left[\sum_{j=1}^{|\mathcal{N}|} \overline{\alpha}_i^{(t)}(\mathbf{x}_i, n_j) + \sum_{j=1}^{|\mathcal{Z}|} \overline{\beta}_i^{(t)}(\mathbf{x}_i, z_{j1}, z_{j2}) \right.\\ & \left. + \sum_{j=1}^{|\mathcal{O}|} \overline{\delta}_i^{(t)}(\mathbf{x}_i, o_{j1}, o_{j2}) + \sum_{l=1}^{|\mathbf{S}|} \overline{\theta}_i^{(t)}(\mathbf{x}_i, \mathcal{S}_l)\right] + \psi^{(t)} \end{aligned} \tag{7.57}$$

$$\begin{aligned} \text{s.t. } & \left[\sum_{j=1}^{|\mathcal{N}|} \underline{\alpha}_i^{(t)}(\mathbf{x}_i, n_j) + \sum_{j=1}^{|\mathcal{Z}|} \underline{\beta}_i^{(t)}(\mathbf{x}_i, z_{j1}, z_{j2}) \right.\\ & \left. + \sum_{j=1}^{|\mathcal{O}|} \underline{\delta}_i^{(t)}(\mathbf{x}_i, o_{j1}, o_{j2}) + \sum_{l=1}^{|\mathcal{S}|} \underline{\theta}_i^{(t)}(\mathbf{x}_i, \mathcal{S}_l)\right] - r_i \leq 0 \end{aligned} \tag{7.58}$$

$$r_i \geq 0 \tag{7.59}$$

$$\mathbf{x}_i \in \mathcal{H}, \tag{7.60}$$

where $\mathcal{H}$ is the convex hull of the customer locations and r_i is the auxilliary variable.

For $i^* = \arg\min_i \{\min F(\mathbf{x}_i, r_i)\}$ and $\min F(\mathbf{x}_{i^*}, r_{i^*}) = F(\mathbf{x}_{i^*}^*, r_{i^*}^*)$, if $\overline{c}^* \geq 0$ the model is optimal, there is no need to add any more columns. However, if $\overline{c}^* < 0$ then attach the column $b_{i*}^{(t)*}$ as such to the model:

$$
\begin{aligned}
& b_{n_j i^*}^{(t)*} = \begin{cases} 1, \text{ if } \alpha_i^{(t)\prime}(\mathbf{x}_i, n_j) \leq 0 \\ 0, \text{ otherwise} \end{cases} \qquad \text{for } j = 1, ..., |\mathcal{N}| \\
& b_{z_{j1} i^*}^{(t)*} = b_{z_{j2} i^*}^{(t)*} = \begin{cases} 1, \text{ if } \beta_i^{(t)\prime}(\mathbf{x}_i, z_{j1}, z_{j2}) \leq 0 \\ 0, \text{ otherwise} \end{cases} \qquad \text{for } j = 1, ..., |\mathcal{Z}|, \\
& b_{o_{j1} i^*}^{(t)*} = 1, b_{o_{j2} i^*}^{(t)*} = 0, \text{ if } \delta_i^{(t)\prime}(\mathbf{x}_i, o_{j1}, o_{j2}) = \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j1}) \qquad \text{for } j = 1, ..., |\mathcal{O}|, \\
& b_{o_{j1} i^*}^{(t)*} = 0, b_{o_{j2} i^*}^{(t)*} = 1, \text{ if } \delta_i^{(t)\prime}(\mathbf{x}_i, o_{j1}, o_{j2}) = \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j2}) \qquad \text{for } j = 1, ..., |\mathcal{O}|, \\
& b_{o_{j1} i^*}^{(t)*} = 0, b_{o_{j2} i^*}^{(t)*} = 0, \text{ if } \delta_i^{(t)\prime}(\mathbf{x}_i, o_{j1}, o_{j2}) = 0 \qquad \text{for } j = 1, ..., |\mathcal{O}|, \\
& b_{s_{jkl}}^{(t)*} = \begin{cases} 1, \text{ if } \mu_i^{(t)}(\mathbf{x}_i, \mathcal{S}_k) = \nu_i(\mathbf{x}_i, \mathcal{S}_{kl}) \\ 0, \text{ otherwise} \end{cases} \\
& \qquad\qquad \text{for } k = 1, ..., |\mathcal{S}|; l = 1, ..., |\mathcal{S}_k|; j = 1, ..., |\mathcal{S}_{kl}|.
\end{aligned}
\tag{7.61}
$$

For $\mathcal{M}_l$ is any node and $M_l = \{\mathcal{M}_l^0, \mathcal{M}_l^1\}$ is node l with zero branched pairs set M_l^0, one branched set M_l^1 and Z_l is the lower bound for node l, the basic branch and price heuristic is formerly listed in Figure 7.5.

Concave minimization problem is again solved by outer approximation [31], and this method has asymptotic convergence. In order to eliminate this issue and speed up the procedure column generation, Strategy (7.62) is used in practice instead of (7.61). Here,

```
1. Set $t \leftarrow 0$, $\mathcal{B} \leftarrow \emptyset$ and $Z_{\text{LB}} \leftarrow +\infty$
2. Let $\mathcal{M}_0 \leftarrow \{\{\},\{\}\}$ and $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathcal{M}_0\}$
3. Repeat (selecting node)
    (a) Choose $\mathcal{M}_l$ such that $\mathcal{M}_l \in \mathcal{B}$, $\mathcal{B} \leftarrow \mathcal{B} \setminus \mathcal{M}_l$
    (b) Find $\mathcal{N}, \mathcal{Z}, \mathcal{O}$ and $\mathcal{S}$ sets by definitions given above for the node $M_l$
    (c) Initialize the RCMFWP$_5$ model with a feasible solution
    (d) Repeat (solving node)
        i. $t \leftarrow t + 1$ and $\overline{c}^* \leftarrow +\infty$
        ii. Solve the main problem and find the dual values $u_i^{(t)}$, $v_j^{(t)}$ and $w^{(t)}$
        iii. For each facility $i$
            A. Solve the BP$^i$ problem for facility $i$
            B. If $\overline{c}_i^* \leq \overline{c}^*$, then $\overline{c}^* \leftarrow \overline{c}_i^* \leq \overline{c}^*$ and $i^* \leftarrow i$
        iv. If $\overline{c}^* < 0$, then insert column for facility $i$ which is found using (7.61)
        v. Until $\overline{c}^* \geq 0$
    (e) Set $Z_l \leftarrow$ the optimal solution for node $l$
    (f) If the solution is not binary and $Z_{\text{LB}} < Z_l$ then choose a branching pair
        $\{j_1, j_2\}$, form two new nodes $\mathcal{M}_{l(\text{left})}$ and $\mathcal{M}_{l(\text{right})}$:
        i. $\mathcal{M}_{l(\text{left})} \leftarrow \left\{\mathcal{M}^0_{l(\text{left})}, M^1_l\right\}$ where $\mathcal{M}^0_{l(\text{left})} \leftarrow M^0_l \cup \{j_1, j_2\}$
        ii. $\mathcal{M}_{l(\text{right})} \leftarrow \left\{\mathcal{M}^0_l, \mathcal{M}^1_{l(\text{right})}\right\}$ where $\mathcal{M}^1_{l(\text{right})} \leftarrow M^1_l \cup \{j_1, j_2\}$
        iii. $Z_{l(\text{left})} \leftarrow Z_l$, $Z_{l(\text{right})} \leftarrow Z_l$ and $\mathcal{B} \leftarrow \mathcal{B} \cup \left\{\mathcal{M}_{l(\text{left})}, \mathcal{M}_{l(\text{right})}\right\}$
    (g) If solution is binary and $Z_{\text{LB}} < Z_l$ then $Z_{\text{LB}} \leftarrow Z_l$ and remove nodes with
        lower bounds less than $Z_{\text{LB}}$ from $\mathcal{B}$
    (h) Until $\mathcal{B} = \emptyset$
```

Figure 7.5. Basic branch and price algorithm for the RCMFWP$_5$

$$b_{n_j i^*}^{(t)*} = \begin{cases} 1, \text{ if } \alpha_i^{(t)\prime}(\mathbf{x}_i, n_j) < -\xi \\ 0 \text{ or } 1, \text{ if } -\xi \leq \alpha_i^{(t)\prime}(\mathbf{x}_i, n_j) \leq \xi \qquad \text{for } j = 1, ..., |\mathcal{N}|; \\ 0, \text{ otherwise} \end{cases}$$

$$b_{z_{j1} i^*}^{(t)*} = b_{z_{j2} i^*}^{(t)*} = \begin{cases} 1, \text{ if } \beta_i^{(t)\prime}(\mathbf{x}_i, z_{j1}, z_{j2}) < -\xi \\ 0 \text{ or } 1, \text{ if } -\xi \leq \beta_i^{(t)\prime}(\mathbf{x}_i, z_{j1}, z_{j2}) \leq \xi \qquad \text{for } j = 1, ..., |\mathcal{Z}|; \\ 0, \text{ otherwise} \end{cases}$$

