LAGRANGEAN HEURISTICS FOR THE CAPACITATED MULTI-FACILITY LOCATION ALLOCATION PROBLEM

by

Buket Avcı

B.S., Industrial Engineering, Boğaziçi University, 2005

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

Graduate Program in Industrial Engineering

Boğaziçi University

2007

LAGRANGEAN HEURISTICS FOR THE CAPACITATED MULTI-FACILITY LOCATION ALLOCATION PROBLEM

APPROVED BY:

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

Assist. Prof. Deniz Aksen .....

Assoc. Prof. Necati Aras .....

DATE OF APPROVAL: 12.06.2007

ACKNOWLEDGEMENTS

I would like to thank to my thesis supervisor Prof. İ. Kuban Altınel for his guidance throughout this research. He did not only serve as my supervisor but also challenged, encouraged and motivated me throughout my thesis period. I feel myself so lucky for having the opportunity to work with him.

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

I am also grateful to my mother for her continuous support and understanding throughout my hard times during the thesis period.

I am deeply grateful to all of my friends without whom I would not be able to finish this study. I would like to thank especially Özlem, Evrim, Hande, Güven and Osman.

Finally, I want to thank to Bulut for his never ending support at the hardest, but the best times of my life. Without him, it would be very hard for me to advance in this study.

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

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

ABSTRACT

LAGRANGEAN HEURISTICS FOR THE CAPACITATED MULTI-FACILITY LOCATION ALLOCATION PROBLEM

In this study, we consider the capacitated multi-facility location-allocation problem, also called multi-facility Weber problem. It is concerned with the determination of the location of m new facilities having known capacities, as well as the allocation of their supplies, to satisfy the demand of customers, such that the total transportation cost proportional to the distance between customers and facilities is minimized. The demand and location of each customer are given. This problem has a nonconvex objective function and is very difficult and sometimes even impossible to solve exactly. Therefore, using a discrete approximation becomes a promising strategy to obtain good approximate solutions. We first present a mixed integer linear programming approximation of the capacitated multi-facility Weber problem. We apply Lagrangean relaxation and subgradient optimization-based heuristics on this approximation and then propose new heuristics for the continuous version of the problem which also make use of these Lagrangean heuristics. Computational results on the test instances indicate that these heuristics are efficient and accurate.

We also study on exact solution procedures. We make use of the fact that the capacitated multi-facility Weber problem has a special transportation network structure and the optimal solution occurs at an extreme point of the feasible region. The procedures we work on range from branch-and-bound methods to affine scaling methods, to collapsing polytopes, and to send-and-split methods. Although we could not find an exact solution methodology by using these procedures, we were able to gain more insight about the problem and this can help us in the development of other heuristic methods in the future.

ÖZET

SIĞA KISITLI YER SEÇİMİ - TAŞIMA PROBLEMLERİNİN ÇÖZÜMÜ İÇİN LAGRANGE GEVŞETMESİ TABANLI SEZGİSEL YÖNTEMLER

Bu çalışmada, sığa kısıtlı çok tesisli yer seçimi-taşıma problemi üzerinde çalışılmıştır. Bu problem, sonlu sığaya sahip m adet tesisin, n adet müşterinin istemlerini karşılarken toplam taşıma giderlerini enküçükleyecek biçimde, düzlemde yerleştirilmesi ile ilgilenir. Bu bir dışbükey olmayan eniyileme problemidir ve eniyi çözümünü hesaplamak çok zor, bazı durumlarda ise olanaksızdır. Bu yüzden, bu çalışmada ilk olarak bu problemin tesis yerlerinin iki boyutlu Öklid düzlemi yerine, verilmiş sonlu boyutlu bir aday yer kümesinden seçildiği, kesikli uyarlaması göz önüne alınmakta ve karışık tam sayılı doğrusal bir programlama modeli geliştirilmektedir. Daha sonra bu model üzerine Lagrange gevşetmesi ve altgradyan algoritması temelli sezgisel yöntemler uygulanmakta ve problemin sürekli bütünleşik sürümü için bu sezgisellerden faydalanan yeni yöntemler önerilmektedir. Yapılan bilgisayısal deneyler bu sezgisel yöntemlerin etkili çalıştığını ve eniyi çözümlere çok yakın sonuçlar bulduğunu göstermektedir.

Bu çalışmada, sığa kısıtlı çok tesisli yer seçimi-taşıma problemi için kesin çözüm yöntemleri üzerinde de durulmuştur. Bu yöntemlerin geliştirilmesinde, problemin kısıtlarının özel bir iki parçalı ağ yapısına sahip olması ve eniyi çözümünün olurlu çözüm kümesinin bir köşe noktasında gerçeklenmesinden yararlanma düşünülmüştür. Denediğimiz yaklaşımlar dal sınır yöntemlerinden afin ölçekleme yöntemlerine, çöken çokyüzlüler yönteminden gönder ve böl yöntemine kadar uzanan geniş yelpazeyi oluşturmaktadır. Bu yöntemleri kullanarak, kesin çözüm veren bir yöntem bulamasak da, problem hakkında daha fazla öngörü kazanmamız, ileride daha etkili sezgisel yöntemlerin geliştirilmesinde yararlı olacaktır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$\mathbf{a}_j = (a_{j1}, a_{j2})^T$	Coordinates of customer j
c_{ij}	Unit shipment cost per unit distance between facility i and customer j
c_{kj}	Unit shipment cost per unit distance between point k and customer j
c_{ikj}	Unit shipment cost per unit distance between facility i located at point k and customer j
c'_{ik}	Unit shipment cost per unit distance from facility i located at point k
d_j	Demand of customer j
d_j^{rem}	Unsatisfied demand of customer j in LP-relaxation algorithm
$d(\mathbf{x}_i, \mathbf{a}_j)$	Distance between points x_i and a_j
i	Index of facilities
I	Subset of demand nodes in send-and-split method
j	Index of customers
$J1$	Set of customer locations within the candidate location set
$J2$	Set of customer locations which are not within the candidate location set
k	Index of candidate locations
K	Number of candidate locations
l_{ij}	Lower bound on the allocation variable w_{ij}
m	Number of facilities to be located
$\mathbf{m}_j$	Cluster centroids in the cellular heuristic
n	Number of customers
(n_x, n_y)	Grid resolutions for x and y axis in the cellular heuristic
P_{ik}	Maximum lower bound found when facility i is forced to be open at location k in Beasley's method
R	Radius where updates are performed within in the cellular heuristic

R_{max}	Maximum radius where updates are performed within in the cellular heuristic
R_{ik}	Maximum lower bound found when facility i is forced not to be open at location k in Beasley's method
S	Feasible region in collapsing polytopes method
s_i	Capacity of facility i
S_k	Containing polytope at iteration k in collapsing polytopes method
T	Number of two phase runs performed in the cellular heuristic
u_{ij}	Upper bound on the allocation variable w_{ij}
u_{kj}	Amount of goods to be shipped from point k to customer j
u_{ikj}	Amount of goods to be shipped from facility i located at point k to customer j
U_{ij}	Dual variables in dual formulation of the location-allocation problem
$\mathbf{x}_i = (x_{i1}, x_{i2})^T$	Coordinates of facility i
w_{ij}	Amount of goods to be shipped from facility i to customer j
z_{ik}	Binary variable which denote whether facility i is located at candidate point k
Z_{max}	Maximum lower bound found by Beasley's method
Z_{UB}	Upper bound to the objective function value
Z_{LR}	Lower bound to the objective function value found by Lagrangean relaxation
λ_j	Lagrange multiplier for customer j
δ	Reference value in knapsack method
δ_i	Dual variables in the dual formulation of linearly relaxed version of DCMLAP_2
π	Coefficient that scales step size in subgradient algorithm
π_j	Dual variables in the dual formulation of linearly relaxed version of DCMLAP_2
μ^t	Step size in subgradient algorithm
η	Learning rate in the cellular heuristic

ρ_k	Dual variables in the dual formulation of linearly relaxed version of $DCMLAP_2$
LAP	Location-allocation problem
CLAP	Continuous location-allocation problem
DLAP	Discrete location-allocation problem
CMLAP	Capacitated multi-facility location-allocation problem
DCMLAP	Discrete capacitated multi-facility location-allocation problem
$DCMLAP_2$	The version of DCMLAP which the unit shipment cost does not depend on the facility
CMFWP	Capacitated multi-facility Weber problem
RDLAP	Rectilinear discrete location-allocation problem
MILP	Mixed integer linear program
RDCMLAP	Relaxed version of the discrete capacitated multi-facility location-allocation problem
$RDCMLAP^i$	Subproblem associated with each facility i for RDCMLAP
ABLH	Adaptation of Beasley's Lagrangean heuristic
LRSO	Lagrangean relaxation and subgradient optimization
LRSOSP	Lagrangean relaxation and subgradient optimization with a single 2-phase method
LRSO2P	Lagrangean relaxation and subgradient optimization with 2-phase method
DACL	Discrete approximation with customer locations heuristic
LP	Linear programming
LR	Lagrangean Relaxation
MCNFP	Minimum concave cost network flow problems
RLT	Reformulation-Linearization Technique
SOM	Self-organizing map
TP	Transportation Polytope

1. INTRODUCTION

Deterministic location-allocation problems are concerned with the simultaneous determination of the best locations and capacity allocations of facilities in order to satisfy given demands of a set of customers with known locations such that the total transportation cost is minimized. There are a number of categorizations of these problems in the literature based on the characterization of the facilities. When the facility can be located anywhere in the Euclidean plane, we have a continuous location-allocation problem (CLAP). Otherwise, facility locations have to be selected from a given set of candidate locations, and the problem becomes a discrete location-allocation problem (DLAP). If there exist capacity constraints associated with the facilities, the resulting problem is capacitated. If the facilities are assumed to have infinite capacities, then we have the uncapacitated version. A final categorization is with respect to the number of facilities to be opened: The problem is a single-facility problem if the objective is to determine an optimal location for a single facility. It becomes the multi-facility problem when more than one facility have to be located optimally.

We study the capacitated multi-facility location-allocation problem (CMLAP) in this thesis. When there is just a single facility to be located, the problem is a pure location problem and there is no allocation decision to be made. However, if there is more than one facility, both the locations and capacity allocations of the facilities must be determined. Therefore, we have a more difficult problem than the single facility case. Each customer is served from the nearest facility for the uncapacitated problems, while this is not necessarily true for the capacitated multi-facility location-allocation problems. Moreover, the objective function of the problem is nondifferentiable at points where a facility location coincides with a customer location, and this makes a direct application of gradient-based algorithms impossible. As a result of these facts, the number of solution methods proposed in the literature for the capacitated multi-facility location-allocation problems is low as opposed to the uncapacitated and single facility versions.

In CMLAP, if the locations of facilities are known, the problem reduces to the ordinary transportation problem. On the other hand, if the allocations of facilities are given, the problem becomes a pure location problem. Although transportation and pure location problems are easy to solve, CMLAP belongs to a difficult class of problems because of the fact that locations and allocations should be determined simultaneously. It has been shown that the CMLAP with the Euclidean distance is NP-hard even if all the customers are located on a straight line (Sherali and Nordai, 1988).

We consider both the continuous and discrete versions of CMLAP in this study. It is very difficult and sometimes even impossible to solve the continuous version of CMLAP, also known as the capacitated multi-facility Weber problem (CMFWP), by using a commercial solver, since it has a nonconvex objective function. Therefore, using a discrete approximation for continuous problems becomes essential to find at least an approximate solution. What we mean by "discrete approximation" is that we solve the problem by using some candidate locations for facilities rather than allowing all the points in the plane to be a candidate for an optimal solution. This approximate solution gets better when the number of candidate facility locations increases, but then the problem size increases up to a point where it becomes impossible to solve the problem due to computational restrictions. In this case, using efficient heuristics for the discrete approximation of CMFWP becomes important to obtain favorable solutions. Therefore, our main objective in this study is to develop efficient and effective heuristics for the discrete version of CMLAP (DCMLAP) in order to solve both the discrete and continuous problems.

In Chapter 2, we give a mathematical programming formulation of CMFWP, which is followed in Chapter 3 by a literature review of the problem. Chapter 4 explains the relaxations that can be applied on a DLAP and Chapter 5 gives the description of proposed solution methods based on these relaxations. In Chapter 6, we explain exact solution procedures for CMFWP and illustrate the computational results of the proposed methods in Chapter 7. Finally, we conclude the thesis by making our comments and giving directions for the future research.

2. PROBLEM FORMULATION

In CMFWP, we are interested in finding the location of m facilities in the Euclidean plane in order to serve customers at n fixed points as well as the allocation of each customer to the facilities so that total transportation cost is minimized. There is also a capacity restriction for each facility. This location-allocation problem can be formulated as follows:

CMFWP:

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

s.t.

$$\sum_{i=1}^{m} w_{ij} = d_j \qquad j = 1, \ldots, n \tag{2.2}$$

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

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

Here n is the number of customers and m is the number of facilities to be located. d_j and $\mathbf{a}_j = (a_{j1}, a_{j2})^T$ represent the demand and coordinates of customer j, respectively. s_i is the capacity of facility i and $\mathbf{x}_i = (x_{i1}, x_{i2})^T$ is its unknown coordinates. w_{ij} represents the amount to be shipped from facility i to customer j with the unit shipment cost per unit distance being c_{ij}. $d(\mathbf{x}_i, \mathbf{a}_j)$ is the distance function between facility i and customer j. In the formulation above, Equation (2.2) assures that demand of each customer must be satisfied and Equation (2.3) guarantees that each facility cannot serve more than its capacity.

In the discrete version of the capacitated multi-facility location-allocation problem (DCMLAP), the facilities cannot be located anywhere in the plane, but only on the candidate locations. DCMLAP can be formulated as follows:

DCMLAP:

$$\min z = \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{j=1}^{n} c_{ikj}u_{ikj} \tag{2.5}$$

s.t.

$$\sum_{i=1}^{m}\sum_{k=1}^{K} u_{ikj} = d_j \qquad j = 1,\ldots,n \tag{2.6}$$

$$\sum_{k=1}^{K} z_{ik} = 1 \qquad i = 1,\ldots,m \tag{2.7}$$

$$\sum_{j=1}^{n} u_{ikj} = s_i z_{ik} \qquad i = 1,\ldots,m;\ k = 1,\ldots,K \tag{2.8}$$

$$u_{ikj} \geq 0,\ z_{ik} \in (0,1) \qquad i = 1,\ldots,m;\ k = 1,\ldots,K;\ j = 1,\ldots,n \tag{2.9}$$

Here K is the number of candidate locations. u_{ikj} represents the amount to be shipped from facility i located at point k to customer j with the unit shipment cost being c_{ikj}. c_{ikj} is obtained by the multiplication of the unit shipment cost per unit distance from facility i located at point k, c'_{ik}, by the distance between point k and customer j, namely $c_{ikj} = c'_{ik}d(\mathbf{x}_k, \mathbf{a}_j)$. Variables z_{ik} are binary variables which denote whether facility i is located at point k.

DCMLAP is a mixed-integer linear programming model. So there is no need for any transformation to linearize the model. It has three sets of constraints. Constraints (2.6) state that all customers' demand should be satisfied. Constraints (2.7) require that each facility can be located at only one point; but more than one facility may be located at a point. Constraints (2.8) ensure that total amount supplied from a facility cannot exceed its capacity. Both CMFWP and DCMLAP assume that the problem is balanced, i.e., total supply is equal to total demand. If total supply is less than total demand, there exists no feasible solution. If total supply is larger than total demand, the problem can be solved by adding a dummy customer.

In DCMLAP, when allocations u_{ikj}'s are given, then the problem reduces to a pure location problem which is separable into m single-facility location problems. The

solution can be found by inspection. On the other hand, when z_{ik}'s are given, the problem becomes an ordinary transportation problem which is quite easy to solve.

In DCMLAP, the unit shipment cost depends both on the location of the facility and the facility itself. This means that sending the same amount of goods from two different facilities located at the same point may have different costs. The assumption that the cost does not depend on the facility seems realistic, therefore we can have a more compact formulation with two indices as the following one:

DCMLAP$_2$:

$$\min\ z = \sum_{k=1}^{K}\sum_{j=1}^{n} c_{kj}u_{kj} \tag{2.10}$$

s.t.

$$\sum_{k=1}^{K} u_{kj} = d_j \qquad j = 1,\ldots,n \tag{2.11}$$

$$\sum_{k=1}^{K} z_{ik} = 1 \qquad i = 1,\ldots,m \tag{2.12}$$

$$\sum_{j=1}^{n} u_{kj} = \sum_{i=1}^{m} s_i\ z_{ik} \qquad k = 1,\ldots,K \tag{2.13}$$

$$u_{kj} \geq 0, z_{ik} \in (0,1) \qquad i = 1,\ldots,m;\ k = 1,\ldots,K;\ j = 1,\ldots,n \tag{2.14}$$

There are different distance functions proposed in the literature to model the distance mathematically. The ones which are frequently used are Euclidean, rectilinear, squared Euclidean and l_p distances. The mathematical formulas of these distance functions can be seen in Table 2.1.

It is known that the optimal locations of facilities lie in the convex hull of the set of customer locations for the continuous multi-facility location-allocation problem with rectilinear, Euclidean, squared Euclidean and l_p distance metrics (Hansen *et al.*, 1972). Moreover, Wendell and Hurter (1973) have shown that rectilinear discrete facility location-allocation problem (RDLAP) has always an optimal solution with facilities

Table 2.1. Most frequently used distance functions in location theory

Distance	Formula
Euclidean	$d(x_1, x_2) = (\|x_{11} - x_{21}\|^2 + \|x_{12} - x_{22}\|^2)^{1/2}$
Rectilinear	$d(x_1, x_2) = \|x_{11} - x_{21}\| + \|x_{12} - x_{22}\|$
Squared Euclidean	$d(x_1, x_2) = \|x_{11} - x_{21}\|^2 + \|x_{12} - x_{22}\|^2$
l_p distance	$d(x_1, x_2) = (\|x_{11} - x_{21}\|^p + \|x_{12} - x_{22}\|^p)^{1/p}$ $p \geq 1$

located at the intersection points of vertical and horizontal lines drawn through the customer locations. Sherali *et al.* (1994) used this property to formulate a mixed integer nonlinear programming model for RDLAP. Aras *et al.* (2006) also used this property to formulate a mixed integer, but linear programming model for RDLAP with the intersection points as the candidate locations. This property is not valid for the other distance metrics, but an approximate solution can still be found by solving an approximating MILP, which uses the points of an arbitrary grid lying in the convex hull of the customer locations. This MILP has exactly the same formulation with DCMLAP with the points on the grids as the candidate locations. This is proposed by Aras *et al.* (2007) in their work where they study different approaches for the determination of most accurate and efficient discrete approximations to CMFWP.

DCMLAP can be solved by using a commercial solver which can solve integer programming problems, but the required computation time may increase exponentially with the problem size. For example, in the case where candidate locations are chosen at the intersection points of customer locations, the maximum number of candidate locations is n^2. Then with a problem with n customers and m facilities, mn^2 binary variables z_{ik} and mn^3 continuous variables u_{ikj} are generated. As we stated earlier, the approximation gets better when the number of grid points, therefore the number of candidate locations, increases, which in effect increases the size of the problem and hence the required computation time.

It is clear that DCMLAP does not guarantee to find the optimal solution for CMFWP if the points on the grid does not include the optimal facility sites. In the limit, when the number of candidate points selected goes to infinity, we would obtain

an optimal solution to CMFWP. Therefore, there is a trade-off between the solution quality and the computation time.

3. LITERATURE SURVEY

Cooper was the first who addressed the capacitated versions of location-allocation problems. He proposed an exact total enumeration algorithm for the Euclidean distance location-allocation problem and provided two heuristic methods (Cooper, 1972). In the exact solution, enumeration of all basic feasible transportation flows is performed based on the fact that optimal solution lies at an extreme point of the feasible region. Since the number of extreme points can be very large, this method performs well for small sized problems. The first heuristic procedure, has the basic idea of alternatively solving the location and allocation problems. When started with an initial location of the facilities, the problem reduces to the ordinary transportation problem and optimal allocations are found by using the transportation algorithm. With these new allocations found, the problem becomes a pure location problem and the optimal location of facilities are determined. This two-phase process continues until no further improvement can be obtained. The second approach uses a heuristic method that ignores the capacity constraints and solves the uncapacitated problem. When the capacity constraints are violated, the sources having capacity which is not used are selected to serve the destinations that have uncovered demand. This allocation process is achieved in such a way that the resulting change in the objective function is relatively small. With this new set of allocations, the algorithm uses an alternating procedure.

