SOLVING THE RECTILINEAR DISTANCE LOCATION-ALLOCATION PROBLEM USING LAGRANGEAN RELAXATION AND SUBGRADIENT OPTIMIZATION

by

Hasan Akyer

B.S., I.E., 9 Eylül University, 1998

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

2006

ACKNOWLEDGEMENTS

Many people helped me through this academic process. Among these, my sincerest regards go to my advisor Prof. İ. Kuban Altınel. His guidance, encouragement, and patience were invaluable to my successful completion of this degree. He provided me with opportunities to grow and learn both intellectually and personally.

I am grateful to other members of my committee, Assoc. Prof. Necati Aras and Assist. Prof. Orhan Feyzioğlu for their interest, comments, encouragement, and willingness to serve as my committee members. I would also like to thank all of the faculty and staff of the Department of the Industrial Engineering.

This thesis would not have been possible without the support of my friends. Among them special thanks to Z. Gamze Öz, Erinç Albey, and Engin Durmaz.

Finally, I owe a big gratitude to my parents, my sisters, and my brother for their support. I thank all of them for always believing in me. I dedicate this thesis to the memory of my grandmother, Elif Akyer.

In the end, I will like to mention the dear beautiful Boğaziçi University. I am very happy to be a member of it.

This research has been partly supported by Boğaziçi Research Grant No. 04HA301D.

ABSTRACT

SOLVING THE RECTILINEAR DISTANCE LOCATION-ALLOCATION PROBLEM USING LAGRANGEAN RELAXATION AND SUBGRADIENT OPTIMIZATION

This thesis is concerned with the rectilinear distance location-allocation problem, which seeks the location of capacitated facilities along with the allocation of their products to customers, so as to minimize total cost proportional to the rectilinear distance and the amount shipped. Knowing each customer's coordinates, unit cost from each facility to that customer, the customer's demand, and the supply at each facility, we determine each facility's optimal location and allocation simultaneously.

This is a non-convex optimization problem and difficult to solve exactly. However, there has been research concerning with the application on the development of accurate and efficient heuristic methods. In this work we continue this line of research and propose new Lagrangean relaxation based heuristics by using the new semi-Lagrangean relaxation approach.

The subproblems solved in the semi-Lagrangean relaxation are almost as intractable as the original one although the solution quality is very high. We therefore propose a subgradient algorithm to solve them approximately in order to increase efficiency. This new approach is implemented and computational results based on extensive experiments are also provided.

ÖZET

SINIRLI SIĞALI DİK-YATAY UZAKLIKLI YER SEÇİMİ-TAŞIMA PROBLEMİNİN LAGRANGE GEVŞETMESİ VE ALTGRADYAN ENİYİLEMESİ İLE ÇÖZÜMÜ

Biz bu çalışmamızda sonlu tesis sığaları ve müşteri istem kısıtlarını dikkate alarak kısıtları sağlayan ve toplam ulaştırma maliyetlerini enküçükleyen tesis yerlerinin belirlenmesi ve kaynak taşıma planının ne olması gerektiğini bulmaya çalışacağız. Müşteri yerleri, birim taşıma maliyetleri, müşteri istemleri ve tesis sığaları bilindiğinde, amacımız aynı anda en iyi tesis yerlerinin seçilmesi ve en iyi kaynak dağıtım planının belirlenmesidir.

Bu tür problemlerin en iyiye çözümünü bulmak zordur. Sezgisel yöntemler kullanılarak etkin ve doğru sonuçlar elde edilmeye çalışılmaktadır. Biz de bu yapılan araştırmalar doğrultusunda Lagrange gevşetme yöntemi tabanlı, yarı-Lagrange gevşetmesi üzerinde çalışacağız.

Yarı-Lagrange gevşetmesi ile test problemlerini çözdüğümüzde, iyi sonuçlar elde etmemize rağmen yeni elde edilen problemlerin orijinali kadar zor olduğu görülmüştür. Etkin bir çözüme ulaşabilmek için altgradyan yaklaşımı önerildi. Sonuç olarak yarı-Lagrange gevşetmesi ile altgradyan yöntemi problemlerin çözümü için birlikte uygulandı.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

K	Number of candidate points
m	Number of facility
n	Number of customer
N	Counter to keep the number of consecutive non-improvement for the subgradient optimization
t	Iteration number for the subgradient optimization
α	Stopping criteria for the subgradient optimization
β_j^c	Current lagrange multiplier β_j
β_j^p	Previous lagrange multiplier β_j
δ	Positive perturbation
λ_j^c	Current lagrange multiplier λ_j
λ_j^p	Previous lagrange multiplier λ_j
v	Step size for the subgradient optimization
π	Constant parameter for the subgradient optimization
φ	Optimality tolerance
Dev	Deviation
LAP	Location-allocation problem
LB	Lower bound
LDP	Lagrangean dual problem
$LLBP$	Lagrangean lower bounding problem
LP	Linear programming
LR	Lagrangean relaxation
NA	Not applicable
Opt	Optimum
P	General integer linear programing
$RLAP$	Rectilinear distance location-allocation problem
RLT	Reformulation-linearization technique

$RRLAP$	Relaxed rectilinear distance location-allocation problem
SLR	Semi-Lagrangean relaxation
$SLBP$	Semi-Lagrangean lower bounding problem
$SLDP$	Semi-Lagrangean dual problem
TET	Total execution time
UB	Upper bound

1. INTRODUCTION

The problem of facility location and allocation is encountered in many forms in various situations. These can range from retail goods supply centers to placement of critical infrastructure in a factory or operations center. Optimal placement of supplier facilities relative to the customers and their respective demands can play a crucial role in the viability, profitability, and efficiency of a supply network. In real-world cases, this optimal placement is constrained by unit costs of the goods or services being supplied and the distances involved in delivering them to the customers.

The generalization of this type of problem was first formalized by Cooper [8] as: "Given the location of each destination, its associated demand and a set of shipping costs for the region of interest, determine the number, location, and capacities of sources so as to minimize the total location and shipping cost". However, as mentioned above, real-world instances of this problem give rise to a variety of problem classes. For example, if the amount of goods each facility will supply to each customer is fixed, this problem reduces to a pure location problem. Conversely, if the locations of each facility is known and fixed beforehand, the problem becomes one of allocation. If neither the location of each facility nor the allocation is known, we have a location-allocation problem in which both an optimal location and allocation need to be determined concurrently.

Given the problem description above, we may be faced with further extensions to problem setup. The first of these is concerned with the number of facilities that need to be placed, single or many. In the simplest case, only one facility is to be located, this kind of problem is referred to as single facility problem. In general case, models involve the simultaneous location of many facilities and are referred to as multi facility problem. Second, location-allocation problems admit facilities that supply an infinite demand, whereas other applications look for the best placement of facilities with limited supply capacity. In this respect, problems are classified as capacitated or uncapacitated. Finally, each customer and allowable facility location can lie either

on a 2-dimensional grid (discrete) or anywhere on a 2-dimensional plane (continuous). Each of the above three characteristics will determine the specific class of the problem to solve. The specific class of problem we deal in this research is the capacitated multi facility location-allocation problem. Although it is originally defined as a continuous optimization problem, there is an equivalent binary integer programing formulation.

An essential part in the formulation of location-allocation problems is to identify the distance measure. We consider the case of planar continuous spaces and denote by (x_{i1}, x_{i2}) and (x_{j1}, x_{j2}) the coordinates of two points x_i and x_j, identifying the position of the facility and the customer location for which we want to measure the distance. General form of the distance measurement is the l_p distance:

$$l_p(x_i, x_j) = (|x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p)^{1/p} \quad 0 \leq p \leq \infty \tag{1.1}$$

Using the l_p distance function to model actual distances results in more accurate distance measures than restricting use to either of the special cases $p = 1$ (rectilinear distance) or $p = 2$ (euclidean distance). Rectilinear distance is one of the most familiar and widely used distance measures. It is variously referred to as rectangular, metropolitan or Manhattan distance in the literature. Rectilinear distance derive from the rectangular norm and can be mathematically expressed as:

$$l_1(x_i, x_j) = (|x_{i1} - x_{j1}| + |x_{i2} - x_{j2}|) \tag{1.2}$$

Rectilinear distance combines the feature of being very simple to treat analytically because of their linearity properties, and the feature of being a very appropriate distance measure for a large number of location-allocation problems. In particular, they are adequate in some urban location analysis where travel occurs along an orthogonal set of streets. The other widely used distance measure is the Euclidean or straight-line measure, denoted by $l_2(x_i, x_j)$. It is derived from the Euclidean norm and can be mathematically written as:

$$l_2(x_i, x_j) = \left(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2\right)^{1/2} \tag{1.3}$$

Euclidean distance applies when movement is allowed homogeneously in all directions. Another type of Euclidean measure is squared-Euclidean distance denoted as:

$$l_3(x_i, x_j) = \left(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2\right) \tag{1.4}$$

Specifically, we are interested in solving a particular type of location-allocation problem, the rectilinear distance multi facility location-allocation problem (RLAP). We know the cost of supplying a unit of product to a particular customer from a particular facility. Also, we know the amount of demand of each customer and the supply contained within each facility. Last, we know the location of each customer as interpreted as being on a discrete 2-dimensional coordinate grid. Our objective is to determine the optimal locations at which facilities should be placed in order to satisfy the customer demand with minimum total cost – which is proportional to the rectilinear distance from a facility to a customer and the amount of product shipped. As pointed out by Francis [13] this type of distance measure is appropriate when the transportation is occurring along a grid of city street (for which reason the measure is also known as a Manhattan norm), or along the aisles of a floor shop in the context of locating interacting machines or taking facilities to serve other existing machines in the workshop. We will give detailed information and formulation of RLAP in the next Chapter 5. The following assumptions are considered for this problem:

- The Rectilinear metric is used to measure the distance between facilities and customers.
- More than one facility can be located at the same point.
- No interaction exists among the suppliers and also no interaction exists among customers.
- Any point in the plane is permissible as a location for any source (there is no forbidden area).
- The number of new facilities is fixed, and subsequently the fixed cost associated with the installation of a source is ignored.
- The cost function is represented by the sum of weighted distances between supplier to customer.