$$\begin{array}{l} \text{Besides if } \min\left\{\gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j1}), \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j2})\right\} < -\xi \\ \qquad \left.\begin{array}{r} b_{o_{j1} i^*}^{(t)*} = 1, b_{o_{j2} i^*}^{(t)*} = 0 \\ \text{or } b_{o_{j1} i^*}^{(t)*} = 0, b_{o_{j2} i^*}^{(t)*} = 1 \end{array}\right\} \text{ if } \left|\gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j1}) - \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j2})\right| \leq \xi \\ \qquad b_{o_{j1} i^*}^{(t)} = 1, b_{o_{j2} i^*}^{(t)} = 0, \text{ if } \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j1}) \leq \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j2}) + \xi \\ \qquad b_{o_{j1} i^*}^{(t)} = 0, b_{o_{j2} i^*}^{(t)*} = 1, \text{ if } \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j2}) \leq \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j1}) + \xi, \\ \text{If } -\xi \leq \min\left\{\gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j1}), \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j2})\right\} \leq \xi \\ \qquad \left.\begin{array}{r} b_{o_{j1} i^*}^{(t)*} = 1, b_{o_{j2} i^*}^{(t)*} = 0 \\ \text{or } b_{o_{j1} i^*}^{(t)} = 0, b_{o_{j2} i^*}^{(t)} = 1 \\ \text{or } b_{o_{j1} i^*}^{(t)} = 0, b_{o_{j2} i^*}^{(t)} = 0 \end{array}\right\} \text{ if } \left|\gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j1}) - \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j2})\right| \leq \xi \\ \qquad b_{o_{j1} i^*}^{(t)*} = 0 \text{ or } 1, b_{o_{j2} i^*}^{(t)*} = 0, \text{ if } \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j1}) \leq \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j2}) + \xi \\ \qquad b_{o_{j1} i^*}^{(t)*} = 0, b_{o_{j2} i^*}^{(t)*} = 0 \text{ or } 1, \text{ if } \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j2}) \leq \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j1}) + \xi, \\ \text{If } \min\left\{\gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j1}), \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j2})\right\} > \xi \text{ then } b_{o_{j1} i^*}^{(t)*} = 0, b_{o_{j2} i^*}^{(t)*} = 0, \\ \hfill \text{for } j = 1, ..., |\mathcal{O}|; \end{array} \tag{7.62}$$

$$b_{s_{jkl}}^{(t)*} = \begin{cases} 1, \text{ if } \nu_i(\mathbf{x}_i, \mathcal{S}_{kl}) - \mu_i^{(t)}(\mathbf{x}_i, \mathcal{S}_k) < -\xi \\ 0 \text{ or } 1, if -\xi \leq \nu_i(\mathbf{x}_i, \mathcal{S}_{kl}) - \mu_i^{(t)}(\mathbf{x}_i, \mathcal{S}_k) \leq \xi \\ 0, \text{ otherwise} \end{cases}$$

$$\text{for } k = 1, ..., |\mathcal{S}|; \ l = 1, ..., |\mathcal{S}_k|; j = 1, ..., |\mathcal{S}_{kl}|.$$

Moreover, since concave minimization problems must be solved in every node created at least once for every facility in the branch and price algorithm, run times are considerably longer compared to pure column generation. In order to start with a promising column set, in branching, the columns that are not violating the restrictions are imported to child nodes' column set. In addition to this, similar to the pure column generation part, in order to decrease the running time, every column created with negative reduced cost are added to the model instead of adding the most negative one in every iteration. On top of all this given above, the previously defined column generation heuristic is altered and used. Overall column generation with branch and price heuristic (BPH) can be seen in Figure 7.6. Similar to the column generation, in Step 3, every column created is added to the column sets and this change improves run time for some cases.

```
1. Set $\mathcal{J}_s^{\text{prev}} \leftarrow \mathcal{J}_s, \mathcal{J}_s \leftarrow \emptyset$
2. For $j = 1, ..., |\mathcal{N}|$, if $\alpha_i^{(t)\prime}(\mathbf{x}_i, n_j) \leq 0$ then $\mathcal{J}_s \leftarrow \mathcal{J}_s \cup \{n_j\}$
   For $j = 1, ..., |\mathcal{Z}|$, if $\beta_i^{(t)\prime}(\mathbf{x}_i, z_{j1}, zj2) \leq 0$ then $\mathcal{J}_s \leftarrow \mathcal{J}_s \cup \{z_{j1}, z_{j2}\}$
   For $j = 1, ..., |\mathcal{O}|$, if $\delta_i^{(t)\prime}(\mathbf{x}_i, o_{j1}, o_{j2}) = \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j1})$ then $\mathcal{J}_s \leftarrow \mathcal{J}_s \cup \{o_{j1}\}$
                    else if $\delta_i^{(t)\prime}(\mathbf{x}_i, o_{j1}, o_{j2}) = \gamma_i^{(t)\prime}(\mathbf{x}_i, o_{j2})$ then $\mathcal{J}_s \leftarrow \mathcal{J}_s \cup \{o_{j2}\}$
   For $k = 1, ..., |\mathcal{S}|\,; l = 1, ..., |\mathbf{S}_k|\,; j = 1, ..., |\mathcal{S}_{kl}|$,
                    if $\mu_i^{(t)}(\mathbf{x}_i, \mathcal{S}_k) = \nu_i(\mathbf{x}_i, \mathcal{S}_{kl})$ then $\mathcal{J}_s \leftarrow \mathcal{J}_s \cup \mathcal{S}_{kl}$
3. Find $c_i^{(t)} = \min_{\mathbf{x}_i} \left\{ \sum_{j \in \mathcal{J}_s} \gamma_i(\mathbf{x}_i, j) \right\}$
4. If $\mathcal{J}_s^{\text{prev}} = \mathcal{J}_s$ then go to Step 5, else go to Step 1
5. Add newly created column containing $\mathcal{J}_s$ and facility $i$ and solve model. If the
   objective function does not improve terminate, else go to Step 6
6. Update $u^{(t)}$, $t \leftarrow t + 1$
```

Figure 7.6. Column generation with branch and price heuristic

After defining the BPH in Figure 7.6, the branch and price algorithm BP$_2$ can be given as Figure 7.7.

```
1. Set $t \leftarrow 0$, $\mathcal{B} \leftarrow \emptyset$ and $Z_{\text{LB}} \leftarrow +\infty$
2. Let $\mathcal{M}_0 \leftarrow \{\emptyset, \emptyset\}$ and $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathcal{M}_0\}$
3. Repeat (node selection)
    (a) Choose $\mathcal{M}_l$ such that $\mathcal{M}_l \in \mathcal{B}$, $\mathcal{B} \leftarrow \mathcal{B} \setminus \mathcal{M}_l$
    (b) Find $\mathcal{N}, \mathcal{Z}, \mathcal{O}$ and $\mathcal{S}$ sets by definitions given above for the node $M_l$
    (c) Initialize the RCMFWP$_5$ model with a feasible solution
    (d) Repeat (solving the subproblem of the selected node)
        i. $t \leftarrow t + 1$ and $\overline{c}^* \leftarrow +\infty$
        ii. Solve the main problem and find the dual values $u_i^{(t)}$, $v_j^{(t)}$ and $w^{(t)}$
        iii. For each facility $i$
            A. Solve the BP$^i$ problem for facility $i$
            B. If $\overline{c}_i^* \leq \overline{c}^*$ then $\overline{c}^* \leftarrow \overline{c}_i^* \leq \overline{c}^*$
        iv. If $\overline{c}_i^* < 0$, then
            A. Insert column for facility $i$ which is found using (7.62)
            B. Solve the main problem and find the dual values $u_i^{(t)}$, $v_j^{(t)}$ and $w^{(t)}$
            C. Run BPH for facility $i$ from the locations found in subproblem BP$^i$
        v. Until $\overline{c}^* \geq 0$
    (e) Set $Z_l \leftarrow$ the optimal solution for node $l$
    (f) If the solution is not binary and $Z_{\text{LB}} < Z_l$ then choose a branching pair
        $\{j_1, j_2\}$, form two new nodes $\mathcal{M}_{l(\text{left})}$ and $\mathcal{M}_{l(\text{right})}$:
        i. $\mathcal{M}_{l(\text{left})} \leftarrow \left\{\mathcal{M}^0_{l(\text{left})}, M^1_l\right\}$ where $\mathcal{M}^0_{l(\text{left})} \leftarrow M^0_l \cup \{j_1, j_2\}$
        ii. $\mathcal{M}_{l(\text{right})} \leftarrow \left\{\mathcal{M}^0_l, \mathcal{M}^1_{l(\text{right})}\right\}$ where $\mathcal{M}^1_{l(\text{right})} \leftarrow M^1_l \cup \{j_1, j_2\}$
        iii. $Z_{l(\text{left})} \leftarrow Z_l$, $Z_{l(\text{right})} \leftarrow Z_l$ and $\mathcal{B} \leftarrow \mathcal{B} \cup \left\{\mathcal{M}_{l(\text{left})}, \mathcal{M}_{l(\text{right})}\right\}$
        iv. Export every feasible columns from $\mathcal{M}_l$ to $\mathcal{M}_{l(\text{left})}$ and $\mathcal{M}_{l(\text{right})}$
    (g) If the solution is binary and $Z_{\text{LB}} < Z_l$ then $Z_{\text{LB}} \leftarrow Z_l$ and remove nodes
        with lower bound less than $Z_{\text{LB}}$ from $\mathcal{B}$
    (h) Until $\mathcal{B} = \emptyset$
```

Figure 7.7. Branch and price algorithm for the RCMFWP$_5$

7.5. Outer Approximation

One of the basic properties of the concave minimization problem on a convex feasible set is that a global optimal point exists at some extreme point of the feasible set. If the feasible set is a polytope, the number of extreme points are finite and the problem can be solved by the enumeration of these vertices. Nevertheless, they can be too many [33] and an efficient method is necessary since it effects the computational cost of pricing and thus the efficiency of the column generation.

Another approach is to find optimal solutions for the concave minimization problems defined on a convex feasible set is the outer approximation method. Simply outer approximation method can be defined as relaxing the feasible set to a simpler including set, finding the optimal point on this relaxed set, adding new constraints on this relaxed set to exclude the optimal point which is not a feasible point for the original problem until a feasible solution –or at least a feasible solution which is almost feasible– is found. The outer approximation algorithm starts by creating a relaxed feasible set for the feasible set of the original problem. Since the extreme points of the relaxed feasible region are the candidate locations for the global optimal solution, they are kept on a set and in every iteration the extreme point with the maximum objective function value is selected and removed from the set. If this point is at most ϵ-infeasible or simply an ϵ-feasible solution for the main problem, this point is regarded as a global optimal point of the concave minimization problem. If not, a cut is added to the relaxed feasible solution which excludes this extreme point but not any part of the feasible set of the original problem. After this cut, the set of extreme points are updated and the process continues until the stopping criteria is obtained.