Apart from Cooper's complete enumeration algorithm, there are two more exact methods to solve the CLAP to the best of our knowledge. The first one is a biconvex cutting plane procedure and can be found in Selim's unpublished dissertation (Selim, 1979). Although this procedure is more tractable than total enumeration, it was found to be effective only for very small instances. The second one is a global optimization procedure proposed by Sherali *et al.* (2002) for the capacitated Euclidean and l_p distance location-allocation problems. This is a branch-and-bound algorithm that implicitly enumerates the vertices of the feasible region of the transportation constraints to obtain a global minimum. The algorithm is based on a partitioning of the allocation space according to the fact that a variable is either zero or positive at an

optimal solution and finitely converges to a global minimum within a specified percentage tolerance. It incorporates some kind of specialized logical tests and cut-set based inequalities to exploit the special structures of the underlying transportation constraint set. Two different lower bounding schemes are used, one based on solving a projected location space subproblem, and the other based on using a Reformulation-Linearization Technique (RLT) to transform an equivalent representation of the original nonconvex problem into a higher dimensional linear programming relaxation. RLT is used in the solution of CLAP before by Sherali and Tuncbilek (1992) where they study the squared-Euclidean distance location-allocation problem. The main difference in these approaches stands from the fact that in the squared-Euclidean distance case, there exist a closed form expression for the allocation variables if the location of the facilities are known. As a result, this problem was shown to be transformable to a quadratic convex maximization problem and it becomes possible to use a linear programming representation to compute upper bounds (due to maximization) via a Lagrangean relaxation scheme since the problem can be represented in only allocation variables.

Beasley (1993a) presented a framework for developing Lagrangean heuristics (heuristics based upon Lagrangean relaxation and subgradient optimization) with respect to facility location problems. Lagrangean Relaxation (LR) is a technique used to find lower (upper) bounds for minimization (maximization) problems (Fisher, 1981). LR attaches Lagrange multipliers to some constraints in the problem and relax them into the objective function. Then, the resulting program is solved to optimality and this solution with given Lagrange multipliers is a lower bound on the optimal solution of the minimization problem. The next step is updating the Lagrange multipliers corresponding to this optimal solution. There are some basic approaches such as subgradient optimization and multiplier adjustment to calculate the values of the multipliers. Subgradient optimization is an iterative procedure that generates new multipliers from a given set of Lagrange multipliers in a systematic fashion. It tries to maximize the lower bound of the relaxed problem by suitable choice of multipliers. Beasley incorporates Lagrangean relaxation and subgradient optimization in the heuristics he presented. The basic idea behind a Lagrangean heuristic is that the information collected while calculating the Lagrangean lower bound (which may produce infeasible solutions for the

original problem due to relaxation) can be used in an attempt to construct a feasible solution to the original problem. In the Lagrangean heuristic, a sequence of Lagrange multipliers (defining lower bounds) and a sequence of feasible solutions (defining upper bounds) can be generated. At the end of the Lagrangean heuristic, the best feasible solution found is a heuristic solution to the original problem.

It is known that the optimal solution of CMFWP is attained at an extreme point of the feasible region (Cooper, 1972). This property is also shared by concave minimization (convex maximization) problems where a concave function is minimized over a constraint set of any form. This observation can help us to develop some methods by inspiring from the ones designed to solve concave minimization problems. The literature on the exact solution procedures for concave minimization problems is immense (Horst and Tuy, 1990). Therefore we briefly summarize benchmark works on concave cost network flow problems only.

An early branch-and-bound algorithm for the minimum concave cost network flow problems (MCNFP) is developed by Florian and Robillard (1971). This approach assumes that there are capacity restrictions and general concave costs on the arcs. The original network is transformed to an uncapacitated bipartite network by using the transformation of Wagner (1959). When this transformation is applied, some slack flows are created with zero cost. Branching corresponds to forcing flow on an arc in the original network or on a slack arc. The nodes of the branch-and-bound tree corresponds to trials with various arcs forced to having extremal flows. The structure of the extreme flows is exploited by constructing them in a way that prevents the formation of positive loops. Bounding is performed by using a linear underestimation function to generate a lower bound for this node of the tree. It is not necessary to explore all branches in the tree since upper and lower bounds are used to control enumeration.

A dynamic programming approach to MCNFP is the send-and-split method found by Erickson *et al.* (1987). The main work of the method relies on solving set-splitting and minimum-cost-chain problems repeatedly. This approach is implemented for uncapacitated networks, but capacitated arcs can be converted to an uncapacitated form

with Wagner's transformation. Since the optimal flow is extreme and the graph induced by this flow is a forest, the flow in an arc incident to any node i is the sum of the demands at nodes in some subset I of demand nodes. To determine how the flow should be sent from i to satisfy the demand at I, it is necessary to solve the subproblem $i \rightarrow I$ in which the demands at nodes not in I are first replaced by zero and then the demand at i is replaced by the total demand at I. The idea of the algorithm is to solve these subproblems inductively on the cardinality of I. Once subproblems are solved for all subsets of I, two decisions have to be made at each node i. The first one is to solve the set-splitting problem of finding a minimum-cost split of I into two nonempty subsets J and $I \backslash J$, and the cost of split is the sum of the minimum costs for the subproblems $i \rightarrow J$ and $i \rightarrow I \backslash J$. The second decision is to find a minimum cost chain along which to ship the demand at I from each node i to a node at which it is optimal to incur the cost of splitting I into subsets.

Falk and Hoffman (1986) propose a procedure to minimize globally a concave function over a bounded polytope. The procedure successively minimize the function over polytopes containing the feasible region and at the end, collapsing to the feasible region. Feasible region S is viewed as a face of a polytope C with dimension one greater than the dimension of S. At some iteration, a partial list of extreme points of C and edges emanating from these extreme points are found and those edges that intersect the hyperplane containing S define another set of vertices whose convex hull is a polytope S_k which S is a subset of. The procedure selects the most promising vertex of the current containing polytope S_k to refine the approximation. The method builds a tree whose terminal nodes coincide with the vertices of the feasible region and optimal solution of the original problem is found when the most promising vertex of S_k is also a vertex of the feasible region S.

The dual formulation of the facility location problem represents interesting properties. The dual formulation of the multi-facility location-allocation problem with l_p distance is given by Love (1974). The dual in this work is formulated by using quasi-linearization. Since the dual is valid only if any pair of facilities to be located or pair of facilities and customers are not located at the same location at optimum, a hyper-

bolic approximation is utilized which is uniformly convergent to the objective function of the primal problem. Love and Juel (1982) used the dual formulation proposed in (Love, 1974) to develop some solution methods for large scale uncapacitated location allocation problems. They show that the location-allocation problem can be stated as a concave minimization problem using the dual formulation. As stated previously, the optimal solution of a concave minimization problem must lie at an extreme point of the feasible region, and they use this property to formulate five new heuristics based on the new dual formulation. These heuristics differ from each other in the manner in which they perturb a given local optimal solution.

Several methods exist in the literature to calculate lower bounds on the objective function of facility location problems. The contribution of these papers is mostly relevant for iterative solution methods, such as Weiszfeld procedure (Weiszfeld, 1937), since it allows these solution methods to terminate when the objective function comes within a fraction of the optimal solution. The first attempt to develop lower bounds is due to Love and Yeong (1981). They discussed two methods to compute a lower bound on the objective function of the facilities location problem which is applicable to both single and multi facility cases. Later, Love and Dowling (1989) develop a bounding method for the multi-facility case with l_p distance which gives better bounds than these two methods. Drezner (1984) also proposes a lower bound for the single facility case with Euclidean distance which gives lower bounds at least as good as the methods proposed by Love and Yeong. Wendell and Peterson (1984) uses the dual formulation of the location problem in order to develop a lower bound.

4. RELAXATIONS FOR DISCRETE LOCATION-ALLOCATION PROBLEM

As stated previously, when the problem size is large, finding the optimal solution to DCMLAP is quite difficult due to high computational cost. Finding a good solution to such difficult problems deals with the consideration of two issues. These are calculation of upper and lower bounds which is as close as possible to optimal solution. Upper bounds are found by heuristic methods and lower bounds are found by solving a relaxed version of the original problem. Two of the most commonly used relaxation methods are Linear Programming relaxation and Lagrangean relaxation. In this chapter, we apply these two relaxation methods to DCMLAP in order to develop efficient and effective heuristics based on these relaxations.

4.1. Lagrangean Relaxation

Lagrangean Relaxation (LR) is a technique which is successfully used in integer and mixed integer linear programming problems with a minimization (maximization) objective function to find lower (upper) bounds (Fisher, 1981). There may exist some constraints which make a problem difficult to solve and the problem can be solved easily if these constraints are removed from the constraint set. Motivated by this fact, LR attaches Lagrange multipliers to those constraints which make the problem difficult and relax by adding their weighted sum to the objective function. Then, the resulting program is solved to optimality and this solution gives a lower bound on the optimal solution of the minimization problem. We apply Lagrangean Relaxation to find lower bounds for DCMLAP by relaxing the demand satisfaction constraints (2.6). This results in the following relaxed problem RDCMLAP.

RDCMLAP:

$$\min Z_{LR} = \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{j=1}^{n} c_{ikj}\ u_{ikj} + \sum_{j=1}^{n}\lambda_j(d_j - \sum_{i=1}^{m}\sum_{k=1}^{K} u_{ikj})$$

$$= \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{j=1}^{n}(c_{ikj} - \lambda_j)u_{ikj} + \sum_{j=1}^{n}\lambda_j\ d_j \tag{4.1}$$

s.t.

$(2.7) - (2.9)$

$$u_{ikj} \le d_j \qquad i = 1,\ldots,m;\ k = 1,\ldots,K;\ j = 1,\ldots,n \tag{4.2}$$

The λ_j's represent the Lagrange multipliers for each customer j. As there are n customers, there are n demand constraints which are relaxed. The constraints, $u_{ikj} \le d_j$, which are redundant in the original problem, should be added in order to improve the objective function of each subproblem, since it does not allow to send items from a facility to a customer more than the demand of this customer.

Lagrangean Relaxation may not seem to be an effective approach since the relaxed problem is also an integer or mixed-integer program as the original one. However, most of the integer or mixed-integer programs are complicated by some set of constraints and when we relax them, we come up with an easy problem to solve. Besides that, practical experience shows that LR gives good lower bounds with reasonable computational cost (Beasley, 1993b).

4.2. Linear Programming Relaxation

One well-known technique to calculate lower bounds for a minimization problem is Linear Programming (LP) relaxation. In LP relaxation, the integrality requirement on the variables is relaxed and the resulting linear program is solved to optimality by a standard method or solved heuristically. In our problem, $z_{ik} = 0$ or 1 can be replaced by the two continuous constraints $z_{ik} \ge 0$ and $z_{ik} \le 1$. For the version of DCMLAP which the unit shipment cost does not depend on the facility ($DCMLAP_2$), (i.e. $c_{ikj} = c_{kj},\ i = 1, ..., m$), we have found a very simple solution method which solves

this relaxed problem to optimality. For each customer j, we find the closest candidate location k and locate an arbitrary facility i to that location. If the demand of customer j is larger than the supply of facility i, we set $z_{ik} = 1$. Otherwise, we set $z_{ik} = d_j/s_i$. This is possible since we relax the requirement that $z_{ik} \in (0, 1)$. Then we update d_j with $d_j - s_i$ and continue this process (locating another facility i' to location k) until the demand of customer j is satisfied. Below, we present the method formally and prove that it finds the optimal solution for the relaxed problem.

Algorithm 1 LP-relaxation algorithm

---

1. Let the facilities be in the arbitrary order $i(1)$, $i(2)$,...,$i(m)$ and l=1.
2. For each customer j, do steps 3, 4 and 5.
3. Find the candidate location k with the minimum c_{kj} value.
4. Let $d_j^{rem} = d_j$ and $u_{kj} = 0$.
5. Repeat

   $u_{kj} \leftarrow u_{kj} + min(s_{i(l)}, d_j^{rem})$

   $z_{ik} = min(s_{i(l)}, d_j^{rem})/s_{i(l)}$

   If $s_{i(l)} > d_j^{rem}$

   $s_{i(l)} = s_{i(l)} - d_j^{rem}$ and $d_j^{rem} = 0$

   Else $d_j^{rem} \leftarrow d_j^{rem} - s_{i(l)}$ and $s_{i(l)} = 0$

   $l \leftarrow l + 1$

   until $d_j^{rem} = 0$.
6. $Z_{LP} = \sum_{k=1}^{K} \sum_{j=1}^{n} c_{kj}\ u_{kj}$.

Proposition: Algorithm 1 computes an optimal solution of the LP relaxation of DCMLAP for $c_{ikj} = c_{kj},\ i = 1, ..., m$.

Proof: The relaxed problem P for DCMLAP$_2$ is below.

$$\text{P: min } z = \sum_{k\in K}\sum_{j\in J1} c_{kj}u_{kj} + \sum_{k\in K}\sum_{j\in J2} c_{kj}u_{kj} \tag{4.3}$$

s.t.

$$\sum_{k\in K} u_{kj} = d_j \qquad j \in J1 \cup J2 \tag{4.4}$$

$$\sum_{k\in K} z_{ik} = 1 \qquad i \in I \tag{4.5}$$

$$\sum_{j\in J1} u_{kj} + \sum_{j\in J2} u_{kj} = \sum_{i\in I} s_i\ z_{ik} \qquad k \in K \tag{4.6}$$

$$u_{kj} \geq 0,\ 0 \leq z_{ik} \leq 1 \qquad i \in I,\ j \in J1 \cup J2,\ k \in K \tag{4.7}$$

where I is the set of facilities, K is the set of candidate locations, $J1$ is the set of customers within the candidate location set and $J2$ is the set of customers which are not in the candidate location set. In order to show that algorithm 1 works correctly, we write the dual of this LP and use the complementary slackness conditions. The dual of this LP is:

$$\text{D: max } z_D = \sum_{j\in J1} \pi_j d_j + \sum_{j\in J2} \pi_j d_j + \sum_{i\in I} \delta_i \tag{4.8}$$

s.t.

$$\pi_j + \rho_k \leq c_{kj} \qquad i \in I,\ j \in J1 \cup J2,\ k \in K \tag{4.9}$$

$$\delta_i - s_i\rho_k \leq 0 \qquad i \in I,\ k \in K \tag{4.10}$$

$$\pi_j,\ \delta_i,\ \rho_k \text{ unrestricted} \qquad i \in I,\ j \in J1 \cup J2,\ k \in K \tag{4.11}$$

For each customer j, let k^* be the candidate location with the minimum unit shipment cost, i.e., $c_{k^*j} = \min_{k\in K} c_{kj}$. Obviously $c_{k^*j} = 0$ for $j \in J1$ and $c_{k^*j} > 0$ for $j \in J2$. Let the facilities be in the arbitrary order $i(1), i(2), ..., i(m)$. Now determine the values of the decision variables as is done in algorithm 1. That is,

- $u_{k^*j} = d_j$
- $u_{kj} = 0 \qquad k \in K$ and $k \neq k^*$

- $z_{i(l')k^*} = min(s_{i(l')}, d_j^{rem})/s_{i(l')}$ where $d_j^{rem} = d_j - \sum_{l=1}^{l'} s_i(l)$ and $l' = 1, ..., L$ and L is the index of the facility which makes $d_j^{rem} = 0$
- $z_{i(l')k} = 0 \quad l' = L+1, ..., m, k = k^*$ and $l' = 1, ..., m,\ k \in K$

We can find a dual feasible solution corresponding to this primal feasible solution by using the complementary slackness conditions.

- $u_{k^*j} = d_j > 0 \ \Rightarrow \ \pi_j + \rho_{k^*} = c_{k^*j} \ \Rightarrow \ \pi_j = c_{k^*j}$ and $\rho_{k^*} = 0$
- $u_{kj} = 0 \ \Rightarrow \ \pi_j + \rho_k \leq c_{kj} \ \Rightarrow \ 0 \leq \pi_j \leq c_{kj}$ and $\rho_k = 0 \quad k \in K$ and $k \neq k^*$
- $z_{i(l')k^*} = min(s_{i(l')}, d_j^{rem})/s_{i(l')} \ \Rightarrow \ \delta_{i(l')} - s_{i(l')}\rho_{k^*} = 0 \ \Rightarrow \ \delta_{i(l')} = 0 \ l' = 1, ..., L$
- $z_{i(l')k} = 0 \ \Rightarrow \ \delta_{i(l')} - s_{i(l')}\rho_{k^*} \leq 0 \ \Rightarrow \ \delta_{i(l')} = 0 \ l' = L+1, ..., m, \ k = k^*$ and $l' = 1, ..., m, \ k \in K$

This solution is feasible as we set the values of the dual variables such that the constraints of the dual problem are satisfied. We can show this as follows:

- $\pi_j + \rho_{k^*} \leq c_{k^*j}$ is satisfied with equality for all $j \in (J1 \cup J2)$ since $\pi_j = c_{k^*j}$ and $\rho_{k^*} = 0$. However, $\pi_j + \rho_k \leq c_{kj}$ is satisfied with strict inequality for all $k \neq k^*$ and $j \in (J1 \cup J2)$ since ρ_k's corresponding to all $k \neq k^*$ are set to 0, $\pi_j = c_{k^*j}$ for all $j \in (J1 \cup J2)$ and $c_{k^*j} = min_{k \in K} c_{kj}$. This means that constraints (4.9) are satisfied.
- $\delta_i - s_i\rho_k \leq 0$ is satisfied with equality for all $i \in I$ and $k \in K$ since $\rho_k = 0$ for all $k \in K$ and $\delta_i = 0$ for all $i \in I$. This says that constraints (4.10) are also satisfied.
- Constraints (4.11) are trivially satisfied.

The last thing we have to do to complete the proof is to show that the objective function values of these primal and dual solutions are the same. The objective value of the primal problem is:

- $Z_P = \sum_{k \in K} \sum_{j \in J1 \cup J2} c_{kj}u_{kj} = \sum_{k \in K} \sum_{j \in J2} c_{kj}u_{kj} = \sum_{j \in J2} c_{k^*j}u_{k^*j} = \sum_{j \in J2} c_{k^*j}d_j$ since $c_{kj} = 0$ for $j \in J1$ and $k \neq k^*$

and the objective value of the dual problem is:

- $Z_D = \sum_{j \in J1} \pi_j d_j + \sum_{j \in J2} \pi_j d_j + \sum_{i \in I} \delta_i = \sum_{j \in J2} \pi_j d_j = \sum_{j \in J2} c_{k^* j} d_j$ since $\pi_j = c_{k^* j} = 0$ for $j \in J1$ and $\delta_i = 0$ for $i \in I$

Therefore, the objective values of the primal and dual solutions are the same. This completes the proof.

Q.E.D.

One interesting observation about the LP relaxation is the fact that the objective function value increases as the cardinality of the set $J1$, i.e., the set of customers within the candidate location set, decreases. We will illustrate this fact with an example. Figure 4.1 shows the change in the objective function value with respect to the cardinality of $J1$ where $|J| = 20$. When $|J1| = |J| = 20$, i.e., all customers are within the candidate location set, objective function value becomes zero. This fact is also intuitive, because the closest candidate location with minimum c_{kj} value for each customer becomes the customer itself with $c_{kj} = 0$. Then, when all customers whose locations are in the candidate location set, the objective function value becomes zero; and when the number of customers whose locations are in the candidate location set decreases, objective function value also increases since the minimum c_{kj} value is positive for each customer.