In the next chapter, we present a brief literature review in two parts. In the first part, we discuss the location-allocation problem in general. We investigate what kinds of algorithms are used and how they are applied to this class of problems. In the second part, we focus on Lagrangean relaxation methodology and its applications. In brief, we present how these techniques can be applied and investigated for the method of solution to different types of problems.

2. LOCATION-ALLOCATION PROBLEM

The uncapacitated version of the location-allocation problem was first considered by Cooper [7]. He proposed a total enumeration algorithm to obtain an optimal solution. Each customer is served by a single facility because there is no restriction on the capacity of the facility. Given the assignment of the destination to each customer, the algorithm finds the optimum locations using the first order optimality conditions.

Cooper [8] also suggested some heuristic solution procedures which are based on the idea of alternately solving the location and allocation problems. The first algorithm restricts the source location to a subset of the destination sites. The second approach assigns a subset of the destinations to a single source, and then, for each new facility a single source location problem is solved using an exact location method.

Cooper [9] was the pioneering researcher on the capacitated version of location allocation problems in which the Euclidean distances are used to measure the distances between the facilities and customers. He proposed an exact, total enumeration algorithm and provided two heuristic methods. In the exact solution, enumeration of all basic feasible transportation flows is performed based on the fact that optimum solution lies at an extreme point of the feasible region, and for each of them, the optimum location is determined. The first heuristic procedure has the basic idea of alternatively solving the location and allocation problems. When started with a randomly selected initial location of the facilities, the problem reduces to the ordinary transportation problem. After solving this problem, optimum allocations are found. With the new set of allocations the problem becomes a pure location problem and the optimum location of facilities are determined. The above mentioned algorithm is called as the two-phase method and it continues until no further improvement can be obtained. The second approach uses a heuristic method that ignores the capacity constraints and solves then uncapacitated problem. When the capacity constraints are violated, the sources having available capacities are chosen to study the customers having unsatisfied demands. This allocation process is achieved in an appropriate way so that the resulting change

in the objective function is relatively small with the new set of allocations. Then the algorithm continues this alternating location-allocation procedure until no improvement in the objective function is possible.

Sherali et al. [26] have proposed a more effective algorithm for the rectilinear distance location-allocation problems, using a localization result; they proposed a new formulation of this problem as a mixed-integer bilinear programming problem. This technique applies Reformulation-Linearization Technique (RLT), proposed by Sherali and Tuncbilek [25] to provide tight lower bounds on the problem.

The RLT essentially generates a hierarchy of progressively tighter, higher dimensional, linear programming or convex relaxations for the underlying mixed-integer, polynomial 0-1 and continuous problems that can assist in solving the given problem within the context of a partitioning scheme. This is achieved within two phases. In the first phase, the Reformulation Phase additional nonlinear implied inequalities are generated and added to the problem. In the second phase, which is the Linearization Phase, the problem resulting from the first phase is transformed into a linear program by substituting a single variable for each distinct variable-product term. A cutting plane procedure is used to strengthen the developed linear programming relaxation and a Lagrangean dual based lower bound is then computed to obtain a quick lower bound on the linear programming problem. This lower bounding technique is embedded within a branch and bound algorithm that enumerates over the locational decision space. Using their algorithm, they were able to solve problems up to size ranging form 5 facilities to 20 customers to within 5% of optimality. Also they found that their algorithm is more efficient for problems in which the number of facilities is small compared to the number of customers, which makes it more suitable for real-world problems.

Sherali and Tuncbilek [25] studied a squared-Euclidean distance location-allocation problem with balanced transportation constraints. First, they fix the transportation flow and using calculus, they showed that this problem can be transformed into a quadratic convex maximization problem by projecting it onto the space of allocation variables. Based on the extreme point optimality property and the nature of the trans-

portation constraints, they propose a partitioning scheme using the dichotomy that a variable can be either zero or positive at optimality. They developed an efficient exact algorithm based on applying a specialized RLT approach to obtain an upper bounding linear program. Using the upper bounding schemes, a branch and bound algorithm was developed by portioning the problem according to the dictionary that a variable is either zero or positive fixed variables at an optimal (vertex) solution. By the help of this algorithm, they were able to handle problems with size up to 20 facilities and 120 customers.

Bongartz et al. [6] propose a new solution algorithm for location-allocation problem with the l_p norm, where $1 < p < \infty$. Their method based on relaxing the $\{0, 1\}$ constraints on the allocations and solves both the location and allocations simultaneously. Related to this kind of location-allocation problems Murtagh and Niwattinsyawong [21] present an approach relaxing the $\{0, 1\}$ constraints on the allocations. MINOS, a large-scaled nonlinear programming package, is then used to solve both the location and allocation problems simultaneously. When the method proceeds, it fixes any allocations, reaching either zero or one and then updates only the free variables. Borgartz et al. [6] algorithm finds a stationary point on a series of subspaces defined by the current set of activities. Once at a stationary point on the current subspace, the algorithm checks the necessary sufficient conditions for local minimal of the relaxed problem, and either accepts the point as a local minimum or determines a new subspace on which to perform the minimization.

3. LAGRANGEAN RELAXATION AND SUBGRADIENT OPTIMIZATION

Good lower and upper bounds on the optimal objective value are crucial in the efficient solution of integer programing problems. Upper bounds are in general obtained by means of heuristics. However, lower bounds are produced by using different relaxation strategies.

One of the well-known general techniques is linear programming (LP) relaxation. In LP relaxation, we relax the integrality requirement on the variables. Then this problem can be solved by using either a standard algorithm like simplex or a heuristic method like dual ascent. The solution gives a lower bound on the optimal solution of the original problem. In many cases however solving the LP relaxation is impracticable, because typically original problem involves a large number of variables and/or constraints. We therefore need alternative techniques for generating lower bounds [4].

Another well-known technique to find a lower bound is Lagrangean relaxation. Lagrangean relaxation is commonly used in combinatorial optimization to generate lower bounds for a minimization or maximization problem. One of the most popular examples of how Lagrangean relexation can be applied to integer programming is given by the pioneering work of Held and Karp [12] on the traveling salesman problem. Briefly, a Lagrangean relaxation corresponds to a problem in which the complicating constraints are put as a penalty term in the objective function. This penalty term is a function of the amount of violation of these constraints weighted by multipliers. The objective value of the relaxed problem gives a lower bound on the optimal solution of the original problem. There are two basic reasons why Lagrangean relaxation is used successfully.

i) Many combinatorial optimization problems consist of an easy problem (in the NP-complete sense, i.e. solvable by a polynomially bounded algorithm) complicated by the addition of extra constraints. This kind of constraints make the problem difficult

to solve. By absorbing these complicating constraints into the objective function, we are left with an easy problem to solve and attention can then be turned to choosing numerical values for the Lagrange multipliers.

ii) In terms of computational cost, practical experience with Lagrangean relaxation has indicated that it gives very good lower bounds. By using Lagrangean relaxation we often solve problems in a reasonable time compared to algorithms which guarantee to find an optimal solution.

Let us consider the following general integer linear programming (P):

$$\min\ z = c^T x \tag{3.1}$$

$$s.t.$$

$$Ax = b \tag{3.2}$$

$$Bx = d \tag{3.3}$$

$$x \in \{0,1\}^n \tag{3.4}$$

Choose (3.2) as the complicating constraints. Then, by introducing the Lagrange multiplier vector (λ) we can obtain the following Lagrangean subproblem called as Lagrangean lower bounding program (LLBP):

$$\min\ z_{LB}(\lambda) = \ c^T x + \lambda^T (b - Ax) \tag{3.5}$$

$$s.t.$$

$$Bx = d \tag{3.6}$$

$$x \in \{0,1\}^n \tag{3.7}$$

Notice $z_{LB}(\lambda) \leqslant z^*$ for any (λ) and the determination of multipliers which give the tightest lower bound becomes important.

This can be done by solving the Lagrangean dual problem (LDP):

$$max \begin{cases} \min & z_{LB}(\lambda) = c^T x + \lambda^T (b - Ax) \\ \text{s.t.} & \\ & Bx = d \\ & x \in \{0,1\}^n \end{cases}$$

Ideally, the optimal value of the Lagrangean relaxation program is equal to the optimal value of the original integer program. If the two programs do not give the same objective function value, then a duality gap is said to exist. Duality gap is measured by the relative deviation between z^* and z^*_{LB} denoting the optimum value of the original problem and optimum value of the Lagrangean dual problem respectively.

Choosing the set of constraints to relax and value of Lagrange multipliers are of key importance in terms of solution quality [4]. Before determining which constraints to be selected to relax, integrality property should be explained. If the solution to LLBP is integral then the Lagrangean relaxation / Lagrangean lower bound program is said to have the integrality property. If the Lagrangean relaxation have the integrality property, we can say that maximum lower bound is equal to the LP relaxation solution otherwise maximum lower bound is greater than the LP relaxation solution. As a result, a Lagrangean relaxation without the integrality property is to be preferred because it offers better lower bounds.