Before defining the outer approximation algorithm, let us define the adjacency list algorithm [34]. It is an efficient way to keep track of the extreme points set in outer approximation method. Current vertex set is defined as $\mathcal{V}$. $\mathcal{N}(v)$ is the new neighbors of vertex v. $\mathcal{G}$ is the set of functions of cuts that are added previously and $g^{\text{new}}(.) \leq 0$ is the function of new cut added to the relaxed set. $\mathcal{P}$ is the hyperplane which can be defined as $\mathcal{P} = \{x \in \mathbb{R}^n : g(x) = 0\}$ and $\mathcal{J}(v) = \{g() \in \mathcal{G} : g(v) = 0\}$ is the set of constraints that defines vertex v. After defining the variables, the adjacency list algorithm can be listed as Figure 7.8.

```
1. Set $\mathcal{V}^- \leftarrow \{\}$ and $\mathcal{V}^+ \leftarrow \{\}$
2. For all vertex $v \in \mathcal{V}$
   (a) If $g(v) \leq 0$, then $\mathcal{V}^- \leftarrow \mathcal{V}^- \cup \{v\}$ else $\mathcal{V}^+ \leftarrow \mathcal{V}^+ \cup \{v\}$
3. Set $\mathcal{V}^{\text{new}} \leftarrow \{\}$
4. For all vertex $v^- \in \mathcal{V}^-$
   (a) For all vertex $v^+ \in \mathcal{N}(v^-) \cap \mathcal{V}^+$
       i. Set $w \leftarrow [v^-, v^+] \cap \mathcal{P}$
       ii. $\mathcal{V}^{\text{new}} \leftarrow \mathcal{V}^{\text{new}} \cup w$
       iii. $\mathcal{N}(v^-) = \{\mathcal{N}(v^-) \setminus \{v^+\}\} \cup \{w\}$
       iv. $\mathcal{N}(w) \leftarrow \{v^-\}$ and $\mathcal{J}(w) \leftarrow \{\mathcal{J}(v^-) \cap \mathcal{J}(v^+)\} \cup g^{\text{new}}()$
5. For all vertex $u \in \mathcal{V}^{\text{new}}$
   (a) For all vertex $v \in \mathcal{V}^{\text{new}}$ and $v \neq u$
       i. If $|\mathcal{J}(u) \cap \mathcal{J}(v)| = n-1$, then $\mathcal{N}(u) \leftarrow \mathcal{N}(u) \cup v$ and $\mathcal{N}(v) \leftarrow \mathcal{N}(v) \cup u$
```

Figure 7.8. Adjacency list algorithm

For the problem,

$$\min f(x) \tag{7.63}$$

$$\text{s.t. } x \in \mathcal{D} \tag{7.64}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuous and $\mathcal{D} \subset \mathbb{R}^n$ is closed, the generic outer approximation scheme is summarized in Figure 7.9 as given in [31].

It is worth to highlight some application details we use when implementing the outer approximation method. In Step 1, initial relaxation $\mathcal{D}^1$ is defined as a hyperplane using the upper bounds on the variables. The bounds that are computable or given explicitly are used as they are. For the unbounded variables, arbitrary large numbers are selected as upper bounds. This approach also simplifies finding the initial vertex set

```
1. Set $t \leftarrow 1$ and choose $\mathcal{D}^1 \supset \mathcal{D}$
2. Initialize $\mathcal{V}$ vertex set with the vertices of $\mathcal{D}^1$
3. Repeat
   (a) $\mathbf{v}^* \leftarrow \arg\min_{\mathbf{v}\in\mathcal{V}} \{f(\mathbf{v})\}$
   (b) If $\mathbf{v}^* \notin \mathcal{D}$, then
       i. Construct a constraint function $l^{(t)} : \mathcal{R}^n \rightarrow \mathcal{R}$ satisfying,
          A. $l^{(t)}(\mathbf{v}) \leq 0$ for all $\mathbf{v} \in \mathcal{D}$
          B. $l^{(t)}(\mathbf{v}^*) > 0$
       ii. Set $\mathcal{D}^{(t+1)} \leftarrow \mathcal{D}^{(t)} \cap \left\{\mathbf{v} : l^{(t)}(\mathbf{v}) \leq 0\right\}$
       iii. Set $g^{\text{new}} \leftarrow l^{(t)}$ and update $\mathcal{V}$ by the adjacency list algorithm
       iv. $t \leftarrow t + 1$
   (c) Until $\mathbf{v}^* \in \mathcal{D}$
```

Figure 7.9. Outer approximation algorithm

$\mathcal{V}$. During the initialization we also found a feasible point $\mathbf{w} \in \mathcal{D}$ to use in further steps. In function construction given in Step 3(b)i, we first draw a line between the feasible point $\mathbf{w}$ and current infeasible point $\mathbf{v}^*$. The intersection of this point and the most infeasible constraint is found. If the constraints of the set $\mathcal{D}$ are differentiable, partial derivatives of these functions are used, if not the partial derivatives are calculated from the formal definition of the derivative for the intersection point. After finding the point and the partial derivatives on that point, a hyperplane which contains that point with the slopes of partial derivatives is drawn and checked by using the feasible point $\mathbf{w}$. That is how we construct function $l^{(t)}$ at iteration (t).

7.6. Alternating Location Allocation Heuristic

ALA heuristic was proposed by Cooper to solve the UMFWP [7]. It simply consists of the sequential solution of location and allocation problems. With its fast convergence rate and near optimal solution, it is still one of the best heuristics available in the literature. Similar to CMFWP, UMFWP is an NP-complete problem, but the two components of the heuristic, location and allocation problems are easy to solve.

Allocation part in this heuristic is a simple inspection over customers in which they are assigned to the nearest facility. Location part is similar to CALA and can be handled by using Weiszfeld procedure given in Figure 5.1. The steps of the ALA heuristic are given in Figure 7.10.

```
1. Define initial facility locations $x_i$ for $i = 1, ..., m$
2. Set facility locations $\overline{x}_i$ for $i = 1, ..., m$ as parameters.
3. For each customer $j$, if facility $i$ is the nearest facility to customer $j$ set $w_{ij} \leftarrow h_j$,
   otherwise set $w_{ij} \leftarrow 0$
4. Set allocations $\overline{w}_{ij}$ for $i = 1, ..., m; j = 1, ..., n$ as parameters and solve a SFWP
   for every facility to find the locations $x_i$ for $i = 1, ..., m$
5. Go to Step 2 until termination criteria are satisfied
```

Figure 7.10. Alternating location allocation heuristic

Notice that in our case the distance function is $\overline{d}_i\left(\mathbf{x}_i, \mathbf{a}_j\right) = c_{ij} d\left(\mathbf{x}_i, \mathbf{a}_j\right) + \lambda_i$. Here λ_i is the Lagrange multiplier and because of it, the new distance function depends on the facility. In short, Lagrangean subproblems are UMFWPs with facility dependent distance functions. This affects the allocation phase: The distance function $\overline{d}_i\left(\mathbf{x}_i, \mathbf{a}_j\right)$ is used. However, Weiszfeld procedure can be adopted directly since λ_i is constant and does not exist, after the differentiations with respect to the facility coordinates.

7.7. Uncapacitated Discrete Approximation Heuristic

Even though it is one of the most efficient heuristics in the literature, ALA's accuracy heavily depends on the initial solution. In order to calculate better solutions with ALA, we have tried to improve initial facility locations. This is done by solving a discrete uncapacitated multifacility location allocation problem (DUMLAP) to find initial facility locations whose pure binary mathematical model can be written as:

DUMLAP:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{r} y_{ijk} h_j \left[c_{ij} d\left(\mathbf{b}_k, \mathbf{a}_j\right) + \lambda_i \right] \tag{7.65}$$

$$\text{s.t.} \sum_{i=1}^{m} \sum_{k=1}^{r} y_{ijk} = 1 \qquad j = 1, ..., n \tag{7.66}$$

$$\sum_{k=1}^{r} x_{ik} = 1 \qquad i = 1, ..., m \tag{7.67}$$

$$y_{ijk} \leq x_{ik} \qquad i = 1, ..., m; j = 1, ..., n; k = 1, ..., r \tag{7.68}$$

$$y_{ijk} \in \{0, 1\} \qquad i = 1, ..., m; j = 1, ..., n; k = 1, ..., r \tag{7.69}$$

$$x_{ik} \in \{0, 1\} \qquad i = 1, ..., m; k = 1, ..., r \tag{7.70}$$

Similar to the variables defined in the previous models, $\mathbf{b}_k$ is the candidate location k for facilities and they are taken as the customer locations. In other words $\mathcal{K} = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_m\}$. x_{ik} is 1 if facility i is opened at location $\mathbf{b}_k$ and y_{ijk} is 1 if facility i is opened at location $\mathbf{b}_k$ serves customer j. Constraints (7.66) force every customer to be assigned to a facility, 0 otherwise. Constraints (7.67) restrict facilities to be opened in only one candidate location. Constraints (7.68) guarantee that customers are served by opened facilities. As mentioned above, the solution of this model is taken as initial facility locations and ALA is run afterwards.

7.8. Relaxed Uncapacitated Discrete Approximation Heuristic

For larger problems, it is difficult to solve the pure binary integer programming model. Fortunately, an accurate Lagrangean heuristic can be devised. Assignment constraints (7.66) are relaxed and the Lagrangean subproblem is formulated as

RDUMLAP:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{r} y_{ijk} \left\{ h_j \left[c_{ij} d\left(\mathbf{b}_k, \mathbf{a}_j\right) + \lambda_i \right] + \mu_j \right\} - \sum_{j=1}^{n} \mu_j \tag{7.71}$$

$$\text{s.t. Constraints (7.67) - (7.70)} \tag{7.72}$$

where μ_j stands for the corresponding Lagrange multipliers.