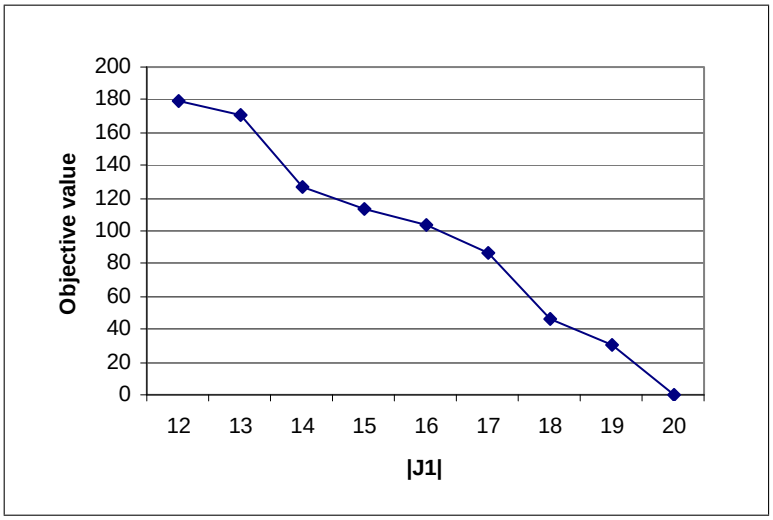

Figure 4.1. Change in the objective function value with respect to $|J1|$

The usefulness of any relaxation is determined by the closeness of its optimal value to the optimal value of the original problem. It is known that the optimal value provided by Lagrangean relaxation is at least as good as the optimal value provided by LP relaxation (Geoffrion, 1974). It is also known that they provide the same optimal value if and only if the polyhedron representing the feasible region of the problem where Lagrangean relaxation is applied has the integrality property, that is, the optimal value of the problem is not altered by dropping the integrality conditions on its variables. DCMLAP does not satisfy the integrality property due to constraints (2.8) as it is stated by Geoffrion (1974). Therefore, Lagrangean relaxation provides better bounds than LP relaxation for DCMLAP. This is the main reason for us to focus on Lagrangean relaxation to design accurate heuristics.

5. LAGRANGEAN HEURISTICS

In section 4.1, we applied Lagrangean relaxation for DCMLAP by relaxing the demand satisfaction constraint and presented the resulting relaxed problem RDCMLAP. When we analyze this problem, we see that it becomes separable over the facilities and the subproblem associated with each facility i (RDCMLAP^i) becomes quite easy to solve. By using this observation, we propose new heuristics based on Lagrangean relaxation in this section. We first apply Lagrangean Relaxation for all the methods we propose and then add new techniques to form new heuristics for making the gaps between lower and upper bounds as small as possible. In the first heuristic, the demand constraint is relaxed and a lower bound is attained by solving the resulting problem. Then this lower bound is maximized using the subgradient optimization. The second heuristic makes use of the same idea; the only difference is that starting with the best facility locations found by the first heuristic, a two phase heuristic is applied to find a better solution. In the third heuristic, two phase heuristic is applied to find a feasible solution at every iteration of the subgradient optimization, starting with the facility locations chosen in that iteration. The last heuristic adapts Beasley's Lagrangean heuristic for location problems. This heuristic tries to reduce the problem size and attain better lower bounds by forcing some facilities to be opened or closed at some locations.

5.1. Methods to Solve RDCMLAP^i Efficiently

When we apply Lagrangean relaxation on DCMLAP, the subproblem associated with each facility i is given as:

RDCMLAP^i :

$$\min Z_{LR^i} = \sum_{k=1}^{K}\sum_{j=1}^{n}(c_{ikj} - \lambda_j)u_{ikj} \tag{5.1}$$

s.t.

$$\sum_{k=1}^{K} z_{ik} = 1 \tag{5.2}$$

$$\sum_{j=1}^{n} u_{ikj} = s_i z_{ik} \qquad k = 1, \ldots, K \tag{5.3}$$

$$u_{ikj} \leq d_j \qquad k = 1, \ldots, K;\ j = 1, \ldots, n \tag{5.4}$$

$$u_{ikj} \geq 0,\ z_{ik} \in (0, 1) \qquad k = 1, \ldots, K;\ j = 1, \ldots, n \tag{5.5}$$

The best location for facility i is determined by placing the facility to each candidate location and finding the location with the minimum transportation cost $Z^*_{LR^i}$. Assume that the Lagrangean multipliers are fixed. The best solution for each location-facility combination is easily found by a greedy inspection procedure where we determine those customers that are supplied from facility i when located at candidate location k so that the total shipment cost $Z_{LR^{ik}} = \sum_{k=1}^{K}\sum_{j=1}^{n}(c_{ikj} - \lambda_j)u_{ikj}$ is minimized. We sort coefficients $(c_{ikj} - \lambda_j)$ in nondecreasing order and starting with the customer with the smallest coefficient $(c_{ikj} - \lambda_j)$, we allocate the quantities to the customers in this order without violating the constraints (5.3) until the supply of that facility is used completely. By this way, we can calculate a cost for each candidate location for each facility and the optimal location for this facility is the one with the minimum cost. The steps of the method used to solve the relaxed problem are formally given below.

There are other methods to solve the subproblems other than the greedy inspection procedure above. These methods differ in how they allocate quantities to customers. The method above sorts the customers with respect to their coefficients $(c_{ikj} - \lambda_j)$ in nondecreasing order and allocate the quantities to the customers in this order. The complexity of Algorithm 2 is dominated by the complexity of sorting done in step 3. For example, when "quicksort" is used, it has $O(n^2)$ worst case complexity

Algorithm 2 Solution of the RDCMLAP for fixed values of λ_j

1. For each facility i, do step 2 to compute $Z^*_{LR^i} = \min_k Z^*_{LR^{ik}}$
2. For each point k, do steps 3 through 5 to compute $Z^*_{LR^{ik}}$
3. Sort the customers in nondecreasing order with respect to $(c_{ikj} - \lambda_j)$. Let the customers in this order be given as $j(1), j(2), ..., j(n)$.
4. Let $s_i^{rem} = s_i$ and $l = 1$.
5. Repeat

   $u_{ikj} = min(s_i^{rem}, d_{j(l)})$

   $s_i^{rem} \leftarrow s_i^{rem} - d_{j(l)}$

   $l \leftarrow l + 1$

   until $s_i^{rem} = 0$
6. $Z^*_{LR^i} = \min_k Z^*_{LR^{ik}}$
7. Report $Z^*_{LR} = \sum_{i=1}^m Z^*_{LR^i} + \sum_{j=1}^n \lambda_j d_j$

since the complexity of quicksort is $O(n^2)$ in the worst case (Hoare, 1962). This value becomes $O(nlogn)$ for heapsort (Williams, 1964).

Observe that RDCMLAP^i is a continuous knapsack problem for $z_{ik} = 1$ and generic formulation of the continuous knapsack problem is:

$$\min(\mathbf{c}^T\mathbf{x} : \mathbf{a}^T\mathbf{x} \geq a_0, \mathbf{0} \leq \mathbf{x} \leq \mathbf{e})$$

where $\mathbf{c}$ and $\mathbf{a}$ are positive vectors, a_0 is a positive scalar and $\mathbf{e}$ is the unit vector. In our solution procedure for the subproblem RDCMLAP^i, we locate facility i at point k, therefore constraint (5.2) is satisfied for this facility-location combination. The constraint (5.3) for a fixed point k can be seen as the constraint $\mathbf{a}^T\mathbf{x} \geq a_0$ in the knapsack problem with inequality replaced by an equality. Constraints (5.4) and (5.5) together form the constraint $\mathbf{0} \leq \mathbf{x} \leq \mathbf{e}$ in the knapsack problem. $(c_{ikj} - \lambda_j)$, u_{ikj} and s_i can be seen as $\mathbf{c}, \mathbf{x}$ and a_0 in the knapsack problem. It is shown that the continuous knapsack problem can be solved in $O(n)$ time (Balas and Zemel, 1980). This solution procedure requires to find the median of n numbers and checks recursively if the knapsack is filled or not by comparing the objective coefficient of the elements

with the one in the median.

We apply the following solution procedure for each facility-location pair (i, k): Let $N_0,\ N_1,\ N_F$ denote the index set of allocation variables fixed at zero, fixed at positive, and free. All allocation variables are free and $\bar{S}_i = s_i$ initially. We find the median of $\bar{c}_{ikj} = c_{ikj} - \lambda_j$ values of the variables in N_F and partition N_F into 3 subsets with respect to the median. The first subset ($N^<$) includes the variables whose $\bar{c}_{ikj}$ values are less than the median while the second subset ($N^>$) includes the variables whose $\bar{c}_{ikj}$ values are larger than the median. And the third one ($N^=$) includes the variables whose $\bar{c}_{ikj}$ values are equal to the median. Then we find the total demand of the variables in $N^<$ (denoted by S_1) and the total demand of the variables in $N^< \cup N^=$ (denoted by S_2). If $\bar{S}_i$ is between S_1 and S_2, we stop since we could be able to find those u_{ikj}'s with the smallest $\bar{c}_{ikj}$ to use $\bar{S}_i$ completely. We set $u_{ikj} = d_j$ in $N^< \cup N_0$, $u_{ikj} = 0$ in $N^> \cup N_1$, $u_{ikj} = d_j$ in $N^=$ until $\bar{S}_i$ is used up (any, possibly including one with $u_{ikj} < d_j$). If $\bar{S}_i$ is smaller than S_1 (i.e., the median is too small), we add the variables in $N^>$ and $N^=$ to N_1 and replace N_F with $N^<$. If $\bar{S}_i$ is greater than S_2 (i.e., the median is too large), we add the variables in $N^<$ and $N^=$ to N_0 and replace N_F with $N^>$. Then we update $\bar{S}_i$ by $\bar{S}_i - S_2$ and apply the same procedures above on the variables in N_F. The steps of the method are formally given in Algorithm 3.

The complexity of this procedure is dominated by median computations. Therefore, as a third approach, we decided to select the reference value δ randomly as an attempt to reduce the computation time for the subproblems on the average. These approaches can perform better than the worst case behavior on the average. Motivated by this fact, we performed some experiments on some test instances to determine which method solves the subproblems with minimum computational effort. Table 5.1 shows the results of this experiment. Table 5.1 includes the number of facilities and customers for each instance and the CPU times in seconds, first one being the CPU times of greedy inspection procedure with quicksort, the second one being the CPU times of knapsack method which uses median as the reference value δ and the third one being the CPU times of the knapsack method which selects reference value randomly.

Algorithm 3 Solution of the RDCMLAP as a Continuous Knapsack Problem

1. For each facility i, do step 2 to compute $Z^*_{LR^i} = \min_k Z^*_{LR^{ik}}$
2. For each point k, do steps 3 through 5 to compute $Z^*_{LR^{ik}}$
3. Initialize set $N_0 = \emptyset, N_1 = \emptyset, N_F = \{1, 2, ..., n\}$ and $\bar{S}_i = s_i$.
4. Determine the median δ of the values $(c_{ikj} - \lambda_j)_{j \in N_F}$, partition N_F into $N^> = \{j \in N_F | (c_{ikj} - \lambda_j) > \delta\}$
   $N^< = \{j \in N_F | (c_{ikj} - \lambda_j) < \delta\}$
   $N^= = \{j \in N_F | (c_{ikj} - \lambda_j) = \delta\}$
   and calculate $S_1(\delta) = \sum_{j \in N^<} d_j$
   $S_2(\delta) = S_1(\delta) + \sum_{j \in N^=} d_j$
5. If $S_1(\delta) < \bar{S}_i \leq S_2(\delta)$, stop: $\delta^* = \delta$. An optimal solution to subproblem is obtained by setting $u_{ikj} = d_j,\ j \in N_0 \cup N^<;\ u_{ikj} = 0,\ j \in N_1 \cup N^>$ and then setting the variables $u_{ikj} = d_j,\ j \in N^=$ until $\bar{S}_i$ is used up (any, possibly including one with $u_{ikj} < d_j$). If $S_1(\delta) \geq \bar{S}_i$, set $N_1 \leftarrow N_1 \cup N^> \cup N^=$ and $N_F \leftarrow N^<$. If $S_2(\delta) \leq \bar{S}_i$, set $N_0 \leftarrow N_0 \cup N^< \cup N^=$, $N_F \leftarrow N^>$ and $\bar{S}_i \leftarrow \bar{S}_i - S_2(\delta)$. Then go to step 4.

Table 5.1. CPU times for different subproblem solution methods

Instance	m	n	Greedy procedure with quicksort	Knapsack method with median selection	Knapsack method with random selection
101	5	20	2.35	7.38	5.65
102	8	20	4.00	10.67	6.66
103	10	25	9.52	22.91	15.50
104	5	30	11.17	19.25	12.25
105	6	25	18.39	35.34	24.55
106	7	30	13.76	19.28	12.02
107	6	20	10.40	23.06	16.58
108	8	15	3.67	10.69	7.22
109	6	30	15.30	27.92	18.09
110	8	25	10.98	26.95	18.02
121	10	50	348.62	586.16	317.89
122	5	50	287.59	337.64	217.34
123	5	75	1318.22	836.25	557.08
124	10	75	1377.72	1708.06	984.72

Continued on Next Page...

Instance	m	n	Greedy procedure with quicksort	Knapsack method with median selection	Knapsack method with random selection
125	10	100	4917.69	4093.42	2339.47
126	20	75	2056.80	4497.44	2501.53
Average CPU time			650.39	766.40	440.91

The worst CPU times are reported for the knapsack method which uses median as the reference value. The greedy inspection procedure gives better results on the average than this one although its complexity is higher than the complexity of knapsack method. The best CPU times are observed for knapsack method which selects δ randomly. Therefore, we decided to use it in our heuristics for solving RDCMLAP^i.

5.2. Lagrangean Heuristics for Discrete Location-Allocation Problem

5.2.1. Lagrangean Relaxation and Subgradient Optimization (LRSO)

Z^*_{LR} provides a lower bound on the optimal solution of RDCMLAP for Lagrange multipliers λ_j. To find the best lower bound, Lagrange multipliers that yield the highest objective function value are to be found. We consider the Lagrangean dual: $\max_\lambda Z^*_{LR}(\lambda)$ and apply subgradient optimization algorithm for this purpose. The subgradient algorithm is formally described below.

Algorithm 4 Subgradient Algorithm

1. Set $\lambda_j^{(0)} = 0,\ t = 0.$
2. Repeat steps 3 and 4 for a predetermined number of iterations.
3. With the current values of λ_j, solve relaxed problems RDCMLAP and obtain $Z^*_{LR} = \sum_{i=1}^{m} Z^*_{LR^i} + \sum_{j=1}^{n} \lambda_j^{(t)} d_j$.
4. Update the multipliers λ_j by setting $\lambda_j^{(t+1)} = \lambda_j^{(t)} + \mu^{(t)}(d_j - \sum_{i=1}^{m}\sum_{k=1}^{K} u_{ikj})$ where the step size, $\mu^{(t)}$ is determined by:

$$\mu^{(t)} = \frac{\pi(Z_{UB} - Z^*_{LR})}{\sum_{j=1}^{n}(d_j - \sum_{i=1}^{m}\sum_{k=1}^{K} u_{ikj})^2}$$

Here, Z_{UB} represents the upper bound to the original problem, which can be the objective value of any feasible solution, Z^*_{LR} represents lower bound to the original problem, found by solving the Lagrangean subproblem and π represents an arbitrary coefficient that scales the step size. The step size $\mu^{(t)}$ depends on the gap between Z_{UB} and Z^*_{LR} and the parameter π defined by the user with $\sum_{j=1}^{n}(d_j - \sum_{i=1}^{m}\sum_{k=1}^{K} u_{ikj})^2$ being the scaling factor.

After each iteration, a solution is found which is feasible with respect to the locations but infeasible with respect to the allocations since it could be shipped to a customer more or less than its demand. Therefore, a transportation problem is solved to find a feasible solution corresponding to these infeasible solutions by using the locations found. A solution method which incorporates the transportation simplex algorithm can solve the transportation problem, however it is known that transportation problem is a special case of the minimum cost flow problem and this network problem has quite efficient solution methods due to its network structure. Therefore, an algorithm which can solve minimum cost flow problem efficiently can also solve the transportation problem efficiently. Therefore, we used an open source minimum cost flow algorithm to solve the transportation problem (Pisa, 2006) and get a feasible solution at the end of the iterations of the subgradient algorithm.

5.2.2. Lagrangean Relaxation and Subgradient Optimization with a Single 2-Phase Method(LRSOSP)

Our second heuristic is an extended version of LRSO with a two phase method. In this method, starting with the best facility locations found by LRSO, the two phase method which is described in Algorithm 5 is applied to find a better solution. It is expected that the solution found by LRSO can be improved by using this method without sacrificing a significant amount of computation time. However, we do not expect an improvement in the lower bounds. The two phase heuristic for DCMLAP is given below.

Algorithm 5 Two Phase Heuristic

1. Locate the facilities at arbitrary selected candidate points $x_i = (x_{i1}, x_{i2})^T$, $i = 1, ..., m$
2. Calculate the distances between the customers and facilities according to the distance metric used.
3. Solve the transportation problem to determine allocations u_{ikj}.
4. Using allocations u_{ikj}, solve m 1-median problems.
5. Repeat steps (2), (3) and (4) until either the facility locations $x_i = (x_{i1}, x_{i2})^T$ or the allocations u_{ikj} remain unchanged.

In step 4 of Algorithm 5, we search for the best location within the candidate location set for each facility i. We ignore the index k in u_{ikj} and assume that u_{ikj} represents the quantity allocated from facility i to customer j. Then we place each facility i to each location k and find the location with the minimum transportation cost (with given u_{ikj}'s).

5.2.3. Lagrangean Relaxation and Subgradient Optimization with 2-Phase Method(LRSO2P)

In this method, two phase heuristic is applied to find a feasible solution at every iteration of the Subgradient Optimization, starting with the facility locations chosen in that iteration. Each iteration of LRSO gives a solution in terms of the facility locations. Then, we apply the two phase method to obtain a good feasible solution at each iteration. At the end, the solution with the minimum cost is selected as the best solution.

This heuristic is not used to improve the lower bound. Our aim is to obtain a better upper bound, i.e., a better objective function value. There is obviously a trade off between resource usage and the quality of the solution obtained. We expect better upper bounds but also more resource usage as we increase the number of operations per iteration.

5.2.4. Adaptation of Beasley's Lagrangean Heuristic (ABLH)

This heuristic is an adaptation of the procedure proposed in Beasley (1993a). Beasley presented a framework in this paper for developing Lagrangean heuristics for facility location problems. The main idea behind this framework is to reduce the problem size by eliminating some candidate locations and by forcing some facilities to be opened at some candidate locations after comparing the objective function values corresponding to these candidate locations. We adapt Beasley's method in this heuristic by making some changes in DCMLAP.

As stated before, RDCMLAP becomes separable over facilities and locations, and the subproblem (SP) associated with each facility i and location k can be given as:

$$\text{SP: min } Z_{ik}(\lambda) = \sum_{j=1}^{n} \bar{c}_{ikj} u_{ikj} \tag{5.6}$$

s.t.

$$\sum_{j=1}^{n} u_{ikj} = s_i \tag{5.7}$$

$$u_{ikj} \leq d_j \qquad j = 1, \ldots, n \tag{5.8}$$

$$u_{ikj} \geq 0 \qquad j = 1, \ldots, n \tag{5.9}$$

where $\bar{c}_{ikj} = c_{ikj} - \lambda_j$.

There are some definitions to be used in the description of the heuristic below.

- Z_{max}: the maximum lower bound
- Z_{UB}: the best feasible solution
- $z(\lambda)$: the optimal objective function value of RDCMLAP
- N: the number of subgradient iterations
- P_{ik}: the maximum lower bound found when facility i is forced to be open at location k
- R_{ik}: the maximum lower bound found when facility i is forced not to be open at

location k

- Q: set of facility-location pairs (i,k) where z_{ik}'s are fixed to 1 (i.e., we decided to open facility i to location k in the optimal solution)
- π: step size parameter

The formal description of this heuristic is presented in Algorithm 6.

In step 1 of Algorithm 6, we initialize both P_{ik} and R_{ik} to zero for $i = 1, ..., m;\ k = 1, ..., K$ since we do not have any information about the lower bounds to be found when a facility is forced to be open or close. After necessary initializations are performed, we have to find a starting feasible solution to the relaxed problem RDCMLAP. In step 2, we solve RDCMLAP with the current set of Lagrange multipliers and update the maximum lower bound found (Z_{max}) if the optimal objective value of RDCMLAP is greater than Z_{max}. Otherwise, we increase the value of N by 1 since we performed an iteration without an increase in the value of Z_{max}. In step 3, we try to obtain a feasible solution to DCMLAP by solving a transportation problem with optimal locations z_{ik}'s found in step 2. If the optimal objective value of this transportation problem is smaller than Z_{UB}, we update Z_{UB} with this value. Step 4 checks whether the maximum lower bound (Z_{max}) is updated or not in the last 30 iterations. If Z_{max} is not updated in the last 30 iterations, we halve the step length parameter π to make a finer search possible and then return to the the solution of RDCMLAP where Z_{max} is last updated and solve a transportation problem with z_{ik}'s of this solution. If necessary, we update Z_{UB} accordingly as in step 3.