Related to applying Lagrangean relaxation to a problem we have to examine some issues. First one is that how good the lower bounds are. According to literature the lower bounds provided by Lagrangean relaxation are extremely sharp. Second issue is that how we select constraints to be relaxed. Two properties are important in evaluating a relaxation, the first one is the sharpness of the bounds produced and the other one is the amount of computation required to obtain these bounds. Usually selecting a relaxation involves a trade off between these two properties; sharper bounds require more time to compute. It is generally difficult to know whether a relaxation with sharper bounds but greater computation time will result in a branch and bound

with better overall performance. However, it is usually possible to at least compare the bounds and computation requirements for different relaxation strategies.

There are two basic approaches to solve LDP: subgradient optimization and multiplier adjustment. The subgradient optimization is an adaptation of the gradient method in which gradients are replaced by subgradients. The subgradient optimization is popular in determining Lagrange multipliers, because it is easy to program and has worked well on many practical problems. Multiplier adjustment is based on generating a starting set of Lagrange multipliers, improving them in some systematic way so as to generate an improved lower bound and repeating the procedure if an improvement is made. The main advantage of multiplier adjustment is that it is computationally cheap. The other advantage is that it produces an increase in the lower bound at each iteration. On the other hand, multiplier adjustment has two fundamental disadvantages. One is that the final lower bound can be poor. The other drawback of multiplier adjustment is that different problems require different multiplier adjustment algorithms, which makes it difficult to apply.

Subgradient optimization is an iterative procedure which updates the multiplier values in a systematic fashion. It can be viewed as a procedure which attempts to maximize the lower bound value obtained from LLBP by suitable choice of multipliers. Switching from matrix notation to summation notation, so that the relaxed Consraints (3.2) are $\sum_{i=1}^{m} a_{ij}x_j = b_i \;\; ; \;\; i = 1, 2, ..., m$. The basic subgradient optimization procedure is described in Algorithm 1.

Usually the initially Lagrange multipliers are set to zero. There are two possible ways to terminate this iterative procedure. The first one is limiting the number of iterations. The other one is reducing the value of π can during the course of procedure. Note that in Step 5, λ can be positive or negative because we relaxed equality constraints. It is possible to give some tactical notes related to applying subgradient optimization. It is clear that if lower bound and upper bound is equal, we can reach the optimal solution and terminate the subgradient optimization procedure. If we can

Algorithm 1 Subgradient Optimization

begin

1. Decide upon an initial set of multipliers. Let π be a user defined parameter satisfying $0 \leq \pi \leq 2$ and set iteration number $t = 0$. Initialize upper bound z_{UB} (e.g. from some heuristic for the problem).

2. Solve LLBP with the current set of multipliers in order to calculate the lower bound.

3. Define subgradients G_i for the relaxed constraints, evaulated at the current solution, by:

$$G_i = b_i - \sum_{j=1}^{n} a_{ij}x_j \quad ; i = 1, 2, ..., m$$

4. Define a scalar step size ν by:

$$\nu = \frac{\pi(z_{UB} - z_{LB})}{\sum_{i=1}^{m} G_i^2}$$

5. Update λ_i using $\lambda_i = \lambda_i + \nu G_i$, set $t \leftarrow t + 1$ and go to Step 2 to resolve LLBP with this new set of multipliers.

end;

not improve the objective value in the last N subgradient iterations with the current value of π, then π is halfed.

One of the main advantage of subgradient optimization is that it works together with Lagrangean relaxation and gives good quality lower bounds. The other advantage is that subgradient optimization works from a general description of the relaxed constraints, and as such is capable of being directly applied without alteration, whatever the exact nature of those constraints [4].

Shapiro [24] specifically considers the zero-one IP problem:

$$\min\ z = c^T x \tag{3.8}$$

$$s.t$$

$$Ax \leq b \tag{3.9}$$

$$Bx = d \tag{3.10}$$

$$x \in \{0,1\}^n \tag{3.11}$$

Constraint (3.9) relaxed and defined the Lagrangean function as

$$L^o\left(\bar{u}\right) = -\bar{u}b + Minimum\left\{\left(c^T + \bar{u}A\right)x\right\} and\ x \in \{0,1\}^n$$

where u denote denote a non-negative vector of Lagrange multipliers. The function $L^o\left(\bar{u}\right)$ is easily optimized by any zero-one solution where x satisfying

$$\bar{x}_j = \begin{cases} 0 & if\ c_j + \bar{u}a_j > 0 \\ 0\ or\ 1 & if\ c_j + \bar{u}a_j = 0 \\ 1 & if\ c_j + \bar{u}a_j < 0 \end{cases}$$

with c_j representing component j of c, and a_j column j of A. If the above three conditions are satisfied for some $\bar{u}$, then zero-one solution $\bar{x}$, which is optimal in the Lagrangean technique is also optimal in the given IP problem. If we can not find an optimal solution of the original IP problem by using Lagrangean relaxation, we say there is a duality gap. This concept is an important measure of solution quality. There are two different approaches to overcome a duality gap. One is to branch on a variable at a fractional level in the LP relaxation, namely use branch and bound. The other approach is to strengthen the dual problem by restricting the solutions permitted in the Lagrangean minimization to be a strict subset of the zero-one solutions. He studied these two approaches in detail and he also showed us how Lagrangean techniques can be applied into a manufacturing system consisting of I items for which production is to be scheduled at minimum cost over T time periods.

Fisher [14] has covered a review of a Lagrangean relaxation based on what has been done in the previous decade. Dualizing the side constraints produces a Lagrangean problem that is easy to solve and whose optimal value is a lower bound for minimization problems on the optimal value of the original problem. The Lagrangean problem can thus be used in place of a linear programming relaxation to provide bounds in a branch and bound algorithm. With the help of this method he improved his algorithm in order to solve some important problems in the areas of routing location, scheduling, set covering, and assignment. He applied Lagrangean relaxation to the generalized assignment problem.

Beasley [3] proposes a framework for developing Lagrangean heuristics based on Lagrangean relaxation and subgradient optimization with respect to location problems. He applied Lagrangean relaxation to p-median, uncapacitated warehouse location, capacitated warehouse location, and capacitated warehouse location with single source constraints. He relaxed both capacity constraints and customer demand constraints for capacitated warehouse location problem. As a result of his study he obtained good quality solutions for each of these different location problems. In order to develop Lagrangean heuristics for the four location problems, instead of formulating each problem separately, he considers them as special cases of the generalized warehouse location problem.

4. SEMI-LAGRANGEAN RELAXATION

In the previous Chapter 3 we have studied Lagrangean relaxation which is commonly used in combinatorial optimization in order to generate lower bounds for a minimization problem [16]. For a given problem, there may exist different Lagrangean relaxations. Beltran et al. [5] proposed a modified Lagrangean relaxation which is used in (linear) combinatorial optimization with equality constraints, and show that the new approach eventually generates an optimal integer solution. They called this new concept as the semi-Lagrangean relaxation (SLR).

The semi-Lagrangean relaxation has the theoretical property that its optimal value is the same as for the original problem. This relaxation thus closes the duality gap. Unfortunately, the associated subproblem are almost as intractable as the original problem itself although the solution quality is very high.

Let us consider the following general integer linear programing (P):

$$\min\ z = c^T x \tag{4.1}$$

$$s.t.$$

$$Ax = b \tag{4.2}$$

$$Bx = d \tag{4.3}$$

$$x \in \{0,1\}^n \tag{4.4}$$

The semi-Lagrangean relaxation consists in relaxing the equality constraints, but keeping in the meantime a weaker form of equality in the constraints set. Choose (4.2) and (4.3) as the complicating constraints. Then by introducing the Lagrange multiplier vectors (λ, β) we can obtain the following semi-Lagrangean subproblems, called semi-

Lagrangean lower bounding program (SLBP):

$$\min z_{SLB}(\lambda,\beta) = c^T x + \lambda^T(b - Ax) + \beta^T(d - Bx) \tag{4.5}$$

$$s.t.$$

$$Ax \leqslant b \tag{4.6}$$

$$Bx \leqslant d \tag{4.7}$$

$$x \in \{0,1\}^n \tag{4.8}$$

Notice $z_{SLB}(\lambda,\beta) \leqslant z^*$ for any (λ,β) and the determination of multipliers which give the tightest lower bound becomes important. This can be done by solving semi-Lagrangean dual problem (SLDP_1):

$$max \begin{cases} \min & z_{SLB1}(\lambda,\beta) = c^T x + \lambda^T(b - Ax) + \beta^T(d - Bx) \\ \text{s.t.} & \\ & x \in \{0,1\}^n \end{cases}$$

In order to strengthen semi-Lagrangean lower bound, we insert the first relaxed constraint in a weaker form into the constraints set. So semi-Lagrangean dual problem (SLDP_2) becomes:

$$max \begin{cases} \min & z_{SLB2}(\lambda,\beta) = c^T x + \lambda^T(b - Ax) + \beta^T(d - Bx) \\ \text{s.t.} & \\ & Ax \leqslant b \\ & x \in \{0,1\}^n \end{cases}$$

Lastly, we add the second relaxed constraint in a weaker form into the constraints set.

As a result, semi-Lagrangean dual problem (SLDP_3) becomes:

$$max \begin{cases} \min \quad z_{SLB3}(\lambda,\beta) = c^T x + \lambda^T(b - Ax) + \beta^T(d - Bx) \\ \text{s.t.} \\ \qquad\qquad Ax \leqslant b \\ \qquad\qquad Bx \leqslant d \\ \qquad\qquad x \in \{0,1\}^n \end{cases}$$

Algorithm 2 Semi-Lagrangean Relaxation Method

begin

1. Solve the semi-Lagrangean dual problem (SLDP_1).

$(\lambda^1, \beta^1) = \arg \underset{\lambda,\beta}{max} z_{SLB1}(\lambda,\beta)$.