As it can be seen easily, the second summation is constant and the first summation is separable over the facilities, and the solution of RDUMLAP becomes equivalent to the solution of

RDUMLAP^i:

$$\min \sum_{j=1}^{n} \sum_{k=1}^{r} y_{ijk} \left\{ h_j \left[c_{ij} d\left(\mathbf{b}_k, \mathbf{a}_j\right) + \lambda_i \right] + \mu_j \right\} \tag{7.73}$$

$$\text{s.t.} \sum_{k=1}^{r} x_{ik} = 1 \tag{7.74}$$

$$y_{ijk} \leq x_{ik} \qquad j = 1, ..., n; k = 1, ..., r \tag{7.75}$$

$$y_{ijk} \in \{0, 1\} \qquad j = 1, ..., n; k = 1, ..., r \tag{7.76}$$

$$x_{ik} \in \{0, 1\} \qquad k = 1, ..., r \tag{7.77}$$

Furthermore, the separable problems, RDUMLAP^i, can be solved by inspection. The inspection consists of a calculating the cost of assigning a single facility to every candidate location for given customer subset and selecting the candidate location which gives the minimum cost.

In addition to the discrete alternating location allocation heuristic (DALA) is used in steps of the the relaxed discrete approximation heuristic (RUDAH). Similar to ALA, in DALA heuristic, location and allocation problems are solved alternately. The only difference is that, in the location phase of DALA, a DLAP is solved for every facility and allocations assigned to it in the previous iteration. The procedure for DALA heuristic can be seen in Figure 7.11.

```
1. Define initial facility locations from the candidate location set $\mathcal{K}$
2. Set facility locations $\overline{x}_i$ for $i = 1, ..., m$ as parameters
3. For each customer $j$, if facility $i$ is the nearest facility to customer $j$ set $w_{ij} \leftarrow h_j$,
   otherwise set $w_{ij} \leftarrow 0$
4. For each facility $i$ do
   (a) $k_i^* \leftarrow \arg\min_{\mathbf{b}_k in \mathcal{K}} \left\{ \sum_j c_{ij} \overline{w}_{ij} d(\mathbf{b}_k, \mathbf{a}_j) \right\}$
   (b) $x_i \leftarrow b_{k_i^*}$
5. Go to Step 2 until termination criteria are satisfied
```

Figure 7.11. Discrete alternating location allocation heuristic

Finally the steps of the overall SO procedure for RUDAH can be listed as Figure 7.12.

```
1. Decide $\mu_j$ and $\pi$ (where $0 \leq \pi \leq 2$)
2. Run DALA heuristic, $C_{\mathrm{UB}}$ is the objective
3. $Z_{\mathrm{UB}} \leftarrow \min(Z_{\mathrm{UB}}, C_{\mathrm{UB}})$
4. Solve seperated problems, $k_i^\star$ is the optimal candidate location for facility $i$
5. Calculate $C_{\mathrm{LB}}$ and set $Z_{\mathrm{LB}} \leftarrow \max(Z_{\mathrm{LB}}, C_{\mathrm{LB}})$
6. Calculate objective by solving ALA. Keep the solution if it is the best so far
7. $G_j = \sum_{i=m}^{n} y_{ijk_i^\star} - 1$        $j = 1, ..., n$
8. $T = \frac{\pi(Z_{\mathrm{UB}} - C_{\mathrm{LB}})}{\sum_{j=n}^{m} G_j^2}$
9. $\mu_j = \mu_j + TG_j$        $j = 1, ..., n$
10. Update $\pi$ if needed
11. Go to Step 2 until termination criteria are satisfied
```

Figure 7.12. Relaxed uncapacitated discrete approximation heuristic

8. COMPUTATIONAL RESULTS

Before giving the computational results, we would like to express the computation environment. All the methods in this research are coded by C#. For solving mixed integer and linear programming problems ILOG Cplex version 11.0 commercial solver with Concert technology is used. The remaining methods, including the outer approximation method based concave minimization solver [31] are coded within the same environment. Results are obtained on a Dell server with two 3.16 Ghz Intel Xeon X5460 processor and 28 GB of RAM. It is worth to note that all runs are realized as single threaded programs.

This chapter consists of three sections. The first one contains results with the alternating location allocation and discrete approximation heuristics. They give upper bounds on the optimal value. In the second section the performances of the approximations using l_1 and l_∞ norms are studied. In the third and final section, we compared the Lagrangean relaxation of the CMFWP with the other lower and upper bounding methods.

Test problems are combined in three groups. Small instances which include some previously solved problems for the Euclidean distance functions, 201-220 which has 5 facilities with 30 customers and 10 facilities with 10 customers as the largest instances. Problems 301-323 are also small instances ranging up to size of 30 customers with 10 facilities and 50 customers with 10 facilities. The problems between 301-323 are unweighted, in other words, the unit shipment cost per unit distance per unit amount of goods (c_{ij}) is defined as unity for every facility customer pair. Test problems 401-423 are obtained by randomly setting c_{ij} of the test problems 301-323 to values different than one. Problems 324-328 and 424-428 are medium instances with 5 facilities and 100 customers where the formers have equal unit shipment cost for every facility customer pair. Large instances have at least 25 facilities and 250 customers which are named as Problems 501-504 and 601-604. The former ones have, unit shipment cost set to one.

The experiments with every test instance are repeated for 5 different distance functions: Euclidean, squared Euclidean, and l_p distance for $p = 1.25$, 1.50 and 1.75.

8.1. Upper Bounds on the Optimal Value

In the upper bounding algorithms, we tested 6 heuristics. The first 4 heuristics are the region rejection heuristics RRH, DRRH, RRH′ and DRRH′. In these heuristics, the radius update factor α is set to 0.7, T is set to the number of customers and E is set to the integer part of number of customers divided by number of facilities. γ is chosen as 0.5 and R, the initial rejection radius, is calculated according to the formula

$$R = \frac{\gamma}{m} \max_j \{\mathbf{a}_{j1}, \mathbf{a}_{j2}\}$$

as suggested in [27]. These four region rejection heuristics are run K times and the best value of these K iterations is reported. The value of K is set to

$$K = \begin{cases} \max\{100, m\} \\ \max\{100, \sqrt[3]{n}\}, \end{cases}$$

which is again suggested in [27].

The remaining two upper bounding heuristics are DAH and RDAH. In RDAH algorithm, initial λ values and π are set to two and π is updated if Z_{UB} or Z_{LB} do not change, by more than 10^{-4} in 20 consecutive iterations. The algorithm terminates if π is updated 15 times or the difference between Z_{UB} and Z_{LB} is less than 10^{-4}.

All individual running times of these six heuristics are limited to 1200 seconds for small instances, 2400 seconds for medium instances and 3600 seconds for large instances, and the current best solution is reported.

Table 8.1. Accuracy of the upper bounds: Percent deviations for Problems 201-220 with the Euclidean distance

Problem	(m, n)	Optimal	RRH	DRRH	RRH′	DRRH′	DAH	RDAH
201	(2,2)	0	0.00	0.00	0.00	0.00	0.00	0.00
202	(2,4)	247.28	0.00	0.00	0.00	0.00	0.00	0.00
203	(2,4)	214.34	0.01	0.01	0.01	0.01	0.01	0.01
204	(3,5)	24	0.00	0.00	0.00	0.00	0.00	0.00
205	(3,5)	73.96	0.00	0.00	0.00	0.00	0.00	0.00
206	(3,9)	221.4	0.06	0.00	28.91	0.00	0.00	0.00
207	(3,9)	871.62	0.00	0.00	0.00	0.00	0.00	0.00
208	(4,8)	609.23	19.79	3.52	19.79	0.00	0.00	0.00
209	(5,15)	8169.8	63.74	0.56	0.56	0.00	0.00	0.56
210	(5,20)	12847	4.13	0.00	4.13	0.37	0.00	0.60
211	(5,20)	1107.2	29.40	21.98	29.40	21.98	0.00	0.00
212	(5,30)	23990	2.83	2.80	2.83	1.63	0.00	2.83
215	(5,10)	2595.5	31.65	23.53	31.65	19.38	0.00	2.32
216	(6,10)	7797.2	3.18	0.00	3.18	3.18	0.00	1.56
217	(7,10)	6967.9	5.05	1.03	18.67	2.51	0.10	0.77
218	(8,10)	1564.5	36.80	23.85	36.80	36.80	0.00	0.00
219	(9,10)	3250.7	20.32	8.88	47.42	13.20	0.00	20.32
220	(10,10)	7719	12.71	1.43	12.71	4.96	0.01	3.77
Average			**12.76**	**4.87**	**13.11**	**5.78**	**0.01**	**1.82**

In Table 8.1, the percent deviations of the upper bounds from optimal solutions of problems 201-220 with the Euclidean distance are reported. The total CPU statistic for these runs can be found in Table 8.2.

As can be seen, DAH heuristic has the highest accuracy and efficiency, which is followed by RDAH. Furthermore, DAH finds the optimal solution on 15 out of 18 test instances. This number is 9 out of 18 for RDAH. The performances of DRRH and DRRH′ are slightly worse; they found an optimal solutions of 8 out of 18 instances.

Table 8.2. Efficiency of the upper bounds: CPU times (seconds) of UB algorithms for Problems 201-220 with the Euclidean distance

Problem	(m,n)	RRH	DRRH	RRH$'$	DRRH$'$	DAH	RDAH
201	(2,2)	0.22	0.16	0.28	0.31	0.39	0.19
202	(2,4)	6.36	8.27	6.45	8.22	0.17	0.20
203	(2,4)	8.11	9.05	9.67	8.61	0.22	0.22
204	(3,5)	11.30	9.22	9.64	8.09	0.25	0.34
205	(3,5)	8.00	9.97	7.80	10.03	0.19	0.23
206	(3,9)	8.89	9.69	10.20	10.00	0.20	0.42
207	(3,9)	14.48	12.44	15.48	12.44	0.19	0.22
208	(4,8)	17.16	18.02	22.33	19.11	0.20	0.83
209	(5,15)	30.45	36.48	31.97	35.78	0.34	1.00
210	(5,20)	30.86	54.27	34.81	43.36	0.44	0.73
211	(5,20)	28.42	32.83	26.06	28.45	0.69	1.13
212	(5,30)	62.64	65.19	60.67	67.77	1.92	2.38
215	(5,10)	15.98	21.05	17.86	27.45	0.28	1.19
216	(6,10)	31.88	34.59	30.31	34.02	0.58	1.03
217	(7,10)	27.59	38.63	31.23	41.86	0.56	1.03
218	(8,10)	19.78	23.06	23.58	24.63	0.55	0.86
219	(9,10)	23.84	26.78	24.22	28.11	0.66	0.84
220	(10,10)	33.00	36.69	31.55	42.19	0.36	1.02
Average		**21.05**	**24.80**	**21.90**	**25.02**	**0.45**	**0.77**

For the rest of the instances, percent deviations from the best known values are reported since the optimal values are not known. Average deviations and CPU times are reported in Table 8.3 and Table 8.4 for the Euclidean distance and l_p distance with $p = 1.25, 1.50, 1.75$. Results for the squared Euclidean distance can be found in Table 8.5 and Table 8.6.