Step 5 checks if the maximum lower bound and the best feasible solution found are equal to each other. If they are equal, the feasible solution giving Z_{UB} is optimal and the algorithm stops. If they are not equal, we continue to step 6 to update the maximum lower bounds found (P_{ik}) when facility i is forced to be open at location k. It is clear that imposing the additional constraint that a particular facility i must be opened at location k in the optimal solution ($z_{ik} = 1$) results in a corresponding lower bound of $z(\lambda) + Z'_{ik}(\lambda) - max(Z'_{pq}(\lambda) :\ z_{pq} = 1, (p,q) \notin Q\ p = 1, ..., m;\ q = 1, ..., K)$ if $z_{ik} = 0$ ($z(\lambda)$ otherwise($z_{ik} = 1$)). Here $Z'_{pq}(\lambda)$ must be computed by solving problem

Algorithm 6 Adaptation of Beasley's Lagrangean Heuristic (ABLH)

1. Set $Z_{max} = -\infty,\ Z_{UB} = \infty,\ N = 0,\ P_{ik} = 0$ and $R_{ik} = 0$ for $i = 1, ..., m;\ k = 1, ..., K,\ \lambda_j = min\{c_{ikj} : c_{ikj} > 0 \text{ for } i = 1, ..., m;\ k = 1, ..., K\},\ Q = \emptyset$.
2. Solve RDCMLAP with the current set of multipliers to obtain $z(\lambda)$, z_{ik} and u_{ikj}. If $z(\lambda) > Z_{max}$, set $N = 0$ and $Z_{max} = z(\lambda)$. Otherwise set $N = N + 1$.
3. Solve the transportation problem with z_{ik}'s (optimal locations found by solving RDCMLAP) to obtain a feasible solution to DCMLAP. If the cost associated with this solution is lower than the current value of Z_{UB}, update Z_{UB} with this value.
4. If $N = 30$, then 30 iterations of the subgradient procedure have been performed without an increase in the maximum lower bound Z_{max} found so far, halve the step length parameter $\pi = (\pi/2)$, set $N = 0$ and solve a transportation problem with the current z_{ik}'s (obtained by solving current RDCMLAP) to obtain a feasible solution to DCMLAP. Update Z_{UB} with the current cost if it is lower.
5. If $Z_{max} = Z_{UB}$, then $STOP$, the feasible solution giving Z_{UB} is optimal.
6. For any facility i and location k, we can update P_{ik} using $P_{ik} = max\{P_{ik},\ z(\lambda)\}$ if $z_{ik} = 1$ and $P_{ik} = max\{P_{ik}, z(\lambda) + Z'_{ik}(\lambda) - max(Z'_{pq}(\lambda) :\ z_{pq} = 1,\ (p, q) \notin Q,\ p = 1, ..., m;\ q = 1, ..., K)\}$ if $z_{ik} = 0$ where $Z'_{pq}(\lambda)$ must be computed by solving problem SP. We can therefore remove facility-location pair (i, k) from the problem if $P_{ik} > Z_{UB}$ and $(i, k) \notin Q$, since any facility-location pair (i, k) with P_{ik} greater than Z_{UB} cannot be in an improved feasible solution.
7. For any facility i and location k, we can update R_{ik} using $R_{ik} = max\{R_{ik}, z(\lambda)\}$ if $z_{ik} = 0$ and $R_{ik} = max\{R_{ik}, z(\lambda) - Z'_{ik}(\lambda) + min(Z'_{pq}(\lambda) :\ z_{pq} = 0,\ p = 1, ..., m;\ q = 1, ..., K)\}$ if $z_{ik} = 1$ for$(i, k) \notin Q$ where $Z'_{pq}(\lambda)$ must be computed by solving SP. We can therefore open facility i at location k by setting $Q = Q \cup (i, k)$ if $R_{ik} > Z_{UB}$ and $(i, k) \notin Q$, since any facility i and location k with R_{ik} greater than Z_{UB} must be open in any improved feasible solution.
8. Calculate the subgradient $G_j = d_j - \sum_{i=1}^{m} \sum_{k=1}^{K} u_{ikj}$ for $j = 1, ..., n$
9. If $\sum_{j=1}^{n} G_j^2 = 0$ or $\pi < 0.0005$, go to step 11.
10. Define step size $\mu^{(t)}$ by $\mu^{(t)} = \frac{\pi(1.05Z_{UB} - Z_{max})}{\sum_{j=1}^{n} G_j^2}$ and update Lagrange multipliers $\lambda_j = \lambda_j + \mu^{(t)} G_j\ j = 1, ..., n$ and go to step 2.
11. Report the best feasible solution which gives Z_{UB} calculated so far.

SP for the facility-location pairs which are not in Q and which are open in the solution giving current $z(\lambda)$. Hence, we can update P_{ik} by comparing the value of the previous P_{ik} with the final lower bound found and choosing the maximum one. One interesting feature of the algorithm is that it is possible to remove some facility-location pairs from the problem if they are proved not to be in the optimal solution. If $P_{ik} > Z_{UB}$, we remove pair (i, k) from the problem since P_{ik} is a lower bound on the optimal solution of DCMLAP when facility i is opened at location k, and it cannot be greater than the best feasible solution found.

In step 7, we update the maximum lower bounds found (R_{ik}) when facility i is forced not to be open at location k. Similarly, it is clear that imposing the additional constraint that a particular facility i cannot be opened at location k in the optimal solution ($z_{ik} = 0$) results in a corresponding lower bound of $z(\lambda) - Z'_{ik}(\lambda) + min(Z'_{pq}(\lambda) : z_{pq} = 0,\ p = 1, ..., m;\ q = 1, ..., K)$ if $z_{ik} = 1$ and $(i, k) \notin Q$ ($z(\lambda)$ otherwise($z_{ik} = 0$)). Here $Z'_{pq}(\lambda)$ must be computed by solving problem SP for the facility-location pairs which are not open in the solution giving current $z(\lambda)$. Hence, we can update R_{ik} by comparing the value of the previous R_{ik} with the final lower bound found and choosing the maximum one. In this case, it is possible to decide to open facility i at location k in the optimal solution if $R_{ik} > Z_{UB}$. If R_{ik} (a lower bound on the optimal solution of DCMLAP when facility i is not opened at location k) is greater than than the best feasible solution found, facility-location pair (i, k) must be open in the optimal solution and we add this pair to set Q.

Remaining steps of the algorithm (steps 8-11) are very similar to the subgradient algorithm (Algorithm 4) in section 5.2.1.

We made some modifications for step 3 and 4 on Beasley's heuristic. Beasley proposed to use an interchange heuristic for that step, but we applied the transportation algorithm instead.

In contrast to the LRSOSP and LRSO2P, the aim of this heuristic is to obtain better lower bounds. Since there are some location fixing and elimination operations,

ABLH implementation is expected to increase the resource usage with respect to LRSO.

5.3. Lagrangean Heuristics for Continuous Location-Allocation Problem

All the methods proposed above can be used to solve DCMLAP. In this section, we develop some heuristics for CMFWP by using DCMLAP as an approximation. These heurictics are described in detail in the following sections.

5.3.1. Discrete Approximation with Customer Locations Heuristic

As stated before, we can use DCMLAP as an approximating problem to solve CMFWP and approximation becomes more accurate when the number of candidate location increases. The required computation time also increases with increasing number of candidate locations. However, if we use some "good" candidate locations instead of giving all points in the grid the same chance of being a candidate location, we can obtain a high quality solution with less number of candidate locations. Therefore, we can prevent the required computation time to increase without sacrificing from solution quality. It is observed that the facility locations are usually close to the customer locations in an optimal solution to Euclidean distance uncapacitated multi-facility location problem (Hansen *et al.*, 1998). We decided to perform some experiments to check if this observation is also valid for the capacitated problem. The results of these experiments can be seen in Table 5.2. The columns of Table 5.2 show the number of facilities and customers for each test instance and the rate obtained by dividing the number of facilities located on the customer locations to the total number of facilities. First eight instances on Table 5.2 are rectilinear distance problems and taken from literature (Sherali *et al.*, 2002; Sherali *et al.*, 1994). The remaining ones are Euclidean distance problems and created randomly.

The first eight instances in Table 5.2 are continuous problems while the remaining ones are discrete problems. As stated before, rectilinear distance facility location problem has always an optimal solution with facilities located at the intersection points of vertical and horizontal lines drawn through the customer locations Wendell and Hurter

Table 5.2. Overlap rate of the facility locations with customer locations

Instance	m	n	Overlap rate (%)
8	4	8	100
9	5	15	80
15	5	10	100
16	4	10	75
23	5	8	100
26	5	12	80
29	5	15	60
30	5	20	40
101	5	20	100
102	8	20	100
103	10	25	100
104	5	30	80
105	6	25	100
106	7	30	100
107	6	20	100
108	8	15	100
109	6	30	100
110	8	25	100
Average			89.72

(1973). As a result, we can solve the continuous version of this problem optimally by using these intersection points as the candidate locations in a discrete approximation heuristic. However, this property is not shared by the other distance metrics and there are only a small number of instances whose optimal solutions are known in the literature. Therefore, we treat the remaining ten problems as discrete problems and solve them by using a discrete approximation. The solutions found are not the optimal solutions of the continuous version of these instances. This difference shows itself in the overlap rate, since the overlap rate is higher for the discrete problems than the continuous ones. However, this may also be due to the distance metric, i.e., Euclidean distance problems may have a higher overlap rate than rectilinear distance problems since the intersections points of vertical and horizontal lines drawn through the customer locations may be sometimes more favorable than customer locations for rectilinear distance problems. Nevertheless, the results on Table 5.2 say that it is highly probable for the optimal facility locations to coincide with the customer locations. Therefore, we are convinced by this observation to use only customer locations as candidate sites for facilities.

In the first step of the discrete approximation with customer locations heuristic (DACL), customer locations are used as candidate locations for facilities and DCMLAP is solved to find the best locations. Then a single facility Weber problem is solved by using Weiszfeld's algorithm for each facility using the set of customers it serves. There are two approaches to solve the first step of this heuristic. We can either solve it to optimality by using a commercial solver or we can use our Lagrangean heuristics explained in Section 5.2 which may not solve DCMLAP to optimality. We have done experiments for both approaches to assess the performance of our heuristics. The results of these experiments can be found in Chapter 7.

5.3.2. Cellular Heuristic

Cellular heuristic also tries to obtain a high quality discrete approximation for CMFWP like DACL heuristic. The difference is that this heuristic generates promising candidate locations by means of the two-phase heuristic rather than using just customer

locations as candidate sites for facilities.

Gamal and Salhi (2003) propose a cellular heuristic to solve Euclidean distance uncapacitated multi-facility CLAP. In this heuristic, a rectangle is drawn to cover all the locally optimal facility locations and the rectangle is divided into cells according to a predefined resolution. Then, facility locations reside within a cell are used to compute a representative point for that cell. This is also used by Aras *et al.* (2007) to determine candidate locations for the solution of the CMFWP using a discrete approximation. We adopt this idea in our cellular heuristic, but other than dividing the rectangle into cells according to a predefined solution, we let a clustering algorithm to cluster facility locations into different classes. In that sense, our method make use of the information given in terms of facility locations and put clusters to locations where facilities are grouped with a high density.

First, we run two phase heuristic for a large number of times starting with randomly chosen initial facility locations within the convex hull of customer locations. Then, we cluster these locally optimal facility locations into classes by using self-organizing maps (SOM) clustering algorithm. SOM is proposed by Kohonen as an unsupervised neural network algorithm (Kohonen, 1990). SOM organizes items into clusters that are situated in some topology that is usually chosen as a rectangular topology. SOM generate these clusters such that neighboring clusters in the topology are more similar to each other than clusters far from each other in the topology. We use the SOM algorithm used to cluster genes at the Human Genome Center, University of Tokyo (2006). We now describe the steps of our cellular heuristic more formally.

Let $\mathbf{x}_i^{(t)} = (x_{i1}^{(t)}, x_{i2}^{(t)})^T \;\; i = 1, ..., m; \;\; t = 1, ..., T$ be the coordinates of facility i obtained in the t^{th} two phase run where T is the number of two phase runs performed. We run self-organizing map algorithm on these locally optimal facility locations. The first step in this procedure is to form an $n_x \times n_y$ grid over the customer locations. Here n_x is the resolution for x axis and n_y is the resolution for y axis. A cluster centroid is assigned for each rectangle formed by the grids on this topology. Let these cluster centroids be $\mathbf{m}_1, \;\; \mathbf{m}_2, ..., \;\; \mathbf{m}_{n_x \times n_y}$. These points represent initial class centroids. The

algorithm proceeds by taking facility locations one at a time, and finding which cluster in the topology has the closest centroid. The centroid of that cluster, as well as those of the neighboring clusters, are adjusted using the data vector of the facility location under consideration. The adjustment is given by $\Delta \mathbf{m}_j = \eta(\mathbf{x}_i^{(t)} - \mathbf{m}_j)^T \; i = 1, ..., m; \; t = 1, ..., T$ where j is the closest centroid and η is the learning rate that decreases at each iteration step. While changes are made rapidly in the beginning of the algorithm, only small changes are made at the end. All clusters within a radius R are also adjusted to this facility location under consideration. This radius decreases as the calculation progresses as $R = R_{max}(1 - i/N)$ where i is the number of the current iteration step, N is the total number of iteration steps to be performed and R_{max} is defined as $R_{max} = \sqrt{n_x^2 + n_y^2}$. This procedure is repeated for a fixed number of iterations. At the end, all facility locations are assigned to classes and some cluster centroids become inactive since no facility is assigned to them. Therefore, at the end, we have k cluster centroids which are active, i.e., $\mathbf{m}_1, \; \mathbf{m}_2, ..., \; \mathbf{m}_k$ to be used as candidate locations in the formulation of DCMLAP. Then, we use our Lagrangean heuristics to solve the resulting DCMLAP and find an approximate solution for CMFWP. In this procedure, we do not differentiate between the facilities when we are clustering facility locations. An alternative approach could be to make the clusters for each facility separately.

We illustrate our cellular heuristic in Figure 5.1. The customer locations are shown by "•", locally optimal facility locations obtained by two phase runs are shown by "×" and the cluster centroids are depicted by "□". n_x and n_y are 5 and 4 respectively. There were $5 \times 4 = 20$ cluster centroids corresponding to each cell in the grid at the beginning of the algorithm. Figure 5.1 represents an intermediate step of the clustering algorithm. In this step, there are 6 cluster centroids that are active, and the remaining $20 - 6 = 14$ cluster centroids are inactive. The facility location depicted by bold "×" is the current location used to update cluster centroids. The large dashed circle around this facility location determines which cluster centroids will be updated. All clusters within the radius of this circle are adjusted to this facility location under consideration. At the end, we use active cluster centroids as candidate facility locations for DCMLAP.

There are three parameters to be set for this new cellular heuristic. These are the

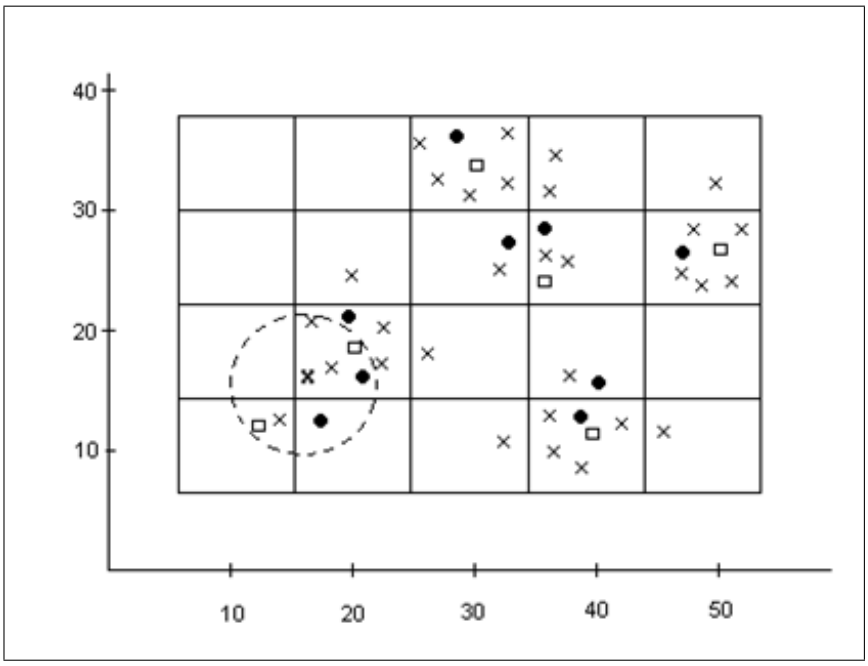

Figure 5.1. Illustration of the cellular heuristic

number of initial two-phase runs and grid resolutions, n_x and n_y. A large number of runs would certainly increase the chance of obtaining good results, but would increase the computational effort. On the other hand, a small number of runs would produce a small number of locally optimal facility locations, therefore inaccurate results. If the generated facility locations in two phase runs do not have a clustering tendency, additional two phase runs are needed. Determination of grid resolutions is even harder than the determination of the number of two phase runs. The selection of n_x and n_y effects the number of clusters obtained at the end of the algorithm, therefore the quality of the solution to DCMLAP. However, we have observed that the number of clusters found is not affected by grid resolution when this resolution becomes large enough. Therefore, we make some trials by increasing n_x and n_y until the number of clusters becomes constant and then solve the cellular heuristic with these clusters found at the end.

One more issue for this heuristic is how to determine initial facility locations for the two phase runs. It should be avoided to start with similar or same initial locations, otherwise the same solutions are produced for a large number of times. Therefore, we should give the same chance to all points to be selected as the initial locations.

6. THOUGHTS ON THE APPLICABILITY OF SOME KNOWN OPTIMIZATION METHODS

As mentioned before, the constraint set of CMFWP represents a transportation polytope (TP), which encourages the usage of network algorithms. It is known that the optimal solution of CMFWP is attained at an extreme point of this TP. This property is also shared by concave minimization (convex maximization) problems where a concave function is minimized over convex sets. Besides that, there is a huge literature on network problems where the cost on each arc in the network is a concave function of the flow on that arc. These problems are called minimum concave cost network flow problems (MCNFP). These observations lead us to develop some methods which can exploit the structure of the transportation polytope by using the methods designed to solve minimum concave cost network flow problems. The literature on the exact solution procedures for MCNFP ranges from branch-and-bound techniques, dynamic programming to extreme point ranking methods. Among these methods, we focus on branch-and-bound techniques, collapsing polytopes method and dynamic programming methods. We also study affine scaling algorithm which is proposed for linear programming problems and try to adapt it for CMFWP by making some changes on the structure of the algorithm. In the following sections, we summarize our approaches which aim at elaborating how these methods can be modified for CMFWP.

6.1. Branch-and-Bound Methods

For a concave minimization problem, there may be many local minima and a local minimum does not have to be a global minimum. Therefore, a solution procedure must search all over these local minima to locate the global minimum. Such a search requirement presents a significant computational burden. This problem is one of the class of problems identified as NP-complete (Jensen, 1980).

A branch-and-bound approach to MCNFP is developed by Florian and Robillard

(1971). This approach assumes that there are capacity restrictions and general concave costs for the arcs. The method is based on the equivalence of this general network flow problem to an uncapacitated network flow problem in a bipartite network of a special form. Since the optimal solution of a concave minimization problem occurs at an extreme point of the feasible region, the counterpart of an extreme point for a network structure is an extremal flow. The purpose of the study is to exploit the structure of these extremal flows in order to construct an implicit enumeration algorithm. The nodes of the branch-and-bound tree corresponds to trials with various arcs forced to having extremal flows. The structure of the extreme flows is exploited by constructing them in a way that prevents the formation of positive loops. It is not necessary to explore all branches in the tree of extremal flows to find the optimal solution. Lower and upper bounds can be used to control the enumeration tree. Lower bounds are found by using a linear underestimation function to the cost of the flow on each arc. Since this cost is a concave function of the flow, the approximating function can be obtained by the straight line that connects the origin with the point corresponding to the maximum flow which can pass over this arc. Figure 6.1 illustrates an example of the concave cost function and the linear underestimation function. When all the cost functions are approximated in this way, lower bounds are found by solving a linear cost transportation problem.