Let x^1 be an optimal solution of $SLDP_1$ problem at (λ^1, β^1).

If x^1 is feasible to P, STOP: x^1 is an optimal solution to P.

2. Solve the semi-Lagrangean dual problem (SLDP_2) by using (λ^1, β^1) as starting vector.

$(\lambda^2, \beta^2) = \arg \underset{\lambda,\beta}{max} z_{SLB2}(\lambda,\beta)$.

Let x^2 be an optimal solution of $SLDP_2$ associated to (λ^2, β^2).

If x^2 is feasible to P, STOP: x^2 is an optimal solution to P.

3. Solve the semi-Lagrangean dual problem (SLDP_3) by using (λ^2, β^2) as starting vector.

$(\lambda^3, \beta^3) = \arg \underset{\lambda,\beta}{max} z_{SLB3}(\lambda,\beta)$.

Let x^3 be an optimal solution of $SLDP_3$ problem at (λ^3, β^3).

If x^3 is feasible (satisfies 4.2, 4.3, 4.4) to P, STOP: x^3 is an optimal solution to P.

4. Compute $z_{SLB3}(\lambda,\beta)$ with $\lambda^4 = \lambda^3 + \delta$, $\beta^4 = \beta^3 + \delta$, for a small and arbitrary positive perturbation.

Let x^4 be an optimal solution of $SLDP_3$ at (λ^4, β^4), STOP: x^4 is an optimal solution to P.

end;

In our computations we were able to reach an optimal solution either by Step 2 or by Step 3 without performing Step 4. At this point the main question is that how we can solve these subproblems $SLDP_1$, $SLDP_2$, and $SLDP_3$. In order to solve these problems we use subgradient optimization represented in Algorithm 1. Semi-Lagrangean relaxation have some important advantage. One of the most crucial is that for large enough Lagrange multipliers semi-Lagrangean relaxation has the same optimal solution as the original problem. So it closes the duality gap. Nevertheless, the subproblem may be as difficult as the original problem itself.

Beltran et al. [5] applied semi-Lagrangean relaxation to p-median problems. In the p-median problem the objective is to open p facilities from a set of m candidate facilities relative to a set of n customers and to assign each customer to a single facility. The p-median problem is an NP-hard problem. For this reason the p-median problem has been solved basically by heuristic methods.

In a general concept they relaxed the equality constraints that ensure that each customer is assigned to exactly one median. This yields a max-min optimization problem, in which the binding equality constraints are no longer present. They realized that the inner minimization problem is thus separable and easy. However, Lagrangean relaxation and linear relaxation give them the same optimal value. In order to strengthen the standard Lagrangean relaxation, they insert into the minimization problem a weaker form of the equality constraint, as a less than or equal inequality. This is the main idea of the semi-Lagrangean relaxation. So we can apply this method to any problem with equality constraints. However, the subproblem with large multipliers may be as difficult as the original problem itself. On the other hand, when the Lagrange multipliers are small, p-median problem and for some others like the set partitioning problem are very easy to solve.

5. SOLVING THE RECTILINEAR DISTANCE LOCATION-ALLOCATION PROBLEM BY LAGRANGEAN RELAXATION

The objective of the rectilinear distance location-allocation problem (RLAP) is to determine the locations (x_i, y_i) $i = 1, ..., m$ of m suppliers and corresponding allocations u_{ij} $i = 1, ..., m$; $j = 1, ..., n$. In order to find an optimal solution, our criterion is that total cost of flows among the facilities are minimum and the demand of customers are satisfied by the suppliers. We can formulate this problem as below.

RLAP:

$$\min\ z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}u_{ij}\,(\mid x_i - a_j \mid + \mid y_i - b_j \mid) \tag{5.1}$$

$s.t.$

$$\sum_{i=1}^{m} u_{ij} = d_j \qquad j = 1, ..., n \tag{5.2}$$

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

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

The indices and parameters are defined as follows:

m : the number of facilities which will be located.

n : the number of customers.

c_{ij} : the cost of transporting one unit of quantity for one unit of distance from i to j.

s_i : the capacity of supplier i.

d_j : the demand of customer j.

(x_i, y_i) : the coordinates of the supplier i.

(a_j, b_j) : the coordinates of customer j.

The decision variables used in the model are defined as follows:

u_{ij} : the quantity allocated from supplier i to customer j.

It is assumed that total supply equals total demand, namely the problem is balanced. However, if total supply is greater than total demand, a dummy customer can be added to obtain an equivalent balanced formulation. Observe that for a fixed set of locations (x_i, y_i) $i = 1, ..., m$ RLAP reduces to the ordinary transportation problem and for a fixed set of allocations RLAP reduces to m single facility, pure location problems with rectilinear distances, each of which is separable in the variables x_i and y_j for $i = 1, ..., m$; $j = 1, ..., n$.

According to Wendell and Hurter [30] the rectilinear distance facility location problem always has an optimal solution with facilities located at the intersection points of vertical and horizontal lines drawn through the customer locations. Hansen et al. [11] have shown that only those intersection points that lie in the convex hull of customer locations should be considered as candidate points for the optimal facility locations. So, we can reformulate RLAP as a discrete location-allocation problem. We define the flow variables u_{ikj} as the amount shipped from a facility i located at candidate point k to customer j. Variables z_{ik} are binary variables denoting whether facility i is located at point k. c_{ikj} is the unit shipment cost from facility i located at point k to customer j. It is obtained by multiplying the unit shipment cost per unit distance, c'_{ikj}, by the distance between point k and customer j:

$$c_{ikj} = c'_{ikj} \left(\mid x_k - a_j \mid + \mid y_k - b_j \mid \right).$$

Then we can formulate the rectilinear distance location-allocation problem (RLAP) as:

$$\min\ z = \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{j=1}^{n} c_{ikj}u_{ikj} \tag{5.5}$$

s.t.

$$\sum_{i=1}^{m}\sum_{k=1}^{K} u_{ikj} = d_j \qquad j = 1, ..., n \tag{5.6}$$

$$\sum_{k=1}^{K} z_{ik} = 1 \qquad i = 1, ..., m \tag{5.7}$$

$$\sum_{j=1}^{n} u_{ikj} = s_i z_{ik} \qquad i = 1, ..., m;\ k = 1, ..., K \tag{5.8}$$

$$u_{ikj} \geq 0 \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n \tag{5.9}$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K \tag{5.10}$$

Constraints (5.6) guarantee that the demand of each customer is satisfied. Constraints (5.8) ensure that the total amount shipped from a point cannot exceed the capacities of the facilities and also that if there is no facility at point k, no shipment can be made. Constraints (5.7) ensure that each facility be located at some point. Also, we can say that more than one facility can be located at the same point.

RLAP is a mixed-integer linear programming model so there is no need for any transformation like the RLT [25]. So we can solve this problem optimally by using standard integer programming solvers. There are also numerous solvers available in the market, however, their computation time differs from each other. Besides, the required computation time may increase exponentially with the size of the problem. When we analyze this problem with n customers and m facilities, the maximum number of intersection points within the convex hull is n^2, which result in mn^2 binary variables z_{ik} and mn^3 variables u_{ikj}. The number of candidate points in the convex hull determine the problem size.

5.1. The Two-Phase Method

Cooper [9] developed a two-phase method that has the basic idea of alternatively solving the location and the allocations problems with the Euclidean distance function. The procedure takes some of the candidate locations as initial locations for the facilities. The resulting transportation problem is then solved to find optimum allocations with this known locations. Using the allocations, the new optimal locations are determined. This process continues until further improvement is not possible. According to this idea, we can say that RLAP is a combination of two simple subproblems. If we know the optimum allocations for RLAP, it turns into the multi facility rectilinear distance pure location problem. On the other hand, if we know the optimum locations, RLAP turns into the ordinary transportation problem.

Algorithm 3 The Two-Phase Method for RLAP

begin

1. Locate the facilities at randomly generated candidate points (x_k, y_k) $k = 1, ..., K$.

2. Calculate the distance between each customer to facility and define the cost function: $c_{ikj} = c_{ij} \left(\mid x_k - a_j \mid + \mid y_k - b_j \mid \right)$.

3. Using the initial locations, solve the transportation problem to find a set of allocations u_{ikj}.

4. Solve the m single facility problems using the current allocations u_{ikj}.

5. Repeat Steps 2 - 4 until either the facility locations (x_k, y_k) or the allocations u_{ikj} remain unchanged.

6. Report final facility locations and allocations.

end;

According to this explanation we develop an algorithm to solve RLAP approximately. We locate the facilities at some random initial points. Then RLAP turns into a transportation problem. Later, we solve this problem and find the allocations for each of the facility and corresponding customers. In the next step, using these allocations, RLAP is solved and using the location results of RLAP we solve again transportation

problem. As a result, this algorithm repeats until no improvement exists.

Choosing the initial facility locations can be done by using randomly generated coordinates. We determine the initial point for facilities by using the following procedure. We sort the cost function value $\bar{c_j} = (c_{ikj} - \lambda_j)$ in non-decreasing order for each candidate point k where λ_j $(j = 1, .., n)$ represents the Lagrange multipliers, then we calculate min $z = \sum_{j=1}^{n} (c_{ikj} - \lambda_j)\ u_{ikj}$. According to the new costs $(c_{ikj} - \lambda_j)$ we determine new locations.