Similar to instances 201-220, DAH has the best performance on the small instances. However, it loses its superiority for the large ones because of the limitations over the CPU time. All of the medium and large instances, except Problem 425, are stopped because of the time limitation. On the other hand, even though it has the minimum average CPU time compared to other methods, RDAH performs the best for all the medium and large instances. Remaining four alternating location allocation heuristics are not only less accurate but also less efficient than RDAH.

As a result, we can state that DAH is the most accurate for the instances that can be solved optimally in time limitations, whereas RDAH has the highest accuracy for the large instances and has fairly short CPU times. It seems better to use RDAH for the cases with more than 100 customers and/or more than 25 facilities. DAH can solve small instances better than the other five heuristics in slightly less CPU seconds.

8.2. Lower Bounds on the Optimal Value

In Chapter 6, in addition to heuristics that compute upper bounds, four lower bounding approaches using discrete approximations of the l_1 and l_∞ norms with special sets of candidate locations are defined. The first two of them, namely L_1 and L_∞, solves the MILP problems obtained for the l_1 and the l_∞ norms exactly whereas the other two, RL_1 and RL_∞, are two Lagrangean heuristics for these two exact models obtained by relaxing the demand constraints. In this section, we are going to compare these four lower bounding algorithms experimentally.

In the L_1 and L_∞, DLAP are solved, whereas in the RL_1 and the RL_∞, Lagrangean relaxations of these DLAP are formed and solved by using the relaxed discrete approx-

Table 8.3. Accuracy of the upper bounds: Average percent deviations for all problems with the Euclidean and l_p distances with $p = 1.25$, 1.50, 1.75

Distance type	Problems	RRH	DRRH	RRH$'$	DRRH$'$	DAH	RDAH
Euclidean	201-220	12.76	4.87	13.11	5.78	0.01	1.82
	301-323	8.48	9.87	9.35	10.03	0.00	1.56
	401-423	33.83	33.23	34.98	33.22	0.00	2.90
	324-328	0.39	0.67	0.06	0.33	0.25	0.00
	424-428	2.50	3.13	0.70	2.36	0.01	0.08
	501-504	9.95	10.01	9.45	9.17	9.13	0.00
	601-604	11.71	18.09	9.24	15.82	6.07	0.41
$p = 1.25$	201-220	10.13	4.56	12.12	6.14	0.08	2.28
	301-323	9.51	9.61	9.01	9.53	0.09	1.58
	401-423	34.24	33.41	32.71	32.70	0.00	2.22
	324-328	0.29	0.54	0.12	0.73	0.55	0.12
	424-428	2.16	2.77	2.17	2.61	0.02	0.19
	501-504	8.52	9.10	10.07	8.95	4.69	0.00
	601-604	9.01	16.75	9.27	15.37	11.93	0.00
$p = 1.5$	201-220	9.68	4.60	14.06	6.14	0.00	2.06
	301-323	8.61	9.69	9.41	10.36	0.01	1.64
	401-423	35.61	32.22	34.38	34.93	0.00	1.91
	324-328	0.13	0.55	0.20	0.65	0.28	0.45
	424-428	3.95	3.40	2.39	2.34	0.13	0.12
	501-504	7.89	8.79	8.45	7.87	6.13	0.00
	601-604	11.07	16.80	10.02	15.35	10.05	0.00
$p = 1.75$	201-220	9.27	4.82	13.26	5.97	0.00	1.83
	301-323	21.09	13.21	12.85	13.49	3.18	4.51
	401-423	36.04	32.18	34.39	35.45	0.00	1.95
	324-328	0.35	0.37	0.11	0.47	0.15	0.00
	424-428	3.98	3.45	2.32	2.71	0.20	0.12
	501-504	8.78	9.65	8.98	9.90	8.20	0.00
	601-604	9.02	17.09	9.66	15.56	5.74	0.00

Table 8.4. Efficiency of the upper bounds: Average CPU times (seconds) for all problems with the Euclidean and l_p distances with $p = 1.25$, 1.50, 1.75

Distance type	Problems	RRH	DRRH	RRH$'$	DRRH$'$	DAH	RDAH
Euclidean	201-220	21.1	24.8	21.9	25.0	0.5	0.7
	301-323	117.3	115.8	120.0	115.3	19.1	3.6
	401-423	79.1	74.5	76.0	73.3	5.3	2.0
	324-328	218.6	225.4	224.2	230.0	2400.7	19.6
	424-428	269.2	266.0	273.7	274.8	2216.5	14.8
	501-504	4091.8	3536.6	3404.3	3456.8	3624.3	1076.8
	601-604	4256.0	3533.7	3356.2	3422.9	3623.2	623.2
$p = 1.25$	201-220	23.6	24.9	23.2	25.0	0.4	0.7
	301-323	117.1	115.9	119.4	119.1	16.3	4.2
	401-423	74.2	72.6	72.1	70.9	5.7	2.2
	324-328	226.9	232.3	225.9	241.5	2402.9	23.3
	424-428	280.1	273.4	281.2	282.9	2223.2	27.1
	501-504	3358.9	3540.4	3361.7	3455.2	3620.5	1151.0
	601-604	3386.4	3447.3	3343.6	3431.4	3621.3	722.3
$p = 1.5$	201-220	21.6	23.3	21.4	24.1	0.4	0.7
	301-323	119.0	117.9	116.7	115.3	21.1	4.1
	401-423	77.2	71.5	80.9	72.4	5.7	2.1
	324-328	215.7	220.4	222.7	225.6	2400.6	18.2
	424-428	277.8	265.5	281.4	284.1	2289.4	16.6
	501-504	3352.9	3541.4	3331.2	3501.9	3621.4	1033.1
	601-604	3378.8	3474.5	3321.1	3486.0	3619.4	651.9
$p = 1.75$	201-220	21.6	23.7	21.0	24.3	0.4	0.7
	301-323	121.1	118.4	118.2	115.6	14.7	4.2
	401-423	78.2	72.0	80.1	73.4	5.4	2.1
	324-328	216.1	223.6	217.5	220.1	2405.1	21.0
	424-428	262.6	261.9	275.9	278.0	2162.7	12.9
	501-504	3378.7	3503.1	3307.1	3487.6	3622.8	1118.4
	601-604	3372.2	3480.1	3321.9	3419.6	3620.9	645.0

Table 8.5. Accuracy of the upper bounds: Average percent deviations for all problems with the squared Euclidean distance

Distance type	Problems	RRH	DRRH	RRH′	DRRH′	DAH	RDAH
Squared Euclidean	201-220	10.70	4.61	10.15	4.51	1.53	4.47
	301-323	15.67	16.49	18.31	13.81	1.12	4.31
	401-423	56.94	53.84	54.13	62.86	1.14	11.38
	324-328	0.89	0.83	1.54	1.50	1.00	0.00
	424-428	6.36	8.56	6.26	7.31	2.82	0.00
	501-504	6.88	7.36	9.55	9.14	9.50	0.92
	601-604	26.40	28.75	26.38	32.43	14.10	0.23

Table 8.6. Efficiency of the upper bounds: Average CPU times (seconds) for all problems with the squared Euclidean distance

Distance type	Problems	RRH	DRRH	RRH′	DRRH′	DAH	RDAH
Squared Euclidean	201-220	9.3	9.4	9.0	9.1	0.5	0.9
	301-323	24.0	24.3	25.1	24.5	109.1	4.7
	401-423	23.8	23.7	23.0	22.9	8.7	2.7
	324-328	76.8	77.2	78.5	76.6	2418.2	31.1
	424-428	74.5	76.4	72.8	73.4	2400.9	21.9
	501-504	2867.7	2831.8	2642.4	2851.9	3623.5	588.0
	601-604	2811.2	2769.6	2683.1	2759.1	3621.7	1976.9

imation heuristic given in Figure 6.4 without Step 4, using an appropriate candidate location set $\mathcal{K}$. Initial μ and π values are set to 2 and π is updated if Z_{UB} or Z_{LB} do not change more than 10^{-4} in 20 consecutive iterations. If π is updated 15 times, the difference between Z_{UB} and Z_{LB} is less than 10^{-4} or the CPU time exceeds the time limit algorithm terminates and Z_{LB} is reported as the lower bound after multiplying it with the constant defined in (6.10) and (6.11). Similar to the previous runs, all individual CPU times of these four heuristics are limited to be below 1200 seconds for small instances, 2400 seconds for medium instances and 3600 seconds for large instances. The percent relative deviations of the solutions of the lower bounding algorithms from the optimal solutions and the total CPU times for these runs of problems 201-220 with the Euclidean distances are reported in Table 8.7 and 8.8 respectively.

As can be seen, for problems 201-220 with the Euclidean distance, L_∞ algorithm performs slightly better than the other three. None of the instances' but problem 201's lower bounds calculated by these four algorithms overlap with the optimal objective function values. The CPU times for all the instances are very short; but the relaxation type algorithms terminate in less than one second on the average. Whereas the other two heuristics find the lower bound for Problem 212 in more than sixty seconds.

For the remaining instances, again the deviations from the best known are compared instead of the the deviations from the optimal objective value. Average deviations from the best value and the average CPU times for the Euclidean and l_p distance for p = 1.25, 1.50, 1.75 are reported in Table 8.9 and Table 8.10 respectively. First, we must state that large instances are not solved optimally by using the methods L_1 and L_∞, since these models composed of more than 200000 binary decision variables. Second, the CPU times are reported regardless of the distance type. The L_1 and RL_1 only multiplies the value found by the algorithm with a different constant whereas L_∞ and RL_∞ returns the same solution for every distance type for the same instance.