There are many different implementations of the branch-and-bound method for MCNFP (Gallo *et al.*, 1980; Guisewite and Pardalos, 1991; Zangwill, 1968). We observe the same approach for the most of these methods to calculate lower bounds on the optimal objective function value: Lower bounds are found by using a linear underestimation function to the cost of the flow on each arc. Existence of such a linear underestimation function is the advantage of concave cost problems, however we cannot find a closed form underestimation or approximation function for the objective funtion of CMFWP since it is neither concave nor convex. As a result, we have to propose another method to find lower bounds. Sherali and Tuncbilek (1992) were successful to develop a method to calculate lower bounds for the CMFWP with the squared-Euclidean distance. In this method, a linear programming representation of

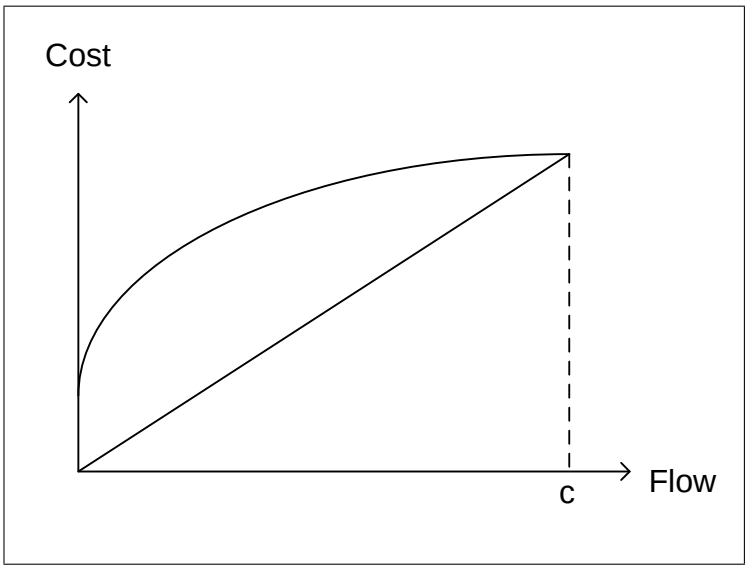

Figure 6.1. An example of a concave cost function and its linear underestimation function

the original problem is utilized to compute lower bounds via a Lagrangean relaxation scheme. The special formulation specific to the squared-Euclidean CMFWP makes such a representation and scheme possible, since there exists a closed form expression for allocation variables as a function of facility coordinates. However, there is not such a specific formulation for other l_p distance problems, therefore the scheme proposed for the squared-Euclidean CMFWP is not applicable to other problems. Nevertheless, Sherali *et al.* (1994) propose another approach to calculate lower bounds for l_p distance problems in general. This method is not based on a special formulation of the original problem, but on designing some bounding intervals for the allocation variables. Some logical tests are developed to determine tighter bounds on the allocation variables w_{ij} based on the structure of the transportation constraints. These bounds are represented by l_{ij} and u_{ij} for lower and upper bounds respectively. Since $w_{ij} \geq l_{ij}$ for all (i, j), the lower bound Z_{LB} on the problem can be calculated via the following location problem:

$$Z_{LB} = min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} l_{ij} d(\mathbf{x}_i, \mathbf{a}_j).$$

This problem can be solved separably for each facility i for $i = 1, ..., m$ using any procedure for the single facility location problem such as Weiszfeld procedure.

Our main purpose at this step is to develop another method to calculate lower bounds for CMFWP more effective and more efficient than the ones in the literature. We decided to focus on dual formulation of our problem. Dual formulation can be helpful since the dual of a minimization problem is a maximization type, therefore the objective value of any dual feasible solution can be a lower bound for the original problem.

The dual formulation of the multi-facility location-allocation problem with l_p distance is given by Love (1974). The dual in this work is formulated for a set of given allocations. In other words, the dual corresponding to each feasible allocation is different. It is clear that when feasible allocation variables are given for CMFWP, the constraints of the problem are satisfied. Then, the dual in fact is written for the objective function of CMFWP with given feasible allocations. The objective function whose dual is formulated is $z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} w_{ij} d(\mathbf{x}_i, \mathbf{a}_j)$ where $d(\mathbf{x}_i, \mathbf{a}_j)$ can be any l_p distance function with $p > 1$. By using quasilinearization, the dual of this problem can be formulated as the following:

DF:

$$\max\ g(\mathbf{U}) = -\sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{a}_j \mathbf{U}_{ij}^T \tag{6.1}$$

s.t.

$$\sum_{j=1}^{n} \mathbf{U}_{ij}^T = \mathbf{0} \qquad i = 1, \ldots, m \tag{6.2}$$

$$\left\|\mathbf{U}_{ij}^T\right\|_q \leq w_{ij} \qquad i = 1, \ldots, m;\ j = 1, \ldots, n \tag{6.3}$$

where $q = p/(p-1)$ and $\mathbf{U}_{ij} = (u_{ij1}, u_{ij2})$. $\mathbf{U}_{ij}$'s are dual variables.

The problem above is a nonlinear programming problem with a linear objective function. Since (6.2) is linear and (6.3) is convex ($\left\|\mathbf{U}_{ij}^T\right\|_q$ is convex in $\mathbf{U}$ (Love and Juel, 1982)), the constraint set is also a convex set. With the help of these properties and by using a theorem for nonlinear programming problems due to Hillier and Lieber-

man (1967), (Love and Juel, 1982) state the location-allocation problem as a concave minimization problem. Using the dual formulation above, CMFWP can be formulated as:

$$\min G^*(\mathbf{w}) \tag{6.4}$$

s.t.

$$\sum_{i=1}^{m} w_{ij} = d_j \qquad j = 1, \ldots, n \tag{6.5}$$

$$\sum_{j=1}^{n} w_{ij} = s_i \qquad i = 1, \ldots, m \tag{6.6}$$

$$w_{ij} \geq 0 \qquad i = 1, \ldots, m;\ j = 1, \ldots, n \tag{6.7}$$

where $G^*(\mathbf{w})$ is the optimal objective function value of the problem DF solved for given allocations $\mathbf{w}$.

Here $G^*(\mathbf{w})$ is concave and the problem becomes a concave minimization problem. It tries to find those w_{ij}'s which minimize $G^*(\mathbf{w})$. We tried to make use of this formulation to calculate lower bounds within a branch-and-bound algorithm. At some step of this algorithm, we are at a node where some partial information about w_{ij}'s are known, such as some allocations are fixed, or there are some lower or upper bounds on w_{ij}'s and so on. We try to solve a problem in undetermined allocation variables in the existence of some partial information available for some of them. In DF given above, the dual is formulated for given w_{ij}'s. However, we do not have fixed and given allocations in any node of the tree except the leaf nodes. Therefore, we must come up with a method to solve the dual problem with unknown w_{ij}'s. We worked on this a little bit further by investigating other lower bound calculating methods (Love and Yeong, 1981; Love and Dowling, 1989; Drezner, 1984; Wendell and Peterson, 1984) on the objective function of facilities location problems. Among these methods, the one proposed by Love and Dowling (1989) is more relevant for our problem since it is

developed for multi-facility location problems. The problem is given by:

$$\min WM_p = \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} l_p(\mathbf{x}_i, \mathbf{a}_j) \tag{6.8}$$

where $l_p(\mathbf{x}_i, \mathbf{a}_j)$ denotes l_p distance between facility i and customer j and w_{ij} is the quantity allocated from facility i to customer j. Then the following inequality always holds:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} l_p(\mathbf{x}_i, \mathbf{a}_j) \geq \sum_{i=1}^{m}\sum_{j=1}^{n} w'_{ij}|x_{i1} - a_{j1}| + \sum_{i=1}^{m}\sum_{j=1}^{n} w''_{ij}|x_{i2} - a_{j2}| \tag{6.9}$$

where $w'_{ij} = w_{ij}|x^k_{i1} - a_{j1}|^{p-1}/l(x^k_i, a_j, p, q)$, $w''_{ij} = w_{ij}|x^k_{i2} - a_{j2}|^{p-1}/l(x^k_i, a_j, p, q)$ and $l(x^k_i, a_j, p, q) = [|x^k_{i1} - a_{j1}|^p + |x^k_{i2} - a_{j2}|^p]^{1/q}$ and $q = p/(p-1)$. Here $\mathbf{x}^k_i$ is any given location for facilities. The right hand side of the inequality (6.9) is a rectangular distance multi-facility location problem. An optimal solution to this problem can be used to yield a lower bound for WM_p.

The drawback of this method (and also the other methods mentioned above) is the same with the dual formulation: It finds lower bounds when w_{ij}'s are given. This is reasonable for these methods, because they are used to find a stopping criterion for iterative solution methods on single-facility location problems. As a result, we could not find a method to calculate lower bounds by using the dual formulation. That makes the implementation of a branch-and-bound method based on the dual formulation impossible. Therefore, we change direction towards to other solution techniques, such as concave minimization via collapsing polytopes which is described in detail in the following section.

6.2. Collapsing Polytopes Method

Falk and Hoffman (1986) propose a procedure to minimize globally a concave function over a bounded polytope. It is known that transportation polytope is a bounded convex polytope (Bolker, 1972), but the objective function of CMFWP is not

a concave function. However, this fact does not restrict us to apply this method on our problem, since concavity of the objective function is required only to guarantee a vertex solution and we know that the optimal solution of CMFWP is attained at an extreme point of the feasible region. Therefore, we can try to apply the method of collapsing polytopes on CMFWP. The general problem which Falk and Hoffman address has the following form:

$$\min\ f(\mathbf{x}) \tag{6.10}$$

$$\text{s.t. } A\mathbf{x} \leq \mathbf{b} \tag{6.11}$$

where f is a concave function defined over R^n, and A is an $m \times n$ matrix with $m > n$. A_i denotes the ith row of A, and $\|A_i\|$ denotes its Euclidean norm. The proposed method has the following assumptions:

1. The set $S = \{x : Ax \leq b\}$ is bounded.
2. S has a nonempty interior.
3. There are no degenerate basic solutions of S.
4. Any basic solution of the system $C = \{v = (x, y) : Ax + ay \leq b, y \geq 0\}$ is nondegenerate where $a = (\|A_1\|, ..., \|A_m\|)^T$ and y is a scalar variable.
5. The solution of the linear program

$$\max\ y \tag{6.12}$$

$$\text{s.t. } Ax + ay \leq b \tag{6.13}$$

   is unique.
6. The constraints $x \geq 0$ are among those defining S.

Our first attempt is to check if these assumptions are satisfied by CMFWP. The first and last assumptions are trivially satisfied, but the second one is not fulfilled if the problem is balanced, i.e., total supply is equal to total demand. We assume that the problem is balanced when we give the mathematical formulation in Chapter 2, but this

assumption is not necessary for the heuristic methods we propose, therefore we can relax it and add a dummy facility with a positive supply to satisfy the second assumption. When this dummy facility is added with a high unit transportation cost per unit distance, CMFWP becomes unbalanced, i.e., total supply exceeds total demand, but this makes the assumption that S has a nonempty interior satisfied. The addition of the dummy facility does not create a change for the optimal solution, since this facility is not used in the optimal solution due to its high transportation cost anyway, and we can remove it from the solution at the end. Besides the addition of the dummy facility, we also change equalities in the problem to "$\leq$" type to obtain the same problem structure given by Falk and Hoffman. Then the problem on which we will apply the method of collapsing polytopes becomes as the following:

$$\min z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}w_{ij}d(\mathbf{x}_i, \mathbf{a}_j) + \sum_{j=1}^{n} M w_{(m+1)j} d(\mathbf{x}_{m+1}, \mathbf{a}_j) \tag{6.14}$$

s.t.

$$-\sum_{i=1}^{m+1} w_{ij} \leq -d_j \qquad j = 1, \ldots, n \tag{6.15}$$

$$\sum_{j=1}^{n} w_{ij} \leq s_i \qquad i = 1, \ldots, m \tag{6.16}$$

$$\sum_{j=1}^{n} w_{(m+1)j} \leq s_{m+1} \tag{6.17}$$

$$-w_{ij} \leq 0 \qquad i = 1, \ldots, m+1;\ j = 1, \ldots, n \tag{6.18}$$

where $m+1$ and $s_{m+1} > 0$ is the index and the supply of the dummy facility respectively and M is an arbitrarily large positive number. We denote the feasible region of this problem by S.

We cannot give a certain conclusion about the other assumptions, especially for assumptions 3 and 4, since whether they are satisfied or not depends on problem parameters and cannot be checked a priori. However, transportation polytopes are mostly and highly degenerate, therefore we should expect problems during the implementation of the algorithm to solve the CMFWP. On the other hand, Falk and Hoffman say that

problems that do not satisfy these assumptions could still be solved successfully if such degenerate vertices do not require attention during the execution of the method.

Let problem CP be

$$\max\ y \tag{6.19}$$

s.t.

$$-\sum_{i=1}^{m+1} w_{ij} + \sqrt{m+1}y \leq -d_j \qquad j = 1, \ldots, n \tag{6.20}$$

$$\sum_{j=1}^{n} w_{ij} + \sqrt{n}y \leq s_i \qquad i = 1, \ldots, m+1 \tag{6.21}$$

$$-w_{ij} + y \leq 0 \qquad i = 1, \ldots, m+1;\ j = 1, \ldots, n \tag{6.22}$$

Let r denote the vector of slack variables for CP. The optimal solution of CP yields a point $\mathbf{w}^0 \in S$ that is the center of the largest sphere contained in S. The value y^0 is the radius of this sphere. First step in the algorithm is to identify neighbors $(\mathbf{w}^{0,i}, y^{0,i}, \mathbf{r}^{0,i})$ of $(\mathbf{w}^0, y^0, \mathbf{r}^0)$. In order to generate the first enclosing polytope S_0 of S, we need to extend the rays beginning at $(\mathbf{w}^0, y^0)$ and passing through the neighbors $(\mathbf{w}^i, y^i)$ until these rays intersect the hyperplane $y = 0$. At stage k, we will have a tree T_k whose terminal nodes $v = (\mathbf{w}^t, y^t)$ will correspond with vertices of C. There will be a value z_t corresponding to the objective function z evaluated at a point associated with $\mathbf{v}^t$ associated with each of these nodes. We chose the vertex $\mathbf{v}^t$ that has the minimum associated z_t value over all such terminal vertices. If the point associated with $\mathbf{v}^t$ is in S, then it is optimal. If this point is not in S, we identify neighbors of $\mathbf{v}^t$ satisfying $y^{t,i} < y^t$ and is not a member of the tree associated with stage k. As before, we extend the rays emanating from $\mathbf{v}^t$ through these neighbors until they intersect the hyperplane $y = 0$. This process continues in this manner until the stopping condition is satisfied. The method can be summarized formally in Algorithm 7.

We decided to implement the method of collapsing polytopes on CMFWP and apply it on a small instance in order to observe if the unsatisfied assumptions are problematic or not. An important issue to be resolved is the calculation of objective function

Algorithm 7 The method of collapsing polytopes for CMFWP

1. Solve CP and obtain solution $\mathbf{v}^0 = (\mathbf{w}^0, y^0, \mathbf{r}^0)$. Set $z_0 = \infty$. This solution builds up and initializes the tree of vertex solutions $T_0 = \{(\mathbf{v}^0, z_0)\}$. Set $k = 0$.
2. With the current T_k at stage k, select a terminal node $\mathbf{v}^t$ whose associated value z_t is minimum of all terminal nodes. If the associated $y_t = 0$, stop with global optimal solution $\mathbf{w}^* = \mathbf{w}^t$ and $z^* = z_t$.
3. Identify the neighbors $\mathbf{v}^{t,i}$ of $\mathbf{v}^t$ satisfying
   a) $y^{t,i} < y^t$
   b) $\mathbf{v}^{t,i} \notin T_k$
4. For each such $\mathbf{v}^{t,i}$, compute the point $\mathbf{f}^{t,i}$, i.e., the extension of the ray from $\mathbf{v}^t$ through $\mathbf{v}^{t,i}$ where it pierces the plane $y = 0$. Compute $z_{t,i}$, the objective function value of the point $\mathbf{f}^{t,i}$. The tree T_{k+1} is the tree T_k with the new nodes $\mathbf{v}^{t,i}$ satisfying *a*) and *b*), and with the links joining $\mathbf{v}^t$ to those new $\mathbf{v}^{t,i}$. Set $k \leftarrow k + 1$ and go to step 2.

value when we implement this method on CMFWP. The method first determines which vertices to take into consideration at some step and then finds objective values corresponding to those vertices. This means that we can calculate objective function value for a vertex by first finding the optimal facility locations corresponding to this vertex, and then calculating distance matrix between facilities and customers. When distance matrix is found, the resulting objective function becomes linear in allocation variables and its value can be calculated with given allocations and cost multipliers. Now, we can illustrate the algorithm on a small instance of CMFWP. However, in order to keep necessary calculations at the minimum, we decided to implement the algorithm on the transportation problem. This can also help us to concantrate on those issues related to degeneracy of transportation polytope, if they exist. In the instance, there are 4 facilities (one of which is dummy) and 4 customers whose supplies and demands are given in Table 6.1 where F denotes facilities and C denotes customers. Unit shipment costs between each facility and customer are also given. The distance metric is the Euclidean distance.

The first step in the algorithm finds an initial point to start which solves the

Table 6.1. An instance of CMFWP for collapsing polytopes method

	Demand	58	46	42	14
Supply	c_{ij}	C1	C2	C3	C4
61	F1	1	5	1	2
69	F2	4	2	4	2
30	F3	3	2	4	1
40	F4	1000	1000	1000	1000

problem

$$\max\ y \tag{6.23}$$

s.t.

$$-\sum_{i=1}^{4} w_{ij} + 2y \leq -d_j \qquad j = 1, ..., 4 \tag{6.24}$$

$$\sum_{j=1}^{4} w_{ij} + 2y \leq s_i \qquad i = 1, ..., 4 \tag{6.25}$$

$$-w_{ij} + y \leq 0 \qquad i = 1, ..., 4;\ j = 1, ..., 4 \tag{6.26}$$

When we solve this problem, we find the initial point $\mathbf{w}^{(0)}$ given in Table 6.2 with $y^{(0)} = 2.5$.

Table 6.2. Initial point of the example for collapsing polytopes method

w_{ij}	C1	C2	C3	C4
F1	2.5	43.5	7.5	2.5
F2	30.5	2.5	19.5	11.5
F3	2.5	2.5	17.5	2.5
F4	27.5	2.5	2.5	2.5

Next step is finding $m \times n + 1 = 17$ neighbors of the initial point by pivoting in those 17 components of $\mathbf{r}^{(0)}$ that are zero using the minimum ratio rule. However, we focus the ones satisfying conditions a) and b) given in step 3 of Algorithm 7. All 17 neighbors satisfy these conditions. Now, we have to generate the first enclosing

polytope S_0 by extending the rays beginning at $(\mathbf{w}^{(0)}, y^{(0)})$ and passing through the neighbors until these rays intersect the hyperplane $y = 0$. These points on $y = 0$ may or may not be feasible to S. Then, we calculate objective function values of these points and choose the neighbor of $(\mathbf{w}^{(0)}, y^{(0)})$ corresponding to the point with minimum objective value. The minimum objective value becomes $z^{(0)} = 613$ for the point $\mathbf{w}^{(1)}$ given in Table 6.3 with $y^{(1)} = 0.375$.

Table 6.3. The point in the first step of the example for collapsing polytopes method

w_{ij}	C1	C2	C3	C4
F1	0.375	45.625	13.875	0.375
F2	53.875	0.375	0.375	13.625
F3	0.375	0.375	28.125	0.375
F4	4.125	0.375	0.375	0.375

Since $y^{(1)} > 0$, we cannot stop. We perform the same operations described above on this point and at the end of step 2, we find the point in Table 6.4 as the one with minimum corresponding objective value $z^{(1)} = 610$ and $y^{(2)} = 0$.

Table 6.4. The point in the second step of the example for collapsing polytopes method

w_{ij}	C1	C2	C3	C4
F1	0	46	15	0
F2	58	0	0	11
F3	0	0	27	3
F4	0	0	3	0

Since $y^{(2)} = 0$, we have to stop, however the point in Table 6.4 is not the optimal solution. When we investigate the reasons, we have seen that this point is a degenerate point. This means that assumption 3 is violated as we expect. Besides that, the method collapses the face of the polyhedron C where $y = 0$ before it finds optimal solution. We made other experiments on this instance by putting an extra rule to the algorithm in order to make it advance: we keep iterating although the stopping condition $y = 0$

is satisfied. However, new solutions are always on the hyperplane where $y = 0$ and the algorithm continues to give the same objective function value as the last iteration, i.e., $z = 610$. This is because of the fact that neighbors which the method finds for the degenerate point is again the same point which has a different basis structure, but with the same objective function value due to degeneracy. As a result, we could not come up with a solution procedure to solve this degeneracy issue. So, we have decided to discard collapsing polytopes method and concentrate on another method.