5.2. A Lagrangean Heuristic

We apply Lagrangean relaxation with subgradient optimization for solving RLAP. In order to solve RLAP by Lagrangean relaxation, we relax Constraints (5.6) with optimal objective function value z_{LR} to obtain relaxed RLAP (RRLAP).

RRLAP(λ):

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

$s.t.$

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

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

$$u_{ikj} \geq 0 \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K$$

After rearranging the objective function we obtain the following equivalent form:

$$\min z_{LR} = \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{j=1}^{n}(c_{ikj} - \lambda_j)\, u_{ikj} + \sum_{j=1}^{n}\lambda_j d_j \tag{5.11}$$

s.t.

$$\sum_{k=1}^{K} z_{ik} = 1 \qquad i = 1, ..., m \tag{5.12}$$

$$\sum_{j=1}^{n} u_{ikj} = s_i z_{ik} \qquad i = 1, ..., m;\ k = 1, ..., K \tag{5.13}$$

$$0 \leq u_{ikj} \leq d_j \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n \tag{5.14}$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K \tag{5.15}$$

Note that Constraints (5.14) $u_{ikj} \leq d_j$ are added to the formulation. These constraints are redundant for the original problem RLAP. They improve the objective function value z_{LB} of the relaxed problem. For given λ_j values, the second term of the objective function $\sum_{j=1}^{n}\lambda_j d_j$ is constant. So it can be disregarded for the minimization problem. After solving the problem we will add this constant to compute optimal objective value of the problem.

Note that RRLAP(λ) decomposes by facility, that is, each facility can be considered separately. Let RRLAP$^i(\lambda)$ denotes the relaxed subproblem for facility i, with optimal objective function value z^i_{LR}.

RRLAP$^i(\lambda)$:

$$\min\ z_{LR}^i = \sum_{k=1}^{K}\sum_{j=1}^{n} (c_{ikj} - \lambda_j)\, u_{ikj} \tag{5.16}$$

$s.t.$

$$\sum_{k=1}^{K} z_{ik} = 1 \qquad i = 1, ..., m \tag{5.17}$$

$$\sum_{j=1}^{n} u_{ikj} = s_i z_{ik} \qquad i = 1, ..., m;\ k = 1, ..., K \tag{5.18}$$

$$0 \leq u_{ikj} \leq d_j \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n \tag{5.19}$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K \tag{5.20}$$

In order to solve RRLAP$^i(\lambda)$ we can use a search procedure. The procedure first evaluates each candidate point for a given supplier. We assume (λ) is fixed. Facility i has to be located at a candidate point k. Our aim is to find the best candidate, so we adopt a search procedure where for each point k we determine those customers that are supplied from facility i located at point k in order to minimize the shipment cost $\sum_{j=1}^{n} (c_{ikj} - \lambda_j)\, u_{ikj}$. This is accomplished by calculating the coefficients $\widetilde{c}_j = (c_{ikj} - \lambda_j)$ for each customer j and sorting them in non-decreasing order. $d_{j(1)}, d_{j(2)}, ..., d_{j(n)}$ show the demand of the customers in this order. Starting with the customer having the smallest coefficient $\widetilde{c}_j$, we determine the quantities which have to be sent to the customers in such a way that Constraints (5.18) and (5.19) are not violated. The quantity shipped to the first customer in the list is found as the minimum of the remaining capacity of facility i and the demand of that customer. Initially, the remaining capacity, s_i^{rem}, of the facility is equal to its capacity s_i. If the demand of this customer is totally satisfied, i.e. $d_{j(1)} \leq s_i^{rem}$, we update the remaining capacity of facility i as $s_i^{rem} \leftarrow s_i^{rem} - d_{j(1)}$. If the remaining capacity does not allow satisfying all the demand of customer, we ship all the remaining capacity. In other words, for any customer $j(l)$, the shipment $u_{ikj}\,(l)$ is equal to $min\left(s_i^{rem}, d_{j(l)}\right)$. As a result, when facility i is located at point k, we denote the objective value as z_{LR}^{ik}. We repeat the above procedure for each candidate point $k = 1, ..., K$ and the optimal location of facility i occurs at point

for which z_{LR}^{ik} is minimum. That is $z_{LR}^{i} = \min_k z_{LR}^{ik}$. We can summarize this method in Algorithm 4.

Algorithm 4 Search Method for RRLAP$^i(\lambda)$

begin

1. For each candidate point $k = 1, ..., K$ do the following.
2. Sort the customer in non-decreasing order with respect to $\widetilde{c}_j = (c_{ikj} - \lambda_j)$.
   Let the customers in this order be given as $d_{j(1)}, d_{j(2)}, ..., d_{j(n)}$.
3. Let $s_i^{rem} = s_i$, $l = 1$
4. Do
   $$u_{ikj(1)} = min\left(s_i^{rem}, d_{j(1)}\right)$$
   $$s_i^{rem} \leftarrow s_i^{rem} - d_{j(1)}$$
   $$l \leftarrow l + 1$$
   while$(s_i^{rem} \neq 0)$
5. Calculate z_{LR}^{ik}.
6. $z_{LR}^{i} = \min_k z_{LR}^{ik}$.

end;

After solving subproblems RRLAP$^i(\lambda)$ to optimality, the optimal objective value of RRLAP(λ) computed as:

$$z_{LR} = \sum_{i=1}^{m} z_{LR}^{i} + \sum_{j=1}^{n} \lambda_j d_j \tag{5.21}$$

z_{LR} is a lower bound on the optimal solution of problem RLAP for any Lagrange multiplier λ_j. To find the best lower bound, we consider the Lagrangean dual problem $max_\lambda\ z_{LR}$. We can improve the lower bound by using subgradient optimization method described in the Algorithm 5. Let define parameters, λ_j^c and λ_j^p denote initial Lagrange multipliers and previous iteration Lagrange multipliers respectively. N is the counter to keep the number of consecutive non-improvement, α is the stopping criteria and t is the iteration number.

Algorithm 5 Subgradient Optimization for RRLAP(λ)

begin

1. Initialization; Set $\lambda_j^c \leftarrow 0 \ \ (j = 1, ..., m)$, $Best\ z_{LR} \leftarrow 0$ and $\pi \leftarrow 2$.

2. Solve relaxed problems RRLAP(λ). Let z_{LR} be the optimum value of RRLAP(λ) with the current values of λ_j^c.

3. If $z_{LR} > Best\ z_{LR}$ then $Best\ z_{LR} \leftarrow z_{LR}$ and set $N \leftarrow 0$

   else

   $N \leftarrow N + 1$

   if $N \geq 10$ then $\pi \leftarrow \pi/2$ and $N \leftarrow 0$

4. Using the locations obtained in Step 2, calculate cost matrix and solve the transportation problem.

   Let z_{UB} be the current optimum upper bound for the transportation problem.

5. If $t = 1$ then $Best\ z_{UB} \leftarrow z_{UB}$

   else

   if $Best\ z_{UB} > z_{UB}$ then $Best\ z_{UB} \leftarrow z_{UB}$

6. Define subgradients G_j for the relaxed constraints, evaulated at the current solution, by:

$$G_j = d_j - \sum_{i=1}^{m} \sum_{k=1}^{K} u_{ikj}$$

7. Calculate step size ν, by:

$$\nu = \frac{\pi(Best\ z_{UB} - z_{LR})}{\sum_{j=1}^{n} G_j^2}$$

8. Update the Lagrange multipliers; Set $\lambda_j^p \leftarrow \lambda_j^c$ and $\lambda_j^c \leftarrow \lambda_j^c + \nu G_j$

9. Check convergence;

   if $\max_j \left(\mid \lambda_j^P - \lambda_j^c \mid \right) \leq \alpha$, then stop

   else go to Step 2 to resolve RRLAP(λ) with this new set of multipliers.

end;

At each iteration we have to compute the step size to update the current Lagrange multipliers. For this purpose we have to find an upper bound, z_{UB}, on the objective function of RLAP, which can be found by using a method which provides a good feasible solution of RLAP. We use three approaches. The first one is to use a constant value which is any feasible solution value of RLAP. It can be found by randomly setting the facilities to any candidate points in the convex hull. According to facility and the customer locations we compute the cost coefficients $\widetilde{c}_{ij} = c_{ij}\,(\mid x_i - a_j \mid + \mid y_i - b_j \mid)$. Here c_{ij} is the given unit cost of shipment from facility i to customer j. So we can solve this transportation problem and we obtain a feasible solution of RLAP.

The second one is by solving the transportation problem where the locations of the facilities are obtained while generating the lower bound z_{LR}. Note that, if any calculated upper bound is not better than the current upper bound than the old one is kept as the current best.

The last method to compute an upper bound is based on the two-phase heuristic where the initial locations of the facilities are provided again during the lower bound calculations. During the course of Lagrangean relaxation with subgradient optimization we use these three methods. According to our observations using the two-phase algorithm to obtain an upper bound gives better results than the others.

Our experimentations has shown that Lagrangean relaxation with subgradient optimization procedure is a very efficient method to solve RLAP. The main advantage of the Lagrangean relaxation is that we can solve large instances of RLAP whereas the branch and bound method can only solve relatively small instances. On the other hand, the major drawback of the Lagrangean relaxation is the duality gap which prevents us from finding an optimal solution for the RLAP. Namely, the best lower bound obtained by solving the Lagrangean dual problem, may not be equal to the optimal objective value of RLAP. The duality gap determines the quality of solution. In order to eliminate the duality gap in the next Chapter 6 we will study Semi-Lagrangean relaxation to solve RLAP.