From the results, we can state that the RL_1 algorithm has the best average accuracy, whereas in small instances L_∞ algorithm gives the best lower bounds for the Euclidean distance. Even though they are solved using the Lagrangean heuristic, RL_1

Table 8.7. Accuracy of the lower bounds: Percent deviations for Problems 201-220 with the Euclidean distance

Problem	(m, n)	Optimal	L_1	L_∞	RL_1	RL_∞
201	(2,2)	0	0.00	0.00	0.00	0.00
202	(2,4)	247.28	6.78	8.61	6.78	8.61
203	(2,4)	214.34	18.84	10.42	18.84	10.42
204	(3,5)	24	29.29	0.00	39.90	35.00
205	(3,5)	73.96	6.31	18.88	6.31	18.88
206	(3,9)	221.4	12.17	10.57	13.03	12.16
207	(3,9)	871.62	17.58	3.97	17.58	3.97
208	(4,8)	609.23	7.96	20.72	14.03	24.23
209	(5,15)	8169.79	16.75	10.62	17.33	10.68
210	(5,20)	12846.87	10.14	9.68	11.17	9.79
211	(5,20)	1107.18	11.67	8.60	18.57	12.57
212	(5,30)	23990.04	18.72	11.57	19.83	14.40
215	(5,10)	2595.47	6.64	12.73	8.79	15.01
216	(6,10)	7797.21	13.03	9.02	17.45	14.19
217	(7,10)	6967.9	11.99	7.49	15.13	12.43
218	(8,10)	1564.46	19.59	7.32	30.44	19.72
219	(9,10)	3250.68	20.10	9.50	28.13	21.01
220	(10,10)	7719	5.97	15.74	6.07	15.79
average			**12.97**	**9.75**	**16.08**	**14.38**

Table 8.8. Efficiency of the lower bounds: CPU times (seconds) for problems 201-220 with the Euclidean distance

Problem	(m, n)	L_1	L_∞	RL_1	RL_∞
201	(2,2)	0.27	0.25	2.38	0.20
202	(2,4)	0.19	0.31	0.28	0.20
203	(2,4)	0.19	0.27	0.30	0.19
204	(3,5)	0.22	0.31	0.38	0.47
205	(3,5)	0.19	0.39	0.27	0.20
206	(3,9)	0.42	0.91	0.45	0.55
207	(3,9)	0.34	0.73	0.31	0.23
208	(4,8)	0.66	0.58	0.52	0.45
209	(5,15)	1.28	3.59	0.70	0.72
210	(5,20)	5.02	9.53	1.56	1.52
211	(5,20)	2.67	11.95	1.38	1.84
212	(5,30)	73.03	153.36	3.75	4.33
215	(5,10)	0.73	1.14	0.47	2.94
216	(6,10)	1.02	2.50	0.77	0.70
217	(7,10)	1.02	1.98	0.64	0.75
218	(8,10)	1.00	1.48	0.77	0.81
219	(9,10)	1.27	1.44	0.77	0.89
220	(10,10)	0.77	1.78	0.75	0.77
Average		**5.01**	**10.70**	**0.91**	**0.99**

Table 8.9. Accuracy of the lower bounds: Average percent deviations for all the problems with the Euclidean distance and l_p distances with $p = 1.25, 1.50, 1.75$

Distance type	Problem	L_1	L_∞	RL_1	RL_∞
Euclidean	201-220	12.97	9.75	16.08	14.38
	301-323	12.42	5.51	7.75	2.77
	401-423	10.39	4.99	9.23	6.31
	324-328	84.02	86.31	0.73	0.95
	424-428	97.42	95.48	1.92	1.11
	501-504	N/A	N/A	0.25	6.09
	601-604	N/A	N/A	0.08	0.68
$p = 1.25$	201-220	0.72	14.84	4.40	19.39
	301-323	7.25	18.60	2.62	16.48
	401-423	6.38	19.17	5.22	20.35
	324-328	83.95	88.81	0.00	18.94
	424-428	97.34	96.29	0.00	18.02
	501-504	N/A	N/A	0.00	23.51
	601-604	N/A	N/A	0.00	19.26
$p = 1.5$	201-220	1.55	8.08	5.14	12.86
	301-323	7.70	11.20	3.00	8.79
	401-423	6.90	11.92	5.70	13.15
	324-328	83.95	87.73	0.00	11.10
	424-428	97.34	95.94	0.00	10.08
	501-504	N/A	N/A	0.00	16.11
	601-604	N/A	N/A	0.00	11.44
$p = 1.75$	201-220	3.44	4.30	6.91	9.16
	301-323	9.09	6.64	4.31	3.97
	401-423	8.06	7.17	6.82	8.39
	324-328	83.95	86.89	0.00	5.03
	424-428	97.36	95.66	0.45	4.41
	501-504	N/A	N/A	0.00	10.38
	601-604	N/A	N/A	0.00	5.40

Table 8.10. Efficiency of the lower bounds: Average CPU times (seconds) for all the problems with the Euclidean distance and l_p distances with $p = 1.25$, 1.50, 1.75

Problem	L_1	L_∞	RL_1	RL_∞
201-220	5.01	10.70	0.91	0.99
301-323	714.20	653.77	14.42	11.57
401-423	629.75	578.20	11.28	9.09
324-328	2410.77	2415.98	284.88	329.28
424-428	2410.28	2412.78	207.06	236.42
501-504	N/A	N/A	3609.70	3612.12
601-604	N/A	N/A	3606.52	3610.10

and RL_∞ terminates because of the time limitation, for large instances, since there are more than 200000 candidate facility locations. All other CPU times of these two algorithms are fairly short compared to L_1 and L_∞. L_1 and L_∞ are terminated by using the time limit even for some of the small instances with 25 customers and 9 facilities.

Regardless of the size of the problem and the distance, we can state that RL_1 has the highest accuracy and efficiency. It solves small and medium sized instances in less than 600 seconds and in most of the cases, finds the best solution compared to other methods.

8.3. Bounds with Lagrangean Relaxation

In this section we will show the computational results for the RCMFWP. The RCMFWP is solved by either column generation or branch and price technique. In both of these methods a concave minimization solver with the outer approximation method is used for solving the subproblems. In order to eliminate the asymptotic convergence issue, several stopping conditions are used to terminate concave minimization solver. The number of iterations and the number of extreme points in the outer convex hull are limited to 5000. As mentioned before, since the outer approximation offers asymptotic convergence, an ϵ value of 10^{-4} is decided and the outer approximation method returns an optimal solution which is at most ϵ infeasible of the constraints. In addition, in column generation and branch and price, algorithms given as Figure 7.4 and Figure 7.7 are used instead of Figure 7.2 and Figure 7.5 respectively. Last but not least, columns are generated using strategies given with (7.43) and (7.62) instead of the ones given with (7.42) and (7.61) by setting the parameter ξ to 1.

In column generation an initial starting feasible solution and column set is created by running CALA heuristic 10 times starting at random location at their first run in any SO iteration. Furthermore, in order to decrease the time required to create initial column set, columns created in the previous SO iteration are used in the next iteration after correcting their cost values, since λ_i change in every iteration of SO.

In order to compute the upper bounds, either DAH or RDAH is used at first iteration. At the remaining iterations of SO, CALA heuristic is used and optimal/best facility locations found by previous steps' lower bounding algorithm is used as starting facility locations. In DAH and RDAH, customer locations are used as the candidate locations. For the cases with more than 30 customers, the RDAH is used with CALA heuristic. A similar approach is also used for lower bounding algorithms. We solve small instances with UDAH algorithm and replace with its Lagrangean heuristic RUDAH for the other two groups of instances.

In SO procedure, initial π and λ_i values are set to 2 and updated as shown in Figure 7.1. π is halved if 20 steps passed without any significant improvements which is more than 10^{-4} in the best lower or upper bounds. If π or the difference between the best lower and upper bound is less than 10^{-4} the SO procedure terminates. The runs are limited to 180 minutes for small instances, 360 minutes for the medium instances and 540 minutes for the large ones, SO procedure terminates and the best values found so far are reported. However, if time elapsed inside one of the lower or upper bounding algorithms, program waits the algorithm to terminate.

In the test runs for the problems 201-220 with the Euclidean distance column generation (CGA) and branch and price algorithms (BPA) are used to compute lower bound in every iterations of the SO method. Whereas, upper bound is first initialized by DAH and updated by CALA heuristic in every iteration. In the other two groups of test runs, instead of running the CGA and BPA in every step, UDAH is used in every iteration and the lower bound is updated by the other two with different periods according to run type. Even though a heuristic is used to calculate lower bounds inside SO, the reported lower bound is calculated by one of the valid lower bounding algorithms to guarantee its validity at the final step.

Two different settings are considered according to the usage of the second lower bounding algorithm. In all types, second lower bounding algorithm is applied and π is halved when the difference between upper and lower bound is less than 10^{-4}. In addition to that, in type one runs (1), second lower bounding algorithm is run in every iteration in which π is updated. In type two runs (2), the second lower bounding algorithm is run after SO terminates, with the lambda values which gives the greatest lower bound in iterations. The optimality gaps and the CPU times for these 6 different test runs and the best gap found by using the L_∞ and DAH for the Euclidean distance are reported in Table 8.11 and Table 8.12 respectively.

Before continuing, we must note that, in all of the iterations using Lagrangean heuristic, upper bound is initialized by running a DAH at the first iterations of the SO algorithm. Even though, a CALA heuristic is run in every step of the SO, the solution found by the DAH is rarely improved. This shows the accuracy of the DAH heuristic for small instances.