6.3. Send-and-Split Method

Send-and-split method is a dynamic programming approach to find a minimum cost flow in a network where the flow cost is additive and concave in the flow in each arc (Erickson *et al.*, 1987). This method makes use of the property that a minimum cost flow is an extreme flow and searches these extreme flows to find one with minimum cost.

Let $G = (N, A)$ be a directed graph of a set N of n nodes and a set A of a arcs. D denotes the set of demand nodes (sources and sinks) that have nonzero demands within the set N and c_{ij} is the cost on arc (i, j). Subproblem $i \rightarrow I$ is constructed by setting $d_j = 0,\ j \in N \backslash I$ and replacing d_i by $d_i = d_i - \sum_{j \in I} d_j$ where I is a subset of D. We can view this subproblem as satisfying only the demands of nodes in set I when we changed the demand of node i to some type of preflow which can be positive or negative depending on I. Let C_{iI} denote the minimum cost solution to subproblem $i \rightarrow I$ and I_j denote $I \backslash j$. Then we find:

- If $\sum_{k \in I} d_k = 0$ then $C_{iI} = C_{jI_j},\ j \in I$.
- If $d_I = \sum_{k \in I} d_k \neq 0$ then $C_{iI} = \min\{\min_{(i,j) \in A_I}[c_{ij}(d_I) + C_{jI}], B_{iI}\}$ where A_I is A if $d_I > 0$ or A with the arc directions reversed if $d_I < 0$ and for $|I| > 1$, $B_{iI} = \min_{\emptyset \subset J \subset I}[C_{iJ} + C_{i,I \backslash J}]$ and for $|I| = 1, B_{iI} = 0$ if $I = \{i\}$ and $B_{iI} = +\infty$ otherwise.

The procedure described above is called send-and-split method due to the fact

that the minimum solution of a subproblem results from performing either sending or splitting operations. Sending operation is to send all flow from node i to node j with a cost of $c_{ij}(d_I)$, then solving the subproblem $j \rightarrow I$ which results in a total cost of $c_{ij}(d_I) + C_{jI}$. This can occur for any node j. Splitting operation is to split the flow at node i to satisfy the requirements in sets J and $I \backslash J$, for any subset $\emptyset \subset J \subset I$. Once the subproblems are solved for all subsets of I, two decisions have to be made at each node i based on these operations. The first one is to solve the set-splitting problem of finding a minimum-cost split of I into two nonempty subsets J and $I \backslash J$, and the cost of split is the sum of the minimum costs for the subproblems $i \rightarrow J$ and $i \rightarrow I \backslash J$. The second decision is to find a minimum cost chain along which to ship the demand at I from each node i to a node at which it is optimal to incur the cost of splitting I into subsets. Then the optimal solution for subproblem $i \rightarrow I$ becomes the minimum of these operations. The desired minimum cost over all flows for the original problem is C_{iD} for all $i \in N$ and the idea of the algorithm is to solve the subproblems inductively on the cardinality of I and to find the minimum cost flow when $I = D$.

The algorithm described above is not applicable to our problem with the form given above. The objective function of our problem is $z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} w_{ij} d(\mathbf{x}_i, \mathbf{a}_j)$, this means that we do not have additive costs for each arc due to the existence of the term $d(\mathbf{x}_i, \mathbf{a}_j)$. If the locations of the facilities were known, then the distance between a facility and a customer ($d(\mathbf{x}_i, \mathbf{a}_j)$) would also be known and our problem would reduce to a linear cost (which is both convex and concave) transportation problem. However, the locations of the facilities are unknown in our problem, this means that $d(\mathbf{x}_i, \mathbf{a}_j)$'s are also decision variables whose values have to be set. Therefore, CMFWP does not satisfy the assumption that total flow cost is additive in the flow on each arc. In order to restore this assumption back, we decided to apply a different procedure. While solving a subproblem $i \rightarrow I$, we can locate a facility in I (or i if it is a facility) with respect to the customers in I. In this case, we are solving a single facility location problem whose solution is quite easy with respect to multi-facility case. The location of a facility can be determined by using an iterative solution approach such as Weiszfeld's procedure. When the location of facility i is found, then $d(\mathbf{x}_i, \mathbf{a}_j)$ terms in the objective function becomes known and the cost becomes additive on each arc. It is useful to illustrate

send-and-split algorithm with this procedure on a small instance of CMFWP.

There are 2 facilities (nodes 1 and 2) and 3 customers (nodes 3, 4 and 5) whose supplies (negative) and demands (positive) are given in the network in Figure 6.2 below. Customer locations are also given next to each customer. The distance metric used is the squared Euclidean distance.

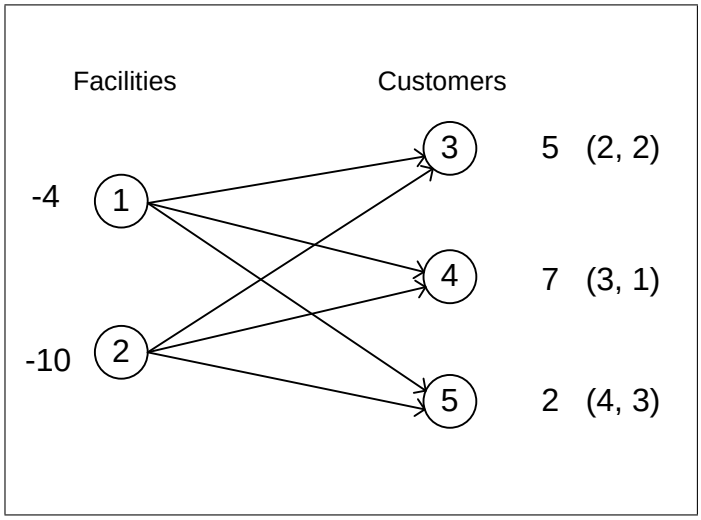

Figure 6.2. An instance of CMFWP illustrated on a network structure

We have to solve all subproblems $i \rightarrow I$ beginning with subsets I whose cardinalities are equal to 1. Since there are 5 demand nodes (both facilities and customers), there are $2^5 - 1 - 1 = 30$ subsets of those demand nodes and therefore we have to solve $5 \times 30 = 150$ subproblems.

Before performing iterations for this example, it is quite useful to mention a point: It is clear that set I may consist of both customers and facilities. When i is a facility and set I also includes another (or the same) facility, say j, existence of the solution to this subproblem depends on the sign of $d_{I\setminus\{i\}}$. There are two possible cases : $d_{I\setminus\{i\}} > 0$ or $d_{I\setminus\{i\}} < 0$. When $d_{I\setminus\{i\}} > 0$, then $d_i = -d_{I\setminus\{i\}}$ becomes negative, therefore i behaves like a supply node for this subproblem. Since i was originally a supply node and arcs are directed such that their tails are incident to facility i, there exists a solution to this subproblem. In the other case where $d_{I\setminus\{i\}} < 0$, d_i becomes positive and i begins to behave like a demand node. Then we cannot find a solution to this problem since

it is not possible to satisfy the requirement of facility i with given network structure (arcs are not directed toward i to make a shipment to i possible). The optimal solution of such subproblems are denoted by ∞ which means that a solution does not exists for this subproblem. The same thing is also relevant for the subproblems where i is a customer and set I consists of another (or the same) customer. In this case, a solution exists if $d_{I\setminus\{i\}} < 0$ and optimal solution is ∞ if $d_{I\setminus\{i\}} > 0$.

Initial step in the solution of each subproblem is locate facilities with respect to customers in I. For subproblems where $|I| = 1$, this is quite easy since we locate a facility on top of the customer in I (or locate the facility in I on top of customer i). In this case, the distance between facility i and customer in I becomes zero, therefore the cost of sending for these problems becomes zero since we try to satisfy the requirement of the customer in I from facility i whose distance to this customer is zero. Then the optimal solution becomes also zero although the cost of splitting may be ∞ when $|I| = 1$ and $I \neq \{i\}$. However, if i and the node in I are both facilities or both customers, the optimal solution becomes ∞ based on the argument given in the preceding paragraph. Table 6.5 summarizes C_{iI} values for all $|I| = 1$ and subsets $\{i\}$.

Table 6.5. Optimal solutions for subproblems where $|I| = 1$

I	d_I	C_{1I}	C_{2I}	C_{3I}	C_{4I}	C_{5I}
$\{1\}$	-4	0	∞	0	0	0
$\{2\}$	-10	∞	0	0	0	0
$\{3\}$	5	0	0	0	∞	∞
$\{4\}$	7	0	0	∞	0	∞
$\{5\}$	2	0	0	∞	∞	0

Since the results obtained for $|I| = 1$ seems realistic up to now, we can continue with $|I| = 2$. There are 10 subsets with cardinality 2. There are different subproblem formations at this step depending on the content of i and I. We elaborate each case below.

- For a subproblem where I consists of just facilities and i is a customer, for example

$3 \rightarrow \{1,2\}$, we can locate facilities in I on top of customer i to satisfy the requirement of i, therefore the cost of sending becomes zero for such subproblems, since the distance between facilities and the customer becomes zero. In order to find the optimal solution, we should also find the cost of splitting, i.e., B_{iI}. Since I just consists of two facilities, when we split I, we obtain two subproblems, i.e., $3 \rightarrow \{1\}$ and $3 \rightarrow \{2\}$. Total cost of this split is also zero since $C_{3\{1\}}$ and $C_{3\{2\}}$ are both zero from Table 6.5. As a result, optimal solution of this subproblem ($C_{3\{1,2\}}$) becomes zero.

- For a subproblem where I consists of a facility and a customer and i is a facility, for example $2 \rightarrow \{1,3\}$, the cost of sending is found as in the previous case and therefore becomes zero. However, the cost of split becomes ∞ in this case since the cost of the subproblem $2 \rightarrow \{1\}$ is ∞. Nevertheless, optimal solution becomes zero due to $min\{0,\infty\} = 0$.
- In a subproblem where I consists of a facility and a customer and i is a customer, for example $4 \rightarrow \{2,3\}$, there are two customers and a single facility. Therefore, we should find the location of facility 2 using Weiszfeld's procedure. Since the requirements of both customer 3 and 4 is 5 for this subproblem, facility 2 is located in the middle of the line segment joining 3 and 4 and the distance to both 3 and 4 becomes 0.5. In order to find the cost of sending, we should find a minimum cost chain, and solve $min_{(4,j)\in A_I}[c_{4j}(d_I) + C_{jI}]$. The minimum is attained for $j = 2$ and the cost of sending becomes $c_{42}(5) + C_{2\{2,3\}} = 0.5 \times 5 + 0 = 2.5$. However, this solution is not correct, because the real cost of sending for this subproblem must be the sum of the costs of sending from facility 2 to customer 3 and facility 2 to customer 4, i.e., $5 \times 0.5 + 5 \times 0.5 = 5$. This difference is due to our solution procedure used to overcome the problem of nonadditive arc costs. When we locate facility 2 between customers 3 and 4, total flow cost found by the algorithm becomes just the cost of sending from facility 2 to customer 3 since the cost of sending from facility 2 to customer 4 is zero. It is zero because while we are solving subproblem $2 \rightarrow \{2,4\}$, we locate facility 2 on top of customer 4 and therefore the distance and the cost becomes zero for this subproblem. Although this observation lead us to conclude that our solution procedure does not work as we expected, we continue to solve subproblems in accordance with send-and-

split method. In other words, we took 2.5, not 5 as the cost of sending of the previous subproblem which is found by send-and-split method. Table 6.6 shows the optimal solutions of the other subproblems where $|I| = 2$.

Table 6.6. Optimal solutions for subproblems where $|I| = 2$

I	d_I	C_{1I}	C_{2I}	C_{3I}	C_{4I}	C_{5I}
{1,2}	-14	∞	∞	0	0	0
{1,3}	1	0	0	0	∞	∞
{1,4}	3	0	0	∞	0	∞
{1,5}	-2	0	0	∞	∞	0
{2,3}	-5	∞	0	0	2.5	6.25
{2,4}	-3	∞	0	1.5	0	3.75
{2,5}	-8	∞	0	10	10	0
{3,4}	12	0	0	∞	∞	∞
{3,5}	7	0	0	∞	∞	∞
{4,5}	9	0	0	∞	∞	∞

Since the optimal solutions of all subproblems with $|I| = 2$ are found, we can continue with $|I| = 3$. There are 10 subsets with cardinality 3. There are again different subproblem formations at this step depending on the content of i and I, but it is better to give attention to the ones where I and i together consist of more than one facility, for example $1 \rightarrow \{2, 3, 4\}$. This subproblem has two facilities to be located. We can locate each facility separately with respect to the requirements of customers 3 and 4, however this approach disregards the fact that these facilities does not satisfy those requirements used to find locations in the optimal solution of this 2-facility, 2-customer subproblem. Therefore, the locations found become nonoptimal, and we have to solve a 2-facility, 2-customer location problem to find the location of these facilities. This means that we cannot obtain any benefit from the dynamic programming approach if we have to solve a multi-facility location problem optimally. As a result, we decided not to implement send-and-split method for CMFWP since all these results make the operation of our solution procedure impossible. Therefore we conclude that send-and-split method cannot be adapted to solve CMFWP.

6.4. Affine Scaling Methods

Affine scaling is an interior point algorithm proposed by Dikin (1967) for solving linear programming problems. It solves problems by following a pathway toward optimality through the interior of the feasible region. It is a variant of Karmarkar's well-known interior point algorithm and has different forms depending on whether it is applied on primal or dual form of the problem. These methods are generally applied on linear programs in the standard form as the following:

$$\begin{aligned} &\text{P: Min } \mathbf{c}^T\mathbf{x} \\ &\text{s.t. } A\mathbf{x} = \mathbf{b} \\ &\mathbf{x} \geq 0 \end{aligned}$$

where P denotes the primal and

$$\begin{aligned} &\text{D: Max } \mathbf{y}^T\mathbf{b} \\ &\text{s.t. } \mathbf{y}^T A + \mathbf{s} = \mathbf{c} \\ &\mathbf{s} \geq 0 \end{aligned} \tag{6.27}$$

where D denotes the dual form of the standard formulation.

Affine scaling algorithms have taken the attention of many researchers in the solution of linear transportation problems (Resende and Veiga, 1993; Portugal *et al.*, 1993). As stated earlier, feasible region of the CMFWP is a transportation polytope, however CMFWP is not a transportation problem, nor it is a linear programming problem since it has a nonlinear objective function. Therefore, we cannot apply affine scaling algorithm in its original form. However, we could design a procedure which incorporates affine scaling algorithm in itself and has some extra update rules to overcome the assumption that the objective function is linear. Our main purpose to apply affine scaling method on CMFWP is to see whether the method converges to the optimal extreme point at the end. We now describe this method in more detail.

Transportation problem can be represented in the form of standard linear programs P and D given above. In order to make A full rank, we have to drop a constraint from TP since it has one redundant constraint if the problem is balanced, i.e., total supply is equal to total demand. Therefore, at the end, we come with a TP where $b \in R^{m+n-1}, x, c \in R^{m \times n}$ and $A \in R^{(m+n-1)\times(m\times n)}$ with full rank in primal form and $y \in R^{m+n-1}$ and $s \in R^{m\times n}$ in dual form. The following algorithm summarizes the steps of affine scaling method applied on the primal form of CMFWP.

Algorithm 8 Primal affine scaling algorithm

1. Pick an interior point $\mathbf{x}^{(0)}$ such that $A\mathbf{x}^{(0)} = \mathbf{b}$ and $\mathbf{x}^{(0)} > 0$. Set $k = 0$.
2. Calculate affine transformation vector $\mathbf{y}$ as $\mathbf{y} = (D^{(k)})^{-1}\mathbf{x}^{(k)}$ where $D^{(k)} = \text{diag}\{x_1^{(k)}, ..., x_{(m\times n)}^{(k)}\}$.
3. Find optimal facility locations corresponding to allocation vector $\mathbf{x}^{(k)}$ and calculate the distance from each facility to each customer.
4. Define the gradient of the objective function $\bar{\mathbf{c}} = \mathbf{c}\mathbf{d}D^{(k)}$ where $\mathbf{d} \in R^{m\times n}$ is the distance vector.
5. Find $\mathbf{C}_p = [I - (AD^{(k)})^T[A(D^{(k)})^2A^T]^{-1}(AD^{(k)})]\bar{\mathbf{c}}^T$.
6. If $\mathbf{C}_p$=$\mathbf{0}$, then STOP. The current solution $\mathbf{x}^{(k)}$ is optimal for P. Otherwise, find a direction vector $\mathbf{r}^{(k)}$ and a step size λ to update $\mathbf{x}^{(k)}$ by using $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda\mathbf{r}^{(k)}$ where $\mathbf{r}^{(k)} = -D^{(k)}\mathbf{C}_p$. λ takes values in the interval $(0, \lambda_{max})$ where $\lambda_{max} = \text{minimum}\left\{\frac{x_j^{(k)}}{-(r_j^{(k)})} : r_j^{(k)} < 0\right\}$. Generally λ is updated as $\lambda = \alpha\lambda_{max}$ where $0 < \alpha < 1$, but in practice $0.95 < \alpha < 0.99$.

Affine scaling algorithm, in its original form, generates a sequence of vectors $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ converging to $\mathbf{x}^*$, the optimal solution of P. However, we are not sure if the method described above can expose such behavior since we do not have constant cost multiplier vector $\mathbf{c}$, but we update $\mathbf{c}$ in each iteration due to the change in the location of facilities in that iteration. This may cause the direction vector $\mathbf{r}^{(k)}$ to show an unstable pattern. Figure 6.3 and Figure 6.4 show the behavior of original affine scaling algorithm applied to a linear transportation problem and expected behavior of our method applied to CMFWP, respectively. First one shows a smoother convergence behavior toward the optimal extreme point $\mathbf{x}^*$, while we expect the second one to show

an unstable behavior as in Figure 6.4 due to change in $\mathbf{c}$ in each iteration.

We are not able prove that our procedure converges to the optimal extreme point as in Figure 6.4. We will see if this occurs by performing some experiments. In the experiment, we use a test problem with 2 facilities and 4 customers whose supplies and demands are given in Table 6.7 where F denotes facilities and C denotes customers. Coordinates of each customer are also given. The distance metric used is the Euclidean distance.

Table 6.7. An instance of CMFWP for affine scaling algorithm

	Supply/Demand	Coordinates
F1	5	-
F2	21	-
C1	5	(1,4)
C2	7	(1,1)
C3	8	(2,2)
C4	6	(3,1)

Optimal solution of the problem in terms of w_{ij}'s is given in Table 6.8.

Table 6.8. Optimal solution of the instance for affine scaling algorithm

w_{ij}	C1	C2	C3	C4
F1	5	0	0	0
F2	0	7	8	6

In the first step, we have to choose an initial interior point. After that, we apply the algorithm until it converges. Table 6.9 summarizes the results of some experiments which we report both the initial interior point and the final extreme point where the algorithm converges at the end.