6. SOLVING THE RECTILINEAR DISTANCE LOCATION-ALLOCATION PROBLEM BY SEMI-LAGRANGEAN RELAXATION

6.1. Three Semi-Lagrangean Approaches

We adopt semi-Lagrangean relaxation to reduce the duality gap given by the standard Lagrangean relaxation courses in the solution of RLAP. First we relax the complicating constraints. Second, in order to strengthen semi-Lagrangean dual problem we insert relaxed constraints into the minimization problem as inequalities. Final, we solve each sub problems step by step using subgradient optimization algorithm.

6.1.1. Relaxing Demand And Assignment Constraints

In Step 1 of the semi-Lagrangean relaxation a subproblem obtained by relaxing two equality constraints of the original mixed integer problem is solved to optimality. Solution space of the new problem is much larger than the original solution space which causes the results of Step 1 to be poor in terms of closeness to the original optimal solution. In Step 2 the first one of the relaxed equality constraints is added to the model as inequalities. This reduces the size of the solution space drastically. Finally, we add the second relaxed equality constraints in the form of less than or equal to constraints in Step 3. In our cases, for the rectilinear distance location-allocation problem, an optimal solution is always obtained and therefore Step 4 seems unnecessary.

We first relax Constraints (5.6) with Lagrange multipliers λ_j , $j = 1, ..., n$ and Constraints (5.7) with $\beta_i, i = 1, ..., m$ to obtain the relaxed RLAP (RRLAP).

Step 1 formulation $\text{RRLAP}_1(\lambda, \beta)$:

$$\min \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{j=1}^{n} c_{ikj}u_{ikj} + \sum_{j=1}^{n}\lambda_j \left(d_j - \sum_{i=1}^{m}\sum_{k=1}^{K} u_{ikj} \right) + \sum_{i=1}^{m}\beta_i \left(1 - \sum_{k=1}^{K} z_{ik} \right)$$

s.t.

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

$$u_{ikj} \geq 0 \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K$$

After rearranging the objective function we obtain the following equivalent form:

$$\min \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{j=1}^{n} (c_{ikj} - \lambda_j)\, u_{ikj} - \sum_{i=1}^{m}\sum_{k=1}^{K}\beta_i z_{ik} + \sum_{j=1}^{n}\lambda_j d_j + \sum_{i=1}^{m}\beta_i \tag{6.1}$$

s.t.

$$\sum_{j=1}^{n} u_{ikj} = s_i z_{ik} \qquad i = 1, ..., m;\ k = 1, ..., K \tag{6.2}$$

$$0 \leq u_{ikj} \leq d_j \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n \tag{6.3}$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K \tag{6.4}$$

Step 2 formulation $\text{RRLAP}_2(\lambda, \beta)$:

$$\min \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{j=1}^{n} (c_{ikj} - \lambda_j)\, u_{ikj} - \sum_{i=1}^{m}\sum_{k=1}^{K}\beta_i z_{ik} + \sum_{j=1}^{n}\lambda_j d_j + \sum_{i=1}^{m}\beta_i \tag{6.5}$$

s.t.

$$\sum_{i=1}^{m}\sum_{k=1}^{K} u_{ikj} \leq d_j \qquad j = 1, ..., n \tag{6.6}$$

$$\sum_{j=1}^{n} u_{ikj} = s_i z_{ik} \qquad i = 1, ..., m; k = 1, ..., K \tag{6.7}$$

$$0 \leq u_{ikj} \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n \tag{6.8}$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K \tag{6.9}$$

Step 3 formulation $\text{RRLAP}_3(\lambda, \beta)$:

$$\min \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{j=1}^{n}(c_{ikj}-\lambda_j)\,u_{ikj} - \sum_{i=1}^{m}\sum_{k=1}^{K}\beta_i z_{ik} + \sum_{j=1}^{n}\lambda_j d_j + \sum_{i=1}^{m}\beta_i \tag{6.10}$$

$s.t.$

$$\sum_{i=1}^{m}\sum_{k=1}^{K}u_{ikj} \le d_j \qquad j = 1, ..., n \tag{6.11}$$

$$\sum_{k=1}^{K}z_{ik} \le 1 \qquad i = 1, ..., m \tag{6.12}$$

$$\sum_{j=1}^{n}u_{ikj} = s_i z_{ik} \qquad i = 1, ..., m;\ k = 1, ..., K \tag{6.13}$$

$$0 \le u_{ikj} \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n \tag{6.14}$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K \tag{6.15}$$

6.1.2. Relaxing Demand Constraints

We have found an optimal solution for all our test problems by using previous adaptation. However, time requirement is very high. Aras et al. [2] point out that Equalities (5.6) and (5.8) can be replaced, respectively, by inequalities $\geq$ and $\leq$ but this does not affect the optimal solution since the problem is balanced. Moreover, Constraints (5.7) become redundant when the problem is balanced. Notice that our all test instances are balanced. As a result we decide to relax only one constraint instead of two.

So demand equality Constraints (5.6) are relaxed with Lagrange multipliers λ_j $j = 1, ..., n$ to yield relaxed problem RRLAP.

Step 1 formulation $\text{RRLAP}_1(\lambda)$:

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

$s.t.$

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

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

$$0 \leq u_{ikj} \leq d_j \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K$$

After rearranging the objective function we obtain the following equivalent form:

$$\min \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{j=1}^{n} \left(c_{ikj} - \lambda_j\right) u_{ikj} + \sum_{j=1}^{n}\lambda_j d_j \tag{6.16}$$

$s.t.$

$$\sum_{k=1}^{K} z_{ik} = 1 \qquad i = 1, ..., m \tag{6.17}$$

$$\sum_{j=1}^{n} u_{ikj} = s_i z_{ik} \qquad i = 1, ..., m;\ k = 1, ..., K \tag{6.18}$$

$$0 \leq u_{ikj} \leq d_j \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n \tag{6.19}$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K \tag{6.20}$$

Step 2 formulation RRLAP$_2(\lambda)$:

$$\min \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{j=1}^{n} (c_{ikj} - \lambda_j)\, u_{ikj} + \sum_{j=1}^{n} \lambda_j d_j \tag{6.21}$$

$s.t.$

$$\sum_{i=1}^{m}\sum_{k=1}^{K} u_{ikj} \leq d_j \qquad j = 1, ..., n \tag{6.22}$$

$$\sum_{k=1}^{K} z_{ik} = 1 \qquad i = 1, ..., m \tag{6.23}$$

$$\sum_{j=1}^{n} u_{ikj} = s_i z_{ik} \qquad i = 1, ..., m;\ k = 1, ..., K \tag{6.24}$$

$$u_{ikj} \geq 0 \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n \tag{6.25}$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K \tag{6.26}$$

6.1.3. A Semi-Lagrangean Heuristic

So far, in order to solve RLAP we have studied Lagrangean relaxation, and two semi-Lagrangean approaches with subgradient optimization method. We can reach optimum solution in terms of lower bound by using semi-Lagrangean approaches. Nevertheless, these approaches are slower than Lagrangean relaxation in terms of total execution time. As a result, we decide to relax demand inequality Constraints (6.22) with optimal objective function value z_{SLR}. Constraints (6.22) are relaxed with Lagrange multipliers $\beta_j \quad j = 1, ..., n$ to obtain RRLAP.

RRLAP(λ, β):

$$\min\ z_{SLR} = \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{j=1}^{n}(c_{ikj} - \lambda_j)\,u_{ikj} + \sum_{j=1}^{n}\lambda_j d_j + \sum_{j=1}^{n}\beta_j\left(d_j - \sum_{i=1}^{m}\sum_{k=1}^{K}u_{ikj}\right)$$

s.t.

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

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

$$0 \le u_{ikj} \le d_j \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K$$

After rearranging the objective function we obtain the following equivalent form:

$$\min\ z_{SLR} = \sum_{i=1}^{m}\sum_{k=1}^{K}\sum_{j=1}^{n}\left[(c_{ikj} - \lambda_j) - \beta_j\right]u_{ikj} + \sum_{j=1}^{n}\lambda_j d_j + \sum_{j=1}^{n}\beta_j d_j \tag{6.27}$$

s.t.

$$\sum_{k=1}^{K} z_{ik} = 1 \qquad i = 1, ..., m \tag{6.28}$$

$$\sum_{j=1}^{n} u_{ikj} = s_i z_{ik} \qquad i = 1, ..., m;\ k = 1, ..., K \tag{6.29}$$

$$0 \le u_{ikj} \le d_j \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n \tag{6.30}$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K \tag{6.31}$$

Notice that, for given Lagrange multipliers λ_j and β_j ($j = 1, ..., n$) the last two terms of the objective function $\sum_{j=1}^{n}\lambda_j d_j$ and $\sum_{j=1}^{n}\beta_j d_j$ are both constants. So they can be disregarded during the solution of the subproblem. We will add these constant values to calculate optimal objective value of the problem.

RRLAP(λ, β) is decomposed by facility. RRLAP$^i(\lambda, \beta)$ denotes the relaxed subproblem for facility i, with optimal objective function value z^i_{SLR}.