If we compare the average gaps, we can state that L_∞/DAH pair outperforms all other three. However if the number of optimal solutions found is compared, which means the number of zero gaps, all the other six Lagrangean heuristics found optimal values for the four cases. Furthermore, we can realize that, for the problem 202, the gap calculated by the Lagrangean type heuristics is 100 % which means the lower bound found for this instance is 0. If we treat it as an outlier and recalculate the gaps, the type two algorithm of BPA where UDAH is run in every step of the SO and BPA is run only in π updates and final iterations, have average gap of 8.79 % which is better

Table 8.11. Accuracy of the Lagrangean heuristic: duality gaps for the problems 201-220 with the Euclidean distance

Problem	CGA(1)	BPA(1)	CGA(2)	BPA(2)	CGA	BPA	L_∞/DAH
201	0.00	0.00	0.00	0.00	0.00	0.00	0.00
202	0.00	0.00	0.00	0.00	0.00	0.00	8.60
203	22.90	20.10	22.90	20.10	21.67	20.10	10.43
204	100.00	100.00	100.00	100.00	100.00	100.00	0.00
205	0.00	0.00	0.00	0.00	0.00	0.00	18.87
206	22.37	11.34	22.95	11.34	21.54	11.35	10.57
207	0.00	0.00	0.00	0.00	0.00	0.00	3.97
208	22.68	17.31	22.69	17.47	19.60	17.59	20.72
209	6.30	6.30	7.00	7.00	5.14	4.93	10.62
210	1.58	1.58	1.58	1.58	1.58	1.78	9.68
211	8.28	5.27	8.28	5.27	7.81	5.63	8.60
212	5.56	4.74	5.58	5.12	5.05	4.46	11.57
215	3.72	3.72	3.72	3.72	3.77	3.82	12.73
216	17.37	17.37	17.39	17.39	17.61	18.78	9.02
217	10.54	10.54	10.54	10.54	14.26	13.97	7.58
218	34.25	22.99	34.25	22.99	28.97	37.36	7.32
219	30.05	27.70	30.06	27.63	31.80	65.13	9.50
220	0.46	0.46	0.46	0.46	1.40	1.40	15.75
Average	**15.89**	**13.86**	**15.97**	**13.92**	**15.57**	**17.02**	**9.75**

Table 8.12. Efficiency of the Lagrangean heuristic: CPU times (second) for the problems 201-220 with the Euclidean distance

Problem	CGA(1)	BPA(1)	CGA(2)	BPA(2)	CGA	BPA	L_∞/DAH
201	0.4	0.5	0.4	0.5	0.5	0.3	0.6
202	63.9	63.7	65.0	63.5	156.0	260.0	0.5
203	103.2	466.4	36.2	67.7	790.0	8014.0	0.5
204	417.6	398.5	113.9	110.4	6557.4	6950.7	0.6
205	66.1	63.2	66.0	63.2	92.3	89.8	0.6
206	239.2	501.2	67.2	167.9	343.6	10819.7	1.1
207	192.9	185.0	130.1	125.2	773.4	793.1	0.9
208	771.9	5038.2	177.3	547.9	10802.4	10872.9	0.8
209	2545.1	2255.3	415.9	408.7	10809.7	10916.0	3.9
210	1481.3	3732.4	321.9	322.0	10806.8	11735.7	10.0
211	320.8	5536.0	176.9	768.3	1801.2	10876.3	12.6
212	888.5	1209.4	577.1	747.6	3059.9	10905.5	155.3
215	1907.9	2108.0	375.0	364.2	10836.4	10949.7	1.4
216	3130.0	2995.2	653.3	644.2	10839.0	11086.0	3.1
217	4305.0	4077.8	961.4	931.7	10840.9	10897.7	2.5
218	3669.0	9823.7	754.3	1326.2	10859.6	10983.4	2.0
219	4259.5	14385.8	898.4	3409.2	10823.1	10974.8	2.1
220	5299.6	5359.2	873.3	854.4	10824.8	11127.5	2.1
Average	**1647.9**	**3233.3**	**370.2**	**606.8**	**6167.6**	**8236.3**	**11.2**

than the L_∞/DAH pairs gap. However, if we compare the CPU times, L_∞/DAH pair outperforms the other six with its very short CPU times.

If the Lagrangean heuristics are compared with each other, we can state that using CGA and BPA in every step do not perform better than using them frequently or only at final step. This is because of the limitations over the CPU time. Since in every step of CGA and BPA, at least one concave minimization problem is solved using outer approximation method, the number of iterations done in 180 minutes decreases. Because of this reason, the SO methods using only CGA or BPA as a lower bounding algorithm are not tested for the other cases.

For the rest of the instances, instead of giving the results separately, we have reported the average gaps and running times of each of the groups for every distance. Since RL_1 gives the best performance in both accuracy and efficiency in lower bounding algorithms and DAH performs the best in upper bounding algorithms for small instances and RDAH the best in medium and large ones, the average gaps found by the Lagrangean heuristic are compared with the gap formed by them.

In all runs, if the time limit is reached, the SO steps finish and a final CGA or BPA is run to make lower bound valid. That is why in some of the instances, CPU times are more than the limits. For small instances DAH is used to generate the initial upper bound. Lower bound is found using UDAH with CGA or BPA of usage type (1) or (2). For medium instances, RDAH is used as the initial upper bounding algorithm. As lower bounding algorithm again UDAH with CGA and BPA are used. However, using the UDAH and updating BPA in every update of π, BPA(2) type lower bounding is removed from the runs because of the long running times. For large instances, instead of UDAH, RUDAH is used with CGA. CGA is run only in the last iteration of the test runs to make the found lower bound valid.

The average gaps and CPU times for the Lagrangean heuristic and the gaps and total CPU times of the other best lower and upper bounding algorithms for the Euclidean and l_p distances with p = 1.25, 1.50, 1.75 are reported in Table 8.13 and

Table 8.14.

Based on the overall computational results we can state that, in cases with high $\frac{n}{m}$ ratio, Lagrangean heuristics perform considerably better than the RL_1/(R)DAH in accuracy, whereas in all the other cases, RL_1/(R)DAH slightly gives better results. In addition to that, the RL_1/(R)DAH pair has higher efficiency compared to all other Lagrangean heuristics. Furthermore, the comparisons between the Lagrangean heuristics using BPA and CGA shows that, BPA makes a minor improvement in the lower bounds of some instances with a great increase in the CPU time. In some instances, it is observed that BPA performs worse than CGA. This phenomenon occurs since BPA's long CPU times reduces the number of iterations and this prevents λ_i's to be updated more accurately. In addition, there is not more than 1 % improvement in the objectives of the algorithms using CGA in every π update compared to final iteration only. On the other hand, using CGA at every π update doubles the CPU times. So it is better to use CGA(2).

Table 8.13. Accuracy of the Lagrangean heuristic: Average duality gap for all the problems with the Euclidean, squared Euclidean and l_p distances with $p = 1.25$, 1.50, 1.75

Distance	Problems	CG(1)	BP(1)	CG(2)	BP(2)	RL$_1$/(R)DAH
Euclidean	201-220	15.89	13.86	15.97	13.92	16.08
	301-323	13.26	13.26	13.26	13.27	18.77
	401-423	17.41	16.11	17.53	16.28	19.54
	324-328	3.27	N/A	3.26	3.23	13.43
	424-428	2.34	N/A	3.04	3.04	14.08
	501-504	N/A	25.42	N/A	N/A	21.39
	601-604	N/A	27.60	N/A	N/A	25.85
$p = 1.25$	201-220	15.61	13.41	15.75	13.44	8.56
	301-323	13.12	13.11	13.18	13.16	10.86
	401-423	17.44	16.40	17.64	16.47	12.02
	324-328	3.36	N/A	3.44	3.45	6.69
	424-428	2.47	N/A	3.26	3.20	7.34
$p = 1.5$	201-220	15.63	13.55	15.75	13.65	11.70
	301-323	13.15	13.15	13.16	13.15	14.12
	401-423	17.34	16.16	17.45	16.45	15.13
	324-328	3.23	N/A	3.35	3.34	9.73
	424-428	3.13	N/A	3.20	3.19	10.13
$p = 1.75$	201-220	15.51	13.70	15.87	13.77	14.13
	301-323	13.24	13.24	13.23	13.23	16.72
	401-423	17.34	16.23	17.46	16.42	17.58
	324-328	3.29	N/A	3.35	3.35	11.62
	424-428	3.09	N/A	3.20	3.20	12.31
Squared Euclidean	201-220	27.06	26.33	27.62	26.37	N/A
	301-323	26.25	26.24	26.45	26.31	N/A
	401-423	29.69	30.66	30.76	29.68	N/A
	324-328	10.10	N/A	10.48	10.50	N/A
	424-428	4.12	N/A	6.67	6.39	N/A

Table 8.14. Efficiency of the Lagrangean heuristics: Average CPU times (seconds) for all the problems with the Euclidean, squared Euclidean and l_p distances with $p =$ 1.25, 1.50, 1.75

Distance	Problems	CG(1)	BP(1)	CG(2)	BP(2)	RL_1/(R)DAH
Euclidean	201-220	1647.9	3233.3	370.2	606.8	1.4
	301-323	818.0	898.8	818.0	586.5	33.5
	401-423	734.5	6057.5	563.5	1398.3	16.5
	324-328	11286.4	N/A	11663.4	17639.4	304.5
	424-428	15136.5	N/A	12182.1	14614.8	221.9
	501-504	N/A	35892.3	N/A	N/A	4686.5
	601-604	N/A	55435.2	N/A	N/A	4229.7
$p = 1.25$	201-220	1543.3	3088.6	163.9	635.2	1.3
	301-323	204.8	340.4	91.8	102.9	30.7
	401-423	387.7	5854.9	208.6	1226.1	17.0
	324-328	10379.7	N/A	6706.0	8252.9	308.1
	424-428	11905.0	N/A	8844.1	15650.9	234.1
$p = 1.5$	201-220	1391.8	2949.8	347.6	561.8	1.4
	301-323	241.0	733.5	100.1	158.3	35.5
	401-423	436.9	5647.4	243.3	1173.5	16.9
	324-328	13826.2	N/A	7660.3	9463.8	303.1
	424-428	13096.3	N/A	8838.5	16309.7	223.7
$p = 1.75$	201-220	1169.6	2675.8	166.2	372.2	1.3
	301-323	294.7	731.4	93.1	95.0	29.2
	401-423	444.7	4944.7	230.1	1188.7	16.7
	324-328	15489.6	N/A	7655.7	9541.5	305.9
	424-428	13141.8	N/A	10513.7	13253.9	220.0
Squared Euclidean	201-220	1342.7	3470.1	586.1	1883.4	N/A
	301-323	1071.6	1625.4	237.3	269.8	N/A
	401-423	2922.3	8213.8	643.1	3073.9	N/A
	324-328	14799.6	N/A	8619.5	10754.8	N/A
	424-428	16665.9	N/A	15171.5	17681.3	N/A

9. CONCLUSIONS

In this thesis we have considered the capacitated multifacility Weber problem with the Euclidean, squared Euclidean and l_p distance for $1 \leq p < 2$. The capacitated multifacility Weber problem has nonconvex objective function and is very difficult to solve exactly. For this reason, instead of finding the exact solutions of this type of problems, we proposed some lower and upper bounding algorithms, which give tight bounds. First we propose four alternating location allocation heuristics and two discretization strategies which gives good upper bounds. Second we define four lower bounding algorithms, which are based on some special properties of the l_1 and l_∞ distances and their relations with general l_p distance. Third we formulate a Lagrangean relaxation of the capacitated multifacility Weber problem and solve the Lagrangean subproblem by using column generation and branch and price. In every iteration of these two algorithms several concave minimization subproblems are solved by outer approximation method.