When we look at the results in Table 6.9, we see that the final solution depends very much on the initial point. The method converges to an extreme point where the initial point is closer to. Therefore, we can conclude that the proposed procedure cannot find the global minimum. Nevertheless, we can use this procedure as a heuristic if the

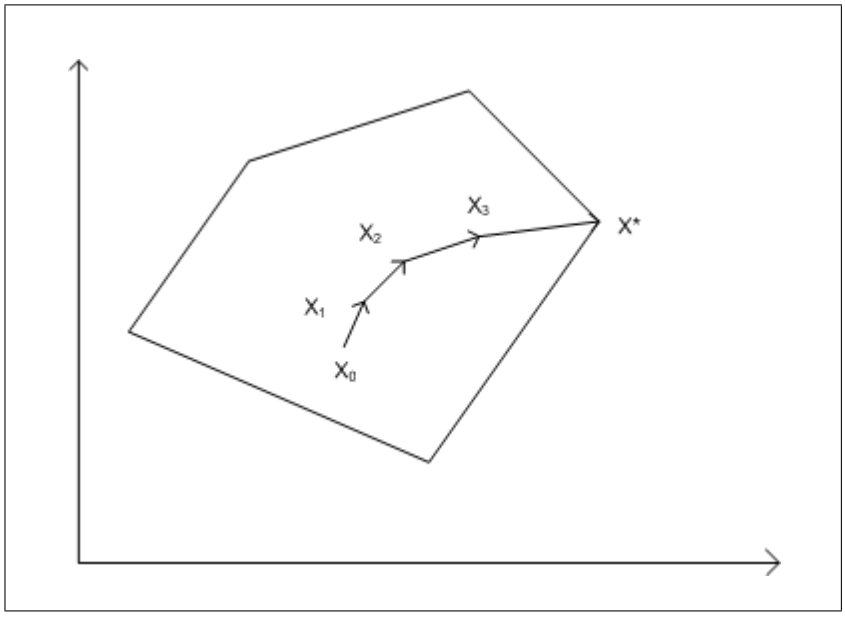

Figure 6.3. Convergence behavior of the affine scaling algorithm applied on linear transportation problem

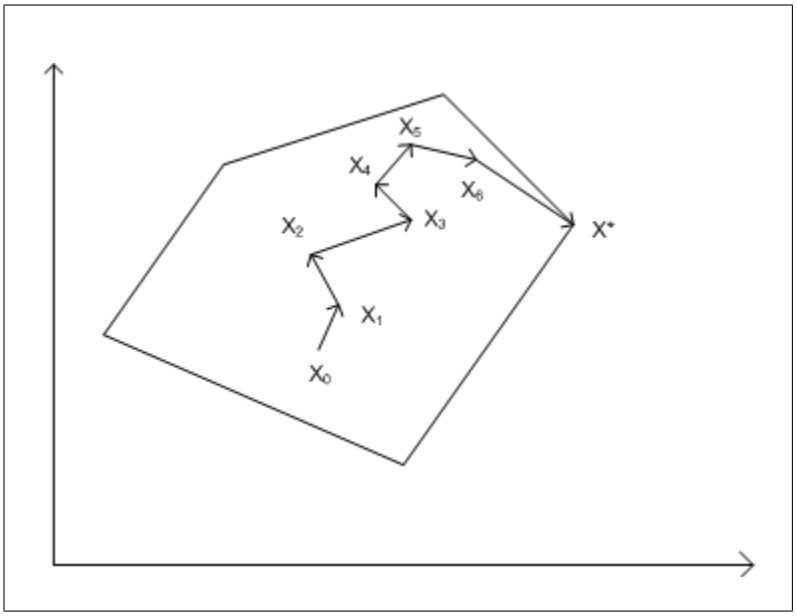

Figure 6.4. Expected convergence behavior of the affine scaling algorithm when applied on CMFWP

Table 6.9. Results of some trials for affine scaling algorithm

Trial 1									
	Initial point					Final solution			
w_{ij}	C1	C2	C3	C4	w_{ij}	C1	C2	C3	C4
F1	1	1	1	2	F1	0	0	0	5
F2	4	6	7	4	F2	5	7	8	1
Trial 2									
	Initial point					Final solution			
w_{ij}	C1	C2	C3	C4	w_{ij}	C1	C2	C3	C4
F1	2	1	1	1	F1	5	0	0	0
F2	3	6	7	5	F2	0	7	8	6
Trial 3									
	Initial point					Final solution			
w_{ij}	C1	C2	C3	C4	w_{ij}	C1	C2	C3	C4
F1	1.25	1.25	1.25	1.25	F1	5	0	0	0
F2	3.75	5.75	6.75	4.75	F2	0	7	8	6
Trial 4									
	Initial point					Final solution			
w_{ij}	C1	C2	C3	C4	w_{ij}	C1	C2	C3	C4
F1	1	3	0.5	0.5	F1	0	5	0	0
F2	4	4	7.5	5.5	F2	5	2	8	6
Trial 5									
	Initial point					Final solution			
w_{ij}	C1	C2	C3	C4	w_{ij}	C1	C2	C3	C4
F1	1	0.5	3	0.5	F1	0	0	5	0
F2	4	6.5	5	5.5	F2	5	7	3	6

final solutions found have good quality on the average. After we generate initial feasible interior points randomly, we can apply the algorithm and find the average deviation of the objective values of the final solutions from the optimal solution. However, the generation of feasible interior points is not an easy task since we have to both satisfy supply and demand constraints and also make all w_{ij}'s positive. We could achieve this for an instance which has 2 facilities and 4 customers, but could not generate random initial points for a larger instance with 4 facilities and 8 customers. As a result of these observations, we conclude that it can be better to apply affine scaling algorithm on dual form of the problem since we do not have such a problem in the generation of initial point in the dual form. We know that the CMFWP does not have a closed form dual if $d(\mathbf{x}_i, \mathbf{a}_j)$'s are unknown. However, if we locate facilities somewhere in the plane, $d(\mathbf{x}_i, \mathbf{a}_j)$'s become known and the objective function becomes linear in terms of c_{ij}'s and we can write the dual of the problem. The following algorithm is based on this fact.

Algorithm 9 Dual affine scaling algorithm

1. Locate facilities randomly within the convex hull of the customer locations and calculate distance vector $\mathbf{d}$ corresponding to these locations. Then find new cost multipliers by $\mathbf{c}' = \mathbf{cd}$.
2. Pick an interior point $\mathbf{y}^{(0)}$ such that $\mathbf{y}^{(0)}A < \mathbf{c}$ and $\mathbf{s}^{(0)} > 0$ where $\mathbf{s}^{(0)} = \mathbf{c}' - A^T\mathbf{y}^{(0)}$. Set $k = 0$ and repeat steps 3-6 until $\mathbf{y}$ converges.
3. Calculate $\Delta\mathbf{y} = (A(S^{(k)})^{-2}A^T)^{-1}\mathbf{b}$ and $\Delta\mathbf{s} = -A^T\Delta\mathbf{y}$ where $S^{(k)} = diag\{s_1^{(k)}, ..., s_{m\times n}^{(k)}\}$.
4. Calculate $\mathbf{x}^{(k+1)} = -(S^{(k)})^{-2}\Delta\mathbf{s}$.
5. If $\mathbf{x}^{(k+1)} > 0$, then find new locations corresponding to the new allocations $\mathbf{x}^{(k+1)}$ and find new distance vector $\mathbf{d}'$. Calculate new cost multipliers by $\mathbf{c}'' = \mathbf{cd}'$. Otherwise, set $\mathbf{c}'' = \mathbf{c}'$.
6. Find a step size λ to update $\mathbf{y}^{(k)}$ by using $\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \lambda\Delta\mathbf{y}$. λ takes values in the interval $(0, \lambda_{max})$ where $\lambda_{max} = \text{minimum}\left\{\frac{c''j - c'_j + s_j^{(k)}}{-\Delta s_j} : \Delta s_j < 0\right\}$. Generally λ is updated as $\lambda = \alpha\lambda_{max}$ where $0 < \alpha < 1$, but in practice $0.95 < \alpha < 0.99$.

In the original formulation of the affine scaling algorithm, λ_{max} is calculated as $\lambda_{max} = \text{minimum}\left\{\frac{s_j^{(k)}}{-\Delta s_j} : \Delta s_j < 0\right\}$. However, in our case, cost multipliers change,

therefore the right-hand side of the constraints $\mathbf{y}A = \mathbf{c}$ also changes. If we find λ_{max} by using this rule, we can come up with an infeasible $\mathbf{y}$. Therefore we change the update rule of λ_{max} in step 6 by adding $c''_j - c'_j$ to the nominator. The most important problem of the dual method is the fact that $\mathbf{x}^{(k)}$'s may become infeasible and negative during iterations. Since these variables represent allocations and are used to find new locations at each step, it is meaningless to use them to calculate locations when they take negative values. Therefore, we put a check as $\mathbf{x}^{(k+1)} > 0$ in step 5 before finding locations and if the check is not satisfied, we continue to work with the previous locations until $\mathbf{x}^{(k+1)}$ become nonnegative.

When we implement this algorithm and apply on the instance given above, we observe so many $\mathbf{x}^{(k)}$'s with negative entries. As a result, the algorithm performs poorly since we cannot see the advantage of updating cost coefficients. Therefore, we conclude that affine scaling methods are not helping us to find a global minimum to CMFWP, but can be used as heuristic procedures when designed suitably; this definitely requires serious attention.

7. COMPUTATIONAL RESULTS

7.1. Lagrangean Heuristics for Discrete Location-Allocation Problem

In this section we first assess the performance of the new Lagrangean heuristics both in terms of the solution quality and computation time on twenty three DCMLAP instances. Eight of these twenty three instances are rectilinear-distance problems and taken from literature (Sherali *et al.*, 2002; Sherali *et al.*, 1994). The remaining fifteen instances are Euclidean-distance problems and created randomly. For each test problem, we give in Table 7.1 the number of facilities to be located (m), the number of customers (n) and the number of candidate locations (K) within the convex hull of the customer locations. The optimal solutions for the rectilinear distance problems and the first ten of the Euclidean distance problems are available, but the optimal solutions for the remaining five of the Euclidean distance problems are not available. The optimal solutions of these problems could not be found due to memory restrictions. The instances for which we could find the optimal solutions form the first group of instances, and the remaining ones form the second group. The optimal solutions for the first group of instances are found by using Cplex 9.0.2 with default options. The experiments are performed on a Pentium 4 3.2 GHz machine with 2 GB RAM working in Windows environment. We solve the transportation problem by using the minimum cost flow problem freeware obtained from The Operation Research Group in University of Pisa (2006). To find the convex hull of the customer locations we used Graham's scan (Graham, 1972).

Table 7.2 displays the percentage deviations from the optimal objective value for all the methods we proposed for DCMLAP. These deviations are calculated as $100 \times (Z^* - Z_{LR})/Z^*$ for lower bounds and $100 \times (Z_{UB} - Z^*)/Z^*$ for upper bounds where Z^* denotes the optimal objective value. They cannot be calculated for those instances whose optimal solution is not known. Therefore, we decided to calculate the deviations for those instances as $100 \times (Z_{UB} - Z_{LR})/Z_{LR}$ and we report them on Table 7.3. LRSO, LRSOSP and LRSO2P give the same lower bound values, therefore the

Table 7.1. Instances in the test set

Instance	m	n	K
8	4	8	22
9	5	15	71
15	5	10	29
16	4	10	33
23	5	8	30
26	5	12	63
29	5	15	116
30	5	20	190
101	5	20	94
102	8	20	57
103	10	25	79
104	5	30	114
105	6	25	222
106	7	30	78
107	6	20	185
108	8	15	81
109	6	30	137
110	8	25	126
121	10	50	1088
122	5	50	1251
123	5	75	1905
124	10	75	1957
125	10	100	3450

deviations for the lower bounds for these methods are shown on a single column in Table 7.2. Table 7.2 also includes the optimal objective values.

Table 7.2. Per cent deviations from the optimal values for Lagrangean heuristics (%)

			LRSO, LRSOSP, LRSO2P	LRSO	LRSOSP	LRSO2P	ABLH	ABLH
Instance	(m, n)	Optimal value	LB	UB	UB	UB	LB	UB
8	(4,8)	793.00	6.63	0.00	0.00	0.00	6.70	0.00
9	(5,15)	9619.00	0.71	0.31	0.31	0.16	0.72	0.31
15	(5,10)	3427.00	2.31	2.16	2.16	2.16	2.26	2.16
16	(4,10)	259.00	6.73	57.14	40.15	0.00	6.98	57.14
23	(5,8)	238.00	5.56	0.00	0.00	0.00	4.87	0.00
26	(5,12)	284.00	3.73	53.52	10.56	3.52	3.56	53.52
29	(5,15)	729.00	1.62	0.00	0.00	0.00	1.45	0.00
30	(5,20)	745.00	1.50	1.88	0.00	0.00	1.66	1.88
101	(5,20)	109.18	31.93	0.00	0.00	0.00	30.30	11.52
102	(8,20)	405.17	4.95	0.71	0.71	0.00	4.50	0.71
103	(10,25)	457.38	5.31	13.12	7.98	2.17	5.51	13.09
104	(5,30)	1228.88	1.52	1.76	1.76	0.00	1.38	1.76
105	(6,25)	1839.78	1.73	0.00	0.00	0.00	1.82	0.00
106	(7,30)	934.26	8.92	2.42	0.00	0.00	8.82	1.66
107	(6,20)	3175.92	3.21	0.39	0.39	0.00	3.26	0.39
108	(8,15)	1674.40	9.94	0.40	0.40	0.19	9.95	2.62
109	(6,30)	1727.73	4.69	1.53	1.53	0.00	4.81	1.53
110	(8,25)	379.45	8.03	8.19	8.19	6.74	8.38	8.07
Average deviation			6.06	7.97	4.12	0.83	5.94	8.69

Table 7.4 includes the CPU times in seconds, first column being the CPU times of optimal solutions found using CPLEX for the instances we could find the optimal solutions and the other columns being the CPU times of LRSO, LRSOSP, LRSO2P and ABLH, respectively. Table 7.5 displays the CPU times of the proposed methods for the second group of instances.

Table 7.3. Duality gap between lower and upper bounds for Lagrangean heuristics

Instance	(m, n)	LRSO	LRSOSP	LRSO2P	ABLH
121	(10,50)	2.28	1.86	1.86	2.27
122	(5,50)	2.76	2.69	2.62	2.77
123	(5,75)	2.88	2.79	2.79	2.88
124	(10,75)	6.67	5.66	1.74	6.68
125	(10,100)	1.56	1.29	0.38	1.54
Average duality gap (%)		3.23	2.86	1.88	3.23

Table 7.4. CPU times of Lagrangean heuristics for the instances with known optimal solutions

Instance	(m, n)	Cplex	LRSO	LRSOSP	LRSO2P	ABLH
8	(4,8)	0.23	0.69	0.67	1.48	0.89
9	(5,15)	3.34	2.92	2.88	4.55	4.59
15	(5,10)	0.36	1.16	1.14	2.09	1.50
16	(4,10)	1.59	1.02	1.00	2.08	1.31
23	(5,8)	0.59	0.89	0.88	1.58	1.28
26	(5,12)	11.02	2.27	2.25	4.03	3.59
29	(5,15)	66.83	4.50	4.44	7.25	8.66
30	(5,20)	927.11	9.27	9.03	13.94	19.63
101	(5,20)	20.09	5.00	4.92	7.25	7.75
102	(8,20)	5.72	4.92	4.84	8.33	7.70
103	(10,25)	96.39	9.98	9.98	17.50	17.66
104	(5,30)	12.56	8.03	8.02	10.98	12.06
105	(6,25)	840.09	15.61	15.36	21.06	36.13
106	(7,30)	116.83	8.38	8.38	14.03	12.17
107	(6,20)	215.89	10.63	10.42	15.11	24.25
108	(8,15)	51.27	5.11	5.05	8.55	10.33
109	(6,30)	625.28	11.83	11.78	17.17	20.05
110	(8,25)	518.42	12.03	11.86	19.09	23.66
Average CPU time		195.20	6.35	6.27	9.78	11.84

Table 7.5. CPU times of Lagrangean heuristics for the instances without optimal solutions

Instance	(m,n)	LRSO	LRSOSP	LRSO2P	ABLH
121	(10,50)	190.31	189.25	316.11	1433.77
122	(5,50)	130.63	129.48	203.59	550.58
123	(5,75)	333.00	330.52	545.85	1240.28
124	(10,75)	586.49	584.63	1042.54	4577.28
125	(10,100)	1391.69	1388.48	2436.69	13557.78
Average CPU time		526.42	524.47	908.96	4271.94

The worst CPU times are reported for ABLH, although it could not bring any improvement on the lower or upper bounds. The CPU time required is even worse for the large sized problems in Table 7.5. The CPU time required by LRSO2P is higher than LRSO and LRSOSP although it gives the best results on the upper bounds. However, LRSO2P is twenty times faster on the average than the exact solution procedure for the first group of instances. Therefore, we decided to use LRSO, LRSOSP and LRSO2P for the other heuristics we designed for CMFWP since ABLH seems inadequate in terms of both the solution quality and CPU time.

To summarize, we could achieve to eliminate most of the optimality gap within a reasonable CPU time for most of the problems by using the Lagrangean heuristics for DCMLAP. Therefore, these methods can be used in other heuristics to solve CMFWP by using an approximating DCMLAP.

7.2. Lagrangean Heuristics for Continuous Location-Allocation Problem

In this section we present the performance of the DACL heuristic and the cellular heuristics both in terms of the solution quality and computation time on twenty three test instances of CMFWP. Eighteen of these twenty three instances are Euclidean-distance problems (E1-E12, E15-E20) and taken from literature (Al-Loughani, 1997; Sherali and Tuncbilek, 1992). Two test instances for the squared Euclidean distance

case (SE9 and SE21) have been obtained from (Al-Loughani, 1997) and three test instances with l_p distance (Lp8, Lp9 and Lp15) have been taken from (Sherali *et al.*, 2002) with p values equal to 1.25, 1.50 and 1.75. The size of these test instances and their best known/optimal objective values (BK/Opt. value) are provided in Table 7.6.

Table 7.7 displays the percentage deviations from the best known/optimal objective value for the DACL heuristic. The first step of this heuristic is solved to optimality by using Cplex 9.0.2 solver as well as by the heuristics we proposed for DCMLAP (i.e.,LRSO, LSROSP and LRSO2P). Then the results found in first step are improved via two phase heuristic. The final results found after applying two phase heuristic is reported here. The percentage deviations are calculated as $100 \times (Z^* - Z)/Z^*$ where Z^* denotes the best known/optimal objective value. The last row shows the average percent deviation for those test instances.

Table 7.8 includes the CPU times in seconds, first column being the CPU times of the solutions found using Cplex 9.0.2 and the other columns being the CPU times of LRSO, LRSOSP and LRSO2P, respectively. Listed CPU times are the totals of the times spent for solving DCMLAP optimally and two phase runs.

When we compare the performances of these two approaches, we see that the solutions found using Cplex is better than the solutions found by the proposed methods on the average. But this difference is mostly due to the instances E19 and SE9 since the proposed methods perform worse for those instances. If we discard these instances, we see that Cplex and the proposed methods perform nearly the same. Besides that, Cplex could achieve to find the optimal solution for 23 out of 29 instances. This observation supports also the claim that it is highly probable for best facility locations to coincide with customer locations. The results on Table 7.8 says that the CPU time required by Cplex is higher than other methods on the average.