RRLAP$^i(\lambda, \beta)$:

$$\min\ z^i_{SLR} = \sum_{k=1}^{K}\sum_{j=1}^{n}\left[(c_{ikj} - \lambda_j) - \beta_j\right] u_{ikj} + \sum_{j=1}^{n}\lambda_j d_j + \sum_{j=1}^{n}\beta_j d_j \tag{6.32}$$

s.t.

$$\sum_{k=1}^{K} z_{ik} = 1 \qquad i = 1, ..., m \tag{6.33}$$

$$\sum_{j=1}^{n} u_{ikj} = s_i z_{ik} \qquad i = 1, ..., m; \quad k = 1, ..., K \tag{6.34}$$

$$0 \leq u_{ikj} \leq d_j \qquad i = 1, ..., m;\ k = 1, ..., K;\ j = 1, ..., n \tag{6.35}$$

$$z_{ik} = 0, 1 \qquad i = 1, ..., m;\ k = 1, ..., K \tag{6.36}$$

Constraints $u_{ikj} \leq d_j$ is inserted into the constraints set which is redundant for the original problem RLAP but it is necessary to improve the objective function value of subproblem RRLAP$^i(\lambda, \beta)$. So as to solve relevant subproblem we used a search procedure presented in the Algorithm 6.

Algorithm 6 Search Method for RRLAP$^i(\lambda, \beta)$

begin

1. For each candidate point $k = 1, ..., K$ do the following.
2. Sort the customer in non-decreasing order with respect to

   $\widetilde{c}_j = [(c_{ikj} - \lambda_j) - \beta_j]$.

   Let the customers in this order be given as $d_{j(1)}, d_{j(2)}, ..., d_{j(n)}$.
3. Let $s^{rem}_i = s_i$, $l = 1$.
4. Do

   $u_{ikj(1)} = min\left(s^{rem}_i, d_{j(1)}\right)$

   $s^{rem}_i \leftarrow s^{rem}_i - d_{j(1)}$

   $l \leftarrow l + 1$

   while $(s^{rem}_i \neq 0)$
5. Calculate z^{ik}_{SLR}.
6. $z^i_{SLR} = \min_k z^{ik}_{SLR}$.

end;

First of all, we evaluate each candidate points for a given facilities. Facility i has to be located at a candidate point k. Our purpose is to determine best candidate point for each supplier i. Later we determine relevant customers that are supplied from i located at candidate point k in order to minimize the shipment cost $\sum_{j=1}^{n} [(c_{ikj} - \lambda_j) - \beta_j] u_{ikj}$. This is accomplished by calculating the cost efficients $\widetilde{c}_j = [(c_{ikj} - \lambda_j) - \beta_j]$ for each customer j and sorting them in non-decreasing order. Then assign the demand of customer in this sequence.

After solving subproblems RRLAP$^i(\lambda, \beta)$ to optimality, the optimal objective value of RRLAP(λ, β) can be computed as:

$$z_{SLR} = \sum_{i=1}^{m} z_{SLR}^i + \sum_{j=1}^{n} \lambda_j d_j + \sum_{j=1}^{n} \beta_j d_j \tag{6.37}$$

z_{SLR} is a lower bound on the optimal solution of problem RLAP for any Lagrange multipliers (λ_j, β_j). To find the best lower bound, we take into account semi-Lagrangean dual problem, $\underset{\lambda,\beta}{max} z_{SLR}(\lambda, \beta)$. In order to improve this lower bound we apply subgradient optimization method described in the Algorithm 7 to RRLAP(λ, β).

Algorithm 7 Subgradient Optimization for RRLAP(λ, β)

begin

1. Initialization: Set $\lambda_j^0 = 0$ and $\beta_j^0 = 0$.
2. Repeats Steps 3-5 for a predetermined number of iterations.
3. With the current values of λ and β solve RRLAP(λ, β).
4. $z_{SLR} = \sum_{i=1}^{m} z_{SLR}^i + \sum_{j=1}^{n} \lambda_j d_j + \sum_{j=1}^{n} \beta_j d_j$.
5. Update the Lagrange multipliers:

$\lambda_j^{t+1} = \lambda_j^t + \nu^t (d_j - \sum_{i=1}^{m} \sum_{k=1}^{K} u_{ikj})$ and $\beta_j^{t+1} = \beta_j^t + \nu^t (d_j - \sum_{i=1}^{m} \sum_{k=1}^{K} u_{ikj})$.

where ν^t is the step size given as:

$$\nu^t = \frac{\pi (Best\ z_{UB} - z_{SLR})}{\sum_{j=1}^{n} (d_j - \sum_{i=1}^{m} \sum_{k=1}^{K} u_{ikj})^2}$$

end;

We adopt two alternatives in order to find out an upper bound. The former one is the solution of the transportation problem related to RLAP where the location of facilities are obtained while generating the lower bound z_{SLR}. The latter alternative is based on the two-phase heuristic where the initial location of facilities are randomly selected.

The main advantage of the Lagrangean relaxation is that we can solve relatively large instances of RLAP. Unfortunately, there is a major drawback of Lagrangean relaxation which is the duality gap. It prevents us to find an optimal solution for the RLAP. Our computational results obtained on different instances indicate that the duality gap may reach large values, conversely there are also instances for which the duality gap is zero. Note that by using semi-Lagrangean relaxation we eliminate the duality gap and get optimum solution. A negative aspect of semi-Lagrangean relaxation is that it requires more execution time than Lagrangean relaxation.

To conclude, we suggest that using standard algorithm like branch and bound method is very efficient and quick method to solve small sized problems. Because of the fact that Lagrangean relaxation with subgradient optimization is an iterative algorithm which has to perform many iterations to reach optimum results regardless of the problem size. Another suggestion is that Lagrangean relaxation is very useful method to solve relatively large sized RLAP.

7. COMPUTATIONAL RESULTS

In this chapter we present the result of our computational experiments performed by using the heuristics proposed in the previous sections. In the first subsection we will give implementation details, then in the other sections we will present our computational results.

7.1. Experimental Setup

In this work we are interested in the rectilinear distance location-allocation problem. We can categorize our test problems in two groups. The first group consists of eight test problems, problems 8, 9 and 15 have been obtained from Sherali et al. [27] and the remaining five test problems have been taken from Sherali et al. [26]. The second group consists of seven test problems generated randomly as follows. Observe that the number of facilities to be located and the number of customers to be satisfied by these facilities are two indicators for the size of the problem. In addition, distribution of the customer locations and the resulting shape of the distribution have profound effects on the size of convex hull and the number of candidate points. Before the determination of the problem details like customer locations, corresponding demands of the customers, facility capacities and unit transportation costs, the number of customers and the number of facilities are stated such that the maximum size of the problems of the second group is greater than the maximum size of the first group of problems. Furthermore, the size of the problems within the second group is increased gradually from the first problem to the last.

Having fixed the number of facilities and number of customers, namely the size, specifics of the problem are produced as follows. Initially, an upper bound for the facility capacities are specified as 400 for all the problems. Later on, capacity of each facility is generated randomly within the range (1-400) by using a uniform random number generator. Moreover, for the relevant problem demand of each customer is also generated within the same range by again using a uniform random number generator.

The total supply for the relevant problem is equated to the sum of the generated facility capacities. Similarly, the total demand is obtained as the sum of the generated individual demands. As a consequence of these operations, we end up with excess demand for all the problems because the number of customers are greater than the number of facilities for all problem instances. To make the problems balanced, it is required to normalize the total demand according to total supply. By dividing total supply to total demand we obtain a ratio which leads to final customer demands when multiplied by demand values. Having applied this procedure for all instances we end up with balanced problems in which total demand equals to total supply.

Unit transportation costs and customer locations are obtained in a manner analogous to that process of determination of facility capacities. By making use of a uniform random number generator as in the case of facility capacities, unit transportation costs and X and Y coordinate values are randomly generated within the range from 1 to 10 and from 0 to 750 respectively.

It is to be underlined that the solution space for facility locations is not R^2 but a set of candidate locations that consists of intersection points of the lines that pass through customer locations vertically and horizontally. It is known by Wendell and Hurter [30] and Hansen et al. [11] that an optimal solution is always in the convex hull of customer location points. Therefore, the intersection points that are outside of the convex hull can be eliminated from the solution space. In brief, candidate facility locations consist of the intersection points that lie in the convex hull of customer location points.

To sum up, facility capacities, customer demands, customer locations, unit transportation costs, and the candidate set of facility locations are determined for all problems by applying the procedure mentioned above. For each instances the number of facilities to be located m, the number of customers n, and the number of candidate points K are given in the following Table (7.1).

Table 7.1. Test problem parameters

Problem no	**m**	**n**	**K**
8	4	8	22
9	5	15	70
15	5	10	29
16	4	10	33
23	5	8	30
26	5	12	63
29	5	15	116
30	5	20	190
50	5	30	748
51	10	30	748
52	5	40	1,295
53	10	40	1,295
54	5	50	1,998
55	10	50	1,998
56	20	50	1,998

7.2. Optimal Solutions

The first group problems have been solved to optimality in advance, optimal results and total execution time to reach optimality exist in the literature. In order to have a fair comparison in terms of required time all of the test problems are solved on a PC with 3.0 GHz Pentium 4 processor and 1.0 GBytes of RAM, under the Windows XP operating system.

We not only try to solve test problems to optimality but also apply Lagrangean relaxation, semi-Lagrangean relaxation, and two-phase method. To accomplish this task we use XA [29] and CPLEX [10] solvers through the instrumentality of GAMS [15] interface. It is realized that XA and CPLEX solvers become inefficient in terms of time requirement for large sized problems. Due to this reason especially for the second group test problems we decided to encode Lagrangean relaxation method in Microsoft

Visual Basic 6.0. The code which implements the transportation simplex method in solving the transportation subproblems of the two-phase method was adapted from the software available at ORLAB [22].

All of the test problems are solved to optimality by both of XA 14 [29] and CPLEX 9.1 [10] solvers. Optimum results and total execution times of solvers are presented in Table 7.2. Both of the solvers managed to reach the optimal solution for all test problems. Nevertheless, one may easily realize that CPLEX solver is seriously more efficient than XA solver in terms of time requirements. This realization is influenced our decision of employing CPLEX solver instead of XA for group 2 test problems which are considerably larger.

Table 7.2. Comparison of XA and CPLEX solvers

Problem no	**Opt.**	**XA TET(sec)**	**CPLEX TET(sec)**
8	793	0.55	0.34
9	9,619	2,399.30	3.06
15	3,427	6.98	0.34
16	259	11.22	1.28
23	238	1.45	0.36
26	284	2,488.31	3.41
29	729	44,168.78	25.17
30	745	1,708,572.18	187.86
Average		219,706.10	27.73

We present optimum results, the number of node sizes, and total execution time of solved RLAP test instances in Table 7.3. They are used as a reference in the rest of the study. The optimal results are obtained using the branch and bound algorithm for different values of the optimality tolerance φ, CPLEX 9.1 [10] was used as solver. From problem no 8 to 52 were solved with $\varphi = 0.00$ whereas problem no 53 φ takes on value 0.10. One important consideration that can be extracted from Table 7.3 is the fact that the required time to reach the optimality grows exponentially as problem size increases. Because of this reason we are not able to solve the problems to optimality that have more than 1300 candidate points for facility locations.

Note that the problems 54, 55, and 56 could not be solved to yield an exact solution within a reasonable range of time. Because of this fact the mentioned problems do not have an optimal solution. Hence, branch and bound algorithm is not applicable to considerably large problems.

Table 7.3. Optimum results of the test problems

Problem no	**Opt.**	**Node size**	**TET(sec)**
8	793	0+	0.34
9	9,619	0+	3.06
15	3,427	0+	0.34
16	259	408	1.28
23	238	100+	0.36
26	284	400	3.41
29	729	1,200	25.17
30	745	2,500	187.86
50	62,317	5,700	10,496
51	48,750	10,000	13,951
52	241,564	147,400	612,680
53	44,650	17,200	95,419
Average			61,064

7.3. Performance of the Heuristics

In this section we compare the new heuristics with the existing methods both in terms of solution quality and execution time efficiency on a number of test instances.

All Lagrangean relaxation applications include subgradient optimization. Consider that each iteration of the subgradient algorithm we compute an upper bound by solving a transportation problem between the customers and the facility locations which are determined during the calculation of the lower bound. It is important note that the upper bound obtained throughout the iterations is not only an upper bound on the optimal objective value of the RRLAP, but also on that of the original RLAP.

The update of the Lagrange multipliers is performed via the subgradient optimization of Algorithm 5 and 7. The initial value of the π is set to 2 and halved every 10 iterations when there is no improvement in the lower bound. Also final precision α is set to $0,001$ and the maximum number of iterations is selected as 500.

During the interpretation of all results we use the following abbreviations: LB and UB represent lower bound and upper bounds respectively, and Opt denotes the optimal objective value of test problems. Finally TET indicate total execution time in the seconds and NA means not applicable.

Table 7.4. Lagrangean relaxation results

Problem no	**Opt.**	**LB**	**% Dev.**	**UB**	**% Dev.**	**TET(Sec)**
8	793	739	6.81	793	0.00	8
9	9,619	9,551	0.71	9,649	0.31	30
15	3,427	3,348	2.31	3,501	2.16	49
16	259	243	6.18	407	57.14	14
23	238	225	5.46	244	2.52	12
26	284	274	3.52	436	53.52	24
29	729	717	1.65	729	0.00	33
30	745	730	2.01	774	3.89	84
50	62,317	60,328	3.19	62,548	0.37	815
51	48,750	45,665	6.33	51,396	5.43	1,326
52	241,564	237,898	1.52	241,564	0.00	2,810
53	44,650	41,840	6.29	50,341	12.75	3,548
54	NA	214,981	NA	222,676	NA	5,599
55	NA	261,191	NA	281,976	NA	14,593
56	NA	133,753	NA	236,743	NA	24,325
Average			3.83		11.51	3,551.33

We applied Lagrangean relaxation method for test problems and obtained lower and upper bounds for the optimal solution and total execution time that are presented

in the Table 7.4. The percentage deviations from the optimal objective value calculated as (*optimal value* − *lower bound*) / *optimal value* for lower bounds and (*upper bound* − *optimal value*) / *optimal value* for upper bounds.

Table 7.5. Lagrangean relaxation with the two-phase method results

Problem no	**Opt.**	**LB**	**% Dev.**	**UB**	**% Dev.**	**TET(sec)**
8	793	740	6.68	793	0.00	15
9	9,619	9,551	0.71	9,634	0.16	78
15	3,427	3,348	2.31	3,501	2.16	20
16	259	243	6.18	259	0.00	35
23	238	225	5.46	238	0.00	17
26	284	269	5.28	284	0.00	34
29	729	717	1.65	729	0.00	50
30	745	729	2.15	746	0.13	350
50	62,317	58,357	6.35	62,317	0.00	1,543
51	48,750	44,845	8.01	53,485	9.71	2,456
52	241,564	237,183	1.81	241,564	0.00	3,374
53	44,650	41,013	8.15	45,035	0.86	8,414
54	NA	214,564	NA	221,420	NA	7,611
55	NA	261,311	NA	280,310	NA	18,453
56	NA	131,886	NA	158,389	NA	33,330
Average			4.56		1.09	5,052.00

From Table 7.4 it is observed that lower bounds deviate less than 7% from the optimal solutions for all test instances. On the average, lower bound deviation is around 4% which is a clear indication of the lower bounds qualities. In addition, except the problems 16 and 26 upper bounds deviate less then 7% from the optimal solutions whereas average upper bounds deviation is around 12%.

However, although all of the results are obtained in reasonable execution time it can be noticed that total execution time of CPLEX solvers are better than Lagrangean relaxation execution times for the first group of test problems. Therefore, it

is improper to refer to Lagrangean relaxation method for small size problems. Nevertheless, execution time does not grow exponentially by increasing problem size when Lagrangean relaxation is employed which makes this method and appropriate one for large problems.

Table 7.6. Demand and assignment constraints relaxed results

P.no	**Opt.**	**LB1**	**% Dev.**	**LB2**	**% Dev.**	**LB3**	**% Dev.**	**TET(sec)**
8	793	347.7	56.16	793.0	0.00	793	0.00	183.1
9	9,619	680.9	92.92	9,466.5	1.59	9,619	0.00	96,539.0
15	3,427	822.5	76.00	3,423.9	0.09	3,427	0.00	35,290.0
16	259	221.1	14.66	259.0	0.0	259	0.00	670.1
23	238	8.1	96.63	222.2	6.6	238	0.00	1,995.0
26	284	41.9	85.24	251.9	11.3	284	0.00	2,283.2
29	729	159.3	78.15	703.7	3.5	729	0.00	83,813.8
30	745	712.8	4.33	745.0	0.0	745	0.00	20,101.1
Avg.			63.01		2.89		0.0	30,109.4

Lagrangean relaxation gives tight lower bounds but weak upper bounds. Using the two-phase method [9] is known to be an efficient way of obtaining good upper bounds for the location-allocation problem we eliminated most of the duality gap in terms of upper bound. The two-phase method iteratively improves the solution by solving allocation problem for fixed locations of facilities in Phase 1, and by solving location problem having assigned allocations at hand in Phase 2. Algorithm continues to applying Phase 1 and 2 alternately until there is no improvement in the best obtained solution. Because of this extra burden total execution time of the two-phase is greater than without the two-phase. When we interpret the result of Table 7.5, we figure out the efficiency of the two-phase method. Average upper bounds deviation is around 1% on the contrary it is 11% without the two-phase Lagrangean relaxation.

In this part of the study we make use of a quite new approach claiming to improve lower bound of Lagrangean relaxation method, namely semi Lagrangean relaxation method. Demand and assignment constraints both were relaxed. Lower bounds and its percentage deviation are presented in Table 7.6.

Table 7.7. Demand constraints relaxed results

P.no	**Opt.**	**LB1**	**% Dev.**	**UB1**	**% Dev.**	**LB2-UB2**	**% Dev.**	**TET(s)**
8	793	738.6	6.87	793	0.00	793	0	23.5
9	9,619	9,551	0.71	9,776	1.63	9,619	0	208.8
15	3,427	3,348	2.31	3,501	2.16	3,427	0	189.1
16	259	243.5	6.00	407	57.14	259	0	100.1
23	238	224.8	5.55	238	0.00	238	0	54.4
26	284	263.2	7.33	474	66.90	284	0	89.9
29	729	714.1	2.04	729	0.00	729	0	240.4
30	745	732.6	1.66	760	2.01	745	0	8,042.9
Avg.			4.1		16.23		0.00	1,118.6

Although the optimum results are attained there is a major drawback of semi-Lagrangean relaxation method that it is quite poor in terms of execution time in comparison with the Lagrangean relaxation method. Therefore, semi-Lagrangean relaxation is an inappropriate heuristic for large problems. We realized that semi-Lagrangean relaxation method is very useful to find good quality lower bound, but this new technique required more time.

In order to diminish total execution time, we decided to relax only demand constraints. In our perspective, we hope to find lower bound within a reasonable time.

Considering the results in Table 7.7 the lower and upper bounds are the optimum results of the original problem. However, the time needed to find these values are not as good as those calculated for the original problem. Furthermore, when we consider the solution quality, the new technique yields excellent results, but the time needed to solve the problem is not within the desired ranges. So, it is observed that the new technique may not provide very efficient results for the solution of large problems.

In order to find lower and upper bounds with in a reasonable time we relaxed demand inequality constraints. Relevant results are represented in Table 7.8 and 7.9. Compared to Lagrangean relaxation results, the average