For upper bounding algorithms, we compared two groups of heuristics. In the first group, we use region rejection type heuristics. The main idea of these heuristics are placing the facilities in the convex hull of the customers as balanced as possible initially and improving the objective by CALA heuristic. In the first two of these heuristics, RRH and DRRH, we first define a constant radius and consider not to locate a facility inside the radius of a previously fixed facility. The only difference between RRH and DRRH is that, in RRH a facility can be located anywhere in the convex hull of the customer locations whereas in DRRH a facility is placed on one of the customer locations. In RRH$'$ and DRRH$'$, previously defined two heuristics are improved by defining a dynamic radius concept and instead of defining a constant radius at the beginning of the iterations, each radius of the facility are assigned according to its capacity and the customers near to it. In the second group of heuristics, we investigate the DAH and its Lagrangean relaxation RDAH. In DAH, the customer locations are set as candidate facility locations and a DLAP is solved and this solution is improved by a CALA. In RDAH, instead of solving the problem exactly, a Lagrangean relaxation is considered, and a CALA is run in every step of the SO algorithm. Experimental results show that, DLAP and RDLAP outperforms the other four. In small instances with less than 50 customers and 10 facilities DAH performs good whereas in larger cases RDAH gives better results in limited time period with minimum CPU time compared to the other four.

As a lower bounding algorithm, we define four types of algorithms: L_1, L_∞ and their relaxations RL_1, RL_∞. Briefly, these are the same problems with different distance metrics, l_1 and l_∞ which are known as rectilinear and Tchebycheff distances. Since, the optimal facility locations of these problems are elements of a finite set of locations, these problems can be modeled as DLAP problems and can be solved optimally. In L_1 and L_∞ algorithms, problems are modeled and solved optimally whereas in RL_1 and RL_∞, a Lagrangean heuristic is devised which gives tight upper bounds in short CPU times. Experiments show that, RL_1 outperforms all other three with tighter lower bounds and shorter running times.

At the last part, we have first proposed a Lagrangean relaxation scheme for the CMFWP. After this relaxation, Lagrangean subproblem looks like an uncapacitated multifacility Weber problem. This Lagrangean subproblem is reformulated as a set partitioning model and solved using column generation and branch and price algorithms. Pricing subproblems are a d.c. programming problems, which are converted into concave minimization problems and solved by the outer approximation method. By using one of these two lower bounding algorithms with an appropriate upper bounding heuristic, a subgradient optimization scheme is generated and tight bounds are computed. The experimental results show that using UDAH or RUDAH algorithms as lower bounding algorithms inside the subgradient optimization scheme and validating the lower bound by solving a column generation or branch and price algorithm with the λ_i's that give the best lower bound in iterations returns the best solution. Finally we compared the best lower and upper bounds with the gaps found by the Lagrangean heuristic. It can be observed that, Lagrangean heuristic works better for instances with small facility to customer ratio. For all of the remaining instances, RL_1 and (R)DLAP outperforms the Lagrangean heuristics.

Although we have not been able to find an exact solution procedure yet, we proposed some methods that can be used in finding exact solution methods. The lower and upper bounding methods researched in the previous chapters can be used in a branch-and-bound scheme. In this type of algorithm, especially the Lagrangean heuristic can be very useful and by reformulating the variables the lower bounds of the nodes can be improved. As a final point it is possible to say that, tighter lower bounds can be obtained by combining other lower bounding methods with the Lagrangean heuristic.

REFERENCES

1. Sherali, H.D., and F.L. Nordai, 1988. "NP-hard, capacitated, balanced p-mediam problems on a chain graph with a continuoum of link demands", *Math Oper Res* 13, 32-49.

2. Krau, Stéphane, 1997. "Extensions du problème de Weber, Ph.D. Dissertation", *École Polytechnique de Montrèal, Montrèal, Canada* (in French).

3. Weiszfeld, E., 1937. "Sur le point lequel la somme des distances de n points donnès est minimum", *TShoku Mathematical Journal*, 43, 355-386.

4. Kuhn, H.W., 1973. "A note on Fermat's problem" *Mathematical Programming*, 4, 98-107.

5. Cooper, L., 1963. "Location-allocation problems", *Operations Research*, 11, 331-343.

6. Cooper, L., 1967. "Solutions of generalized locational equilibrium models", *Journal of Regional Science*, 7, 1-18.

7. Cooper, L., 1964. "Heuristic methods for location-allocation problems", *SIAM Review*, 6, 37-53.

8. Cooper, L., 1972. "The transportation-location problem", *Operations Research*, 20, 94-108.

9. Ostresh Jr., L.M., 1975. "An efficient algorithm for solving the two center location-allocation problem", *Journal of Regional Science*, 15, 209-216.

10. Rosing, K.E., 1992. "An optimal method for solving the (generalized) multi-Weber problem", *European Journal of Operational Research*, 58, 414-426.

11. Bongartz, I., P.H. Calamai, and A.R. Conn, 1994. "A projection method for l_p norm location-allocation problems", *Mathematical Programming*, 66, 283-312.

12. Bischoff, M., and K. Klamroth, 2007. "Two branch bound methods for a generalized class of location-allocation problems", *EURO Winter Institute on Location and Logistics.*

13. Chen, P.C., P. Hansen, B. Jaumard, and H. Tuy, 1998. "Solution of the multi-source Weber and conditional Weber problems by D-C programming", *Operations Research*, 46, 548-562.

14. Righini, G., and L. Zaniboni, 2007. "A branch-and-price algorithm for the multi-source Weber problem", *International Journal of Operational Research*, 2, 188-207.

15. du Merle, O., D. Villeneuve, J. Disrosiers, and P. Hansen, 1999. "Stabilized column generation", *Discrete Mathematics*, 194, 229-237.

16. Rousseau, L.M., M. Gendreau, and D. Feillet, 2007. "Interior point stabilization for column generation", *Operations Research Letters*, 35, 660-668.

17. Sherali, H.D., and C.H. Tunçbilek, 1992. "A squared Euclidean distance location-allocation problem", *Naval Research Logistics*, 39, 447-469.

18. Sherali, H.D., and W.P. Adams, 1999. "A reformulation-linearization technique for solving discrete and continuous nonconvex problems", Kluwer Academic Publishers, Dordrecht.

19. Sherali, H.D., I. Al-Loughani, and S. Subramanian, 2002. "Global optimization procedures for the capacitated Euclidean and l_p distance multifacility location-allocation problems", *Operations Research*, 50, 433-447.

20. Aras, N., M. Orbay, and İ.K. Altınel, 2008. "Efficient heuristics for the rectilinear distance capacitated multi-facility Weber problem", *Journal of the Operational Research Society*, 59, 64-79.

21. Aras, N., İ.K. Altınel, and M. Orbay, 2007. “New heuristic methods for the capacitated multi-facility Weber problem”, *Naval Research Logistics*, 54, 21-32.

22. Wendell, R.E., and A.P. Hurter, 1971. “Location theory, dominance and convexity”, *Oper Res* 21, 314-320.

23. Hansen, P., J. Perreur, and F. Thisse, 1980. “Location theory, dominance and convexity: some further results”, *Oper Res*, 28, 1241-1250.

24. Held, M., and R.M. Karp, 1971. “The traveling-salesman problem and minimum spanning trees: Part II.” *Math Programming*, 1, 6-25.

25. Beasley, J.E., 1993. “’Lagrangean Relaxation’, in Modern Heuristics Tecniques for Combinatorial Problems”, *C. Reeves ed., Wiley, NY.*

26. Brimberg, J., and R.F. Love, 1993. “Global convergence of a generalized iterative procedure for the minimum location problem with l_p distances”, *Operations Research* 41, 1153-1163.

27. Luis, M., S. Salhi, and G. Nagy, 2009. “Region-rejection based heuristics for the capacitated multi-source Weber problem”, *Computer & Operations Research*, 36, 2007-2017.

28. Thisse, J.F., J.E. Ward, and R.E. Wendell, 1984. “Some Properties of Location Problems with Block and Round Norms”, *Operations Research* 32, 1309-1327.

29. Dantzig, G.B., and P. Wolfe, 1960. “Decomposition principle for linear programs”, *Operations Research*, 8, 101-111.

30. Horst, R., P.M. Pardalos, and N.V. Thoai, 2000. *Introduction to global optimization 2nd edition*, Kluwer Academic Publishers, Dordrecht.

31. Horst, R., and H. Tuy, 1990. *Global optimization : deterministic approaches*, Springer-Verlag, Berlin.

32. Ryan, D.M., and B.A. Foster, 1981 "An integer programming approach to scheduling", *Computer Scheduling of Public Transport*, North-Holland, New York.

33. McMullen P., and G.C. Shephard, 1971. "Convex polytopes and the upper-bound conjecture", *London Math. Soc. Lecture Notes Series*, 3.

34. Chen, P.C., P. Hansen, and B. Jaumard, 1991. "On-Line and Off-Line vertex enumeration by Adjacency Lists", *Operations Research Letter*, 10, 403-409.