Table 7.9 displays the percentage deviations from the best known/optimal objective value for the cellular heuristic. For each instance, $T = 50$ initial two phase runs are carried out. The grids for the rectangular topology are set as 5 for both x

Table 7.6. Instances in the test set

Instance	m	n	BK/Opt.value
E1	2	2	0
E2	2	4	247.28
E3	2	4	214.34
E4	3	5	24
E5	3	5	73.96
E6	3	9	221.4
E7	3	9	871.62
E8	4	8	609.23
E9	5	15	8169.79
E10	5	20	12846.87
E11	5	20	1107.18
E12	5	30	23990.04
E15	5	10	2595.47
E16	6	10	7797.21
E17	7	10	6967.9
E18	8	10	1564.46
E19	9	10	3250.68
E20	10	10	7719
SE9	4	8	875.34
SE21	4	24	6805.43
Lp8, p=1.25	4	8	710.2
Lp8, p=1.5	4	8	661.9
Lp8, p=1.75	4	8	630.72
Lp9, p=1.25	5	15	8998.93
Lp9, p=1.5	5	15	8646.61
Lp9, p=1.75	5	15	8350.95
Lp15, p=1.25	5	10	3046.07
Lp15, p=1.5	5	10	2827.55
Lp15, p=1.75	5	10	2689.12

Table 7.7. Per cent deviations from the best known/optimal values for DACL heuristic

Problem	(m,n)	Cplex	LRSO	LRSOSP	LRSO2P
E1	(2,2)	0.00	0.00	0.00	0.00
E2	(2,4)	0.00	0.00	0.00	0.00
E3	(2,4)	0.01	0.01	0.01	0.01
E4	(3,5)	0.00	0.00	0.00	0.00
E5	(3,5)	0.00	0.00	0.00	0.00
E6	(3,9)	0.00	0.00	0.00	0.00
E7	(3,9)	0.00	0.00	0.00	0.00
E8	(4,8)	0.00	0.00	0.00	0.00
E9	(5,15)	0.00	0.56	0.56	0.56
E10	(5,20)	0.00	0.52	0.52	0.52
E11	(5,20)	0.00	0.00	0.00	0.00
E12	(5,30)	0.00	1.20	1.20	0.00
E15	(5,10)	0.00	2.32	2.32	2.32
E16	(6,10)	0.00	1.64	1.64	1.56
E17	(7,10)	0.10	2.51	2.51	0.77
E18	(8,10)	0.00	0.00	0.00	0.00
E19	(9,10)	0.00	32.22	32.22	20.32
E20	(10,10)	0.01	6.08	6.08	3.77
SE9	(4,8)	33.78	137.62	106.21	68.62
SE21	(4,24)	20.23	24.82	20.90	20.90
Lp8, p=1.25	(4,8)	0.00	0.00	0.00	0.00
Lp8, p=1.5	(4,8)	0.00	0.00	0.00	0.00
Lp8, p=1.75	(4,8)	0.00	0.00	0.00	0.00
Lp9, p=1.25	(5,15)	0.00	0.33	0.33	0.33
Lp9, p=1.5	(5,15)	0.00	0.02	0.02	0.02
Lp9, p=1.75	(5,15)	0.10	0.53	0.53	0.53
Lp15, p=1.25	(5,10)	0.00	2.20	2.20	2.20
Lp15, p=1.5	(5,10)	0.00	2.25	2.25	2.25
Lp15, p=1.75	(5,10)	0.00	2.29	2.29	2.29
Average gap		1.87	7.49	6.27	4.38

Table 7.8. CPU times for DACL heuristic

Problem	(m,n)	Cplex	LRSO	LRSOSP	LRSO2P
E1	(2,2)	0.20	0.02	0.02	0.03
E2	(2,4)	0.16	0.03	0.03	0.05
E3	(2,4)	0.22	0.03	0.03	0.06
E4	(3,5)	0.08	0.03	0.05	0.11
E5	(3,5)	0.23	0.03	0.05	0.08
E6	(3,9)	0.17	0.08	0.11	0.19
E7	(3,9)	0.08	0.06	0.09	0.14
E8	(4,8)	0.25	0.09	0.13	0.22
E9	(5,15)	0.41	0.19	0.25	0.48
E10	(5,20)	0.75	0.27	0.38	0.69
E11	(5,20)	1.50	0.27	0.38	0.69
E12	(5,30)	11.02	0.45	0.67	1.36
E15	(5,10)	0.33	0.13	0.17	0.33
E16	(6,10)	0.28	0.14	0.19	0.44
E17	(7,10)	0.41	0.17	0.23	0.52
E18	(8,10)	0.55	0.19	0.27	0.61
E19	(9,10)	0.64	0.20	0.30	0.59
E20	(10,10)	0.42	0.22	0.30	0.66
SE9	(4,8)	0.80	0.43	0.48	0.75
SE21	(4,24)	3.22	1.90	1.95	2.89
Lp8, p=1.25	(4,8)	0.09	0.13	0.14	0.33
Lp8, p=1.5	(4,8)	0.16	0.16	0.14	0.30
Lp8, p=1.75	(4,8)	0.12	0.15	0.13	0.29
Lp9, p=1.25	(5,15)	0.41	0.30	0.33	0.68
Lp9, p=1.5	(5,15)	0.36	0.37	0.39	0.66
Lp9, p=1.75	(5,15)	0.35	0.37	0.37	0.66
Lp15, p=1.25	(5,10)	0.24	0.19	0.23	0.41
Lp15, p=1.5	(5,10)	0.27	0.24	0.21	0.42
Lp15, p=1.75	(5,10)	0.31	0.24	0.25	0.41
Average CPU time		0.83	0.24	0.28	0.52

and y axis. Facility locations found by initial two phase runs are clustered into groups and these groups are represented by their centroids. The number of cluster centroids found at the end is reported on the third column of the table. These centroids and the customer locations are used in the formulation of DCMLAP and LRSO, LSROSP and LRSO2P are applied to solve DCMLAP. The relative percent deviations are calculated as $100 \times (Z^* - Z)/Z^*$ where Z^* denotes the best known/optimal objective value.

Table 7.10 shows the results obtained by the cellular heuristics of Aras *et al.* (2007) for the same test instances. We present it here in order to compare two different cellular heuristic approaches. The method proposed by Aras *et al.* (2007) is based on determining candidate locations by dividing the rectangle covering all locally optimal facility locations into cells according to a predefined solution, while our method is based on clustering those locally optimal locations into different classes by using a clustering algorithm. They divide the rectangle into 5 equal intervals, which makes a total of 25 cells and perform $T = 25$ initial two phase runs. Then, they solve the resulting problem optimally by using the candidate facility locations found by the cellular heuristic. The details of the heuristic and the results in Table 7.10 can be seen in Aras *et al.* (2007).

Table 7.11 includes the total CPU times in seconds required to find initial two phase locations, construct the cells and solve the cellular heuristic,where the fourth, fifth and sixth columns being the CPU times of LRSO, LRSOSP and LRSO2P, respectively.

The results say that the cellular heuristic improves the solutions found by DACL heuristic for the instances E19 ans SE9. However, that improvement causes an increase in the CPU times as can be seen from the Table 7.11. The average CPU time required to solve these instances by the cellular heuristic is three times higher than by the DACL heuristic. The percentage deviations over other instances are close for the DACL and the cellular heuristic.

Table 7.9. Per cent deviations from the best known/optimal values for the cellular heuristic

Problem	(m, n)	No. of Centroids	LRSO	LRSOSP	LRSO2P
E1	(2,2)	0	0.00	0.00	0.00
E2	(2,4)	2	0.00	0.00	0.00
E3	(2,4)	4	0.01	0.01	0.01
E4	(3,5)	2	0.00	0.00	0.00
E5	(3,5)	3	0.00	0.00	0.00
E6	(3,9)	4	0.00	0.00	0.00
E7	(3,9)	4	0.00	0.00	0.00
E8	(4,8)	4	0.00	0.00	0.00
E9	(5,15)	8	0.56	0.56	0.56
E10	(5,20)	10	0.52	0.52	0.52
E11	(5,20)	5	0.00	0.00	0.00
E12	(5,30)	17	0.00	0.00	0.00
E15	(5,10)	11	2.32	2.32	2.32
E16	(6,10)	10	1.64	1.64	1.56
E17	(7,10)	6	2.51	2.51	0.77
E18	(8,10)	11	0.00	0.00	0.00
E19	(9,10)	6	24.09	24.09	20.32
E20	(10,10)	8	6.08	6.08	3.77
SE9	(4,8)	5	126.09	103.71	68.62
SE21	(4,24)	13	25.17	20.90	20.49
Lp8, p=1.25	(4,8)	4	0.00	0.00	0.00
Lp8, p=1.5	(4,8)	4	0.00	0.00	0.00
Lp8, p=1.75	(4,8)	4	0.00	0.00	0.00
Lp9, p=1.25	(5,15)	13	0.33	0.33	0.33
Lp9, p=1.5	(5,15)	11	0.02	0.02	0.02
Lp9, p=1.75	(5,15)	11	0.53	0.53	0.53
Lp15, p=1.25	(5,10)	10	2.20	2.20	2.20
Lp15, p=1.5	(5,10)	10	2.25	2.25	2.25
Lp15, p=1.75	(5,10)	12	2.29	2.29	2.29
Average gap			6.78	5.86	4.36

Table 7.10. Per cent deviations from the best known/optimal values for the cellular heuristic of Aras, Altınel and Orbay

Problem	(m, n)	No. of Centroids	Cellular Heuristic
E1	(2,2)	2	0.00
E2	(2,4)	2	0.00
E3	(2,4)	4	0.00
E4	(3,5)	4	0.00
E5	(3,5)	4	0.00
E6	(3,9)	5	0.1
E7	(3,9)	4	0.00
E8	(4,8)	6	12.70
E9	(5,15)	12	1.60
E10	(5,20)	11	0.10
E11	(5,20)	8	0.50
E12	(5,30)	12	3.70
E15	(5,10)	9	20.10
E16	(6,10)	5	16.90
E17	(7,10)	7	12.70
E18	(8,10)	8	48.90
E19	(9,10)	6	24.80
E20	(10,10)	10	3.30
SE9	(4,8)	17	18.30
SE21	(4,24)	21	3.40
Lp8, p=1.25	(4,8)	6	14.40
Lp8, p=1.5	(4,8)	5	14.90
Lp8, p=1.75	(4,8)	6	17.00
Lp9, p=1.25	(5,15)	12	1.80
Lp9, p=1.5	(5,15)	12	1.20
Lp9, p=1.75	(5,15)	10	1.60
Lp15, p=1.25	(5,10)	7	17.50
Lp15, p=1.5	(5,10)	6	13.70
Lp15, p=1.75	(5,10)	5	28.40
Average gap			9.57

Table 7.11. CPU times for the cellular heuristic

Problem	(m,n)	No. of Centroids	LRSO	LRSOSP	LRSO2P
E1	(2,2)	0	0.03	0.03	0.03
E2	(2,4)	2	0.08	0.08	0.09
E3	(2,4)	4	0.10	0.09	0.13
E4	(3,5)	2	0.07	0.07	0.10
E5	(3,5)	3	0.10	0.10	0.11
E6	(3,9)	4	0.24	0.23	0.28
E7	(3,9)	4	0.20	0.21	0.21
E8	(4,8)	4	0.59	0.57	0.65
E9	(5,15)	8	0.98	0.96	1.07
E10	(5,20)	10	1.35	1.89	2.23
E11	(5,20)	5	0.94	0.95	1.08
E12	(5,30)	17	1.11	1.11	1.47
E15	(5,10)	11	0.50	0.50	0.61
E16	(6,10)	10	1.93	1.93	2.06
E17	(7,10)	6	0.86	0.85	1.01
E18	(8,10)	11	1.71	1.80	2.17
E19	(9,10)	6	0.99	0.96	1.20
E20	(10,10)	8	1.06	1.07	1.20
SE9	(4,8)	5	0.19	0.19	0.24
SE21	(4,24)	13	0.46	0.52	0.76
Lp8, p=1.25	(4,8)	4	0.54	0.55	0.62
Lp8, p=1.5	(4,8)	4	0.58	0.60	0.68
Lp8, p=1.75	(4,8)	4	0.64	0.66	0.72
Lp9, p=1.25	(5,15)	13	0.87	0.93	1.03
Lp9, p=1.5	(5,15)	11	1.02	1.06	1.17
Lp9, p=1.75	(5,15)	11	1.15	1.20	1.31
Lp15, p=1.25	(5,10)	10	0.71	0.73	0.85
Lp15, p=1.5	(5,10)	10	0.55	0.58	0.70
Lp15, p=1.75	(5,10)	12	0.69	0.70	0.85
Average CPU time			0.70	0.73	0.85

8. CONCLUSIONS

In this thesis we have considered the capacitated multi-facility location-allocation problem with the Euclidean, squared Euclidean, rectilinear and l_p distance. We have studied both the discrete and continuous versions of this problem. The continuous location-allocation problem has a nonconvex objective function and is very difficult and sometimes impossible to solve with a commercial solver. Therefore, using a discrete approximation for continuous problems becomes essential to find at least an approximate solution. First, we have proposed new heuristic methods for the discrete location-allocation problem and based on these methods, we designed new heuristics for the continuous location-allocation problem.

Four heuristic methods have been proposed for the discrete location-allocation problem. Their accuracy and efficiency are evaluated on 23 test instances. All these heuristics are based on Lagrangean relaxation. The percentage deviations from the lower bounds for all the methods are nearly the same. When we observe the upper bounds, it is seen that the best upper bounds are given by the method where we apply two phase method at every iteration. The computational requirement of this method is larger than the method at which we apply Lagrangean relaxation with subgradient optimization (LRSO) and just a single two phase on LRSO at the end. However, this method is twenty times faster on the average than the exact solution procedure for the first group of instances. The worst CPU times are reported for the adaptation of Beasley's Lagrangean heuristic (ABLH), although it could not bring any improvement on the lower or upper bounds. We decided to discard this heuristic from the other methods we proposed for the continuous location-allocation problem since ABLH seems inadequate in terms of both the solution quality and CPU time.

We have proposed two heuristic methods for the continuous location-allocation problem, and tested them on 23 test instances available in the literature. Discrete approximation using customer locations (DACL) and cellular heuristics perform nearly the same except for a few instances. The cellular heuristic was able to improve the

solutions found by the DACL heuristic for those instances. The cost of such an improvement was additional CPU time required to solve cellular heuristics.

It is clear that the effective solution of the discrete version of the capacitated multi-facility location-allocation problem (DCMLAP) is very important for all heuristics proposed in our study. Therefore, the development of more efficient and accurate heuristics for DCMLAP improves the performance of the proposed heuristics. One more issue is the selection of promising candidate locations for facilities. Different strategies can be used to detect promising candidate locations. For example, in the cellular heuristic, we do not differentiate between the facilities when we are clustering the facility locations. An alternative approach could be to form the clusters for each facility separately.

We could not make use of the dual formulations of the continuous version of capacitated multi-facility location-allocation problem (CMFWP) to find an exact solution procedure, since the dual can be formulated only for a given allocation vector. These allocations form the right-hand side of the constraints in the dual formulation. Therefore, some parametric analysis, if possible, can be employed after the dual problem is solved for a given set of allocations (corresponding to a spanning tree on the graph induced by the problem). Then, we can move to a better spanning tree solution based on the results of parametric analysis. This procedure performs like an implicit enumeration method, however we can reach the optimal extreme point in a more efficient way due to the usage of parametric analysis.

Although we could not find an exact solution method by using affine scaling algorithm, a heuristic procedure can be designed on primal form of CMFWP. The problem here is to find random initial interior points, because it is quite difficult to find a feasible interior point satisfying both supply and demand constraints and also being positive. If a procedure can be designed to find initial interior points randomly, we can judge the effectiveness of such a heuristic method employing affine scaling.

Problem structure of CMFWP is not suitable for applying network algorithms where the total cost is additive in the flow in each arc, such as send-and-split method. Objective function of CMFWP is not additive in the flow in each arc due to the existence of distance terms. However, this structure allows for the application of extreme point enumeration or vertex ranking methods since it is possible to find an objective function value corresponding to extreme points. Therefore, more attention can be paid to these methods in the future to solve CMFWP efficiently.

REFERENCES

AL-Loughani, L., 1997. "Algorithmic approaches for solving the Euclidean distance location-allocation problems", *Ph. D. dissertation*, Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburgh, Virginia.

Aras, N., M. Orbay and İ. K. Altınel, 2006. "Efficient heuristics for the rectilinear distance capacitated multi-facility Weber problem", *Journal of the Operational Research Society*, advance online publication, October 18, 2006.

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.

Balas, E. and E. Zemel, 1980. "An algorithm for large zero-one knapsack problems", *Operations Research*, 28, 1130-1153.

Beasley, J. E., 1988. "An algorithm for solving large capacitated warehouse location problems", *European Journal of Operational Research*, 33, 314-325.

Beasley, J. E., 1993a. "Lagrangean heuristics for location problems", *European Journal of Operational Research*, 65, 383-399.

Beasley, J. E., 1993b. "Lagrangean relaxation", *Modern heuristic techniques for combinatorial problems (C.R.Reeves, ed)*, 243-303, Blackwell Scientific Publications.

Bolker, E. D., 1972. "Transportation polytopes", *Journal of Combinatorial Theory (B)*, 13, 251-262.

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

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

94-108.

Dikin, I. I., 1967. "Iterative solution of problems of linear and quadratic programming", *Soviet Mathematics Doklady*, 8, 674-675.

Dowling, P. D. and R. F. Love, 1987. "An evaluation of the dual as a lower bound in facilities location problems", *IIE Transactions*, 19, 160-166.

Drezner, Z., 1984. "The planar two-center and two-median problems", *Transportation Science*, 18, 351-361.

Erickson, R. E., C. L. Monma and A. F. Veinott, Jr., 1987. "Send-and-split method for minimum-concave-cost network flow", *Mathematics of Operations Research*, 12, 634-664.

Falk, J. E. and K. L. Hoffman, 1986. "Concave minimization via collapsing polytopes", *Operations Research*, 34, 919-929.

Fisher, M., 1981. "The lagrangean relaxation method for solving integer programming problems", *Management Sciences*, 27, 1-18.

Florian, M. and P. Robillard, 1971. "An Implicit Enumeration Algorithm for the Concave Cost Network Flow Problem", *Management Science*, 18, 184-193.

Gallo, G., C. Sandi and C. Sodini, 1980. "An algorithm for the min concave cost flow problem", *European Journal of Operations Research 4*, 248-255.

Gamal, M. D. H. and S. Salhi, 2003. "A cellular heuristic for the multisource Weber problem", *Computers and Operations Research*, 30, 1609-1624.

Geoffrion, A. M., 1974. "Lagrangean relaxation for integer programming", *Mathematical Programming Study*, 2, 82-114.

Graham, R. L., 1972. "An efficient algorithm for determining the convex hull of a finite

planar set", *Information Processing Letters*, 7, 175-180.

Guisewite, G. M. and P. M. Pardalos, 1991. "Global search algorithms for minimum concave-cost network flow problems", *Journal of Global Optimization*, 1, 309-330.

Hansen, P., J. Perreur and F. Thisse, 1972. "Location theory, dominance and convexity: Some further results", *Operations Research*, 28, 1241-1250.

Hansen, P., N. Mladenović and *É*. Tiallard, 1998. "Heuristic solution of the multisource Weber problem as a p-median problem", *Operations Research Letters*, 22, 55-62.

Hillier, F. S. and G. J. Lieberman, 1967. *Introduction to Operations Research*, 1st ed. Holden-Day, San-Francisco.

Hoare, C. A. R., 1962. "Quicksort", *Computer Journal*, 5, 10-15.

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

Jensen, P. A., 1980. *Network Flow Programming*, John Wiley&Sons Inc.

Kohonen, T., 1990. "The Self-Organizing Map", *Proceedings of the IEEE*, 78, 1464-1480.

Love, R. F., 1974. "The Dual of a Hyperbolic Approximation to the Generalized Constrained Multi-Facility Location Problem with l_p Distance", *Management Science*, 21, 22-33.

Love, R. F. and W. Y. Yeong, 1981. "A stopping rule for facilities location algorithms", *AIIE Transactions*, 13, 357-362.

Love, R. F. and H. Juel, 1982. "Properties and solution methods for large location-allocation problems", *Operations Research Society*, 33, 443-452.

Love, R. F. and P. D. Dowling, 1989. "A generalized bounding method for multifacility location models", *Operations Research*, 37, 653-657.

University of Pisa, OR Group, *http://www.di.unipi.it/di/ groups/optimize* (Last access in March 2006).

Portugal L., F. Bastos, J. Judice, J. Paixao and T. Terlaky, 1993. "An investigation of interior point algorithms for the linear transportation problems", *Technical Report No. 93-100*, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands.

Resende, M. G. C and G. Veiga, 1993. "An implementation of the dual affine scaling algorithm for minimum cost flow on bipartite uncapacitated networks", *SIAM Journal on Optimization*, 516-537.

Selim, S., 1979. "Biconvex programming and deterministic and stochastic location allocation problems", *Ph. D. dissertation*, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia.

Sherali, H. D. and F. L. Nordai, 1988. "NP-hard, capacitated, balanced p-median problems on a chain graph with a continuum of link demands", *Mathematics of Operations Research*, 13, 32-49.

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

Sherali, H. D., S. Ramachandran and S. Kim, 1994. "A localization and reformulation discrete programming approach for the rectilinear distance location-allocation problem", *Discrete Appl. Math*, 49, 357-378.

Sherali H. D., I. Al-Loughani and S. Subramanian, 2002. "Global optimization procedures for the capacitated Euclidean and lp distance multi-facility location-allocation problems", *Operations Research*, 50, 433-448.

The University of Tokyo, Institute of Medical Science, Human Genome Center, *http://bonsai.ims.u-tokyo.ac.jp/mdehoon/software/cluster/* (Last access at August 2006).

Wagner, H. M., 1959. "On a class of capacitated transportation problems", *Management Science*, 5, 304-318.

Weiszfeld, E., 1937. "Sur le point lequel la somme des distances de n points donn est minimum", *Thoku Mathematics Journal*, 43, 355-386.

Wendell, R. E. and A. P. Hurter, 1973. "Location theory, dominance and convexity", *Operations Research*, 21, 314-320.

Wendell, R. E. and E. L. Peterson, 1984. "A dual approach for obtaining lower bounds to the Weber problem", *Journal of Regional Science*, 24, 219-228.

Williams, J. W. J., 1964. "Algorithm 232-Heapsort", *Communications of the ACM*, 7, 347-348.

Zangwill, W. I., 1968. "Minimum concave cost flows in certain networks", *Management Science*, 14, 429-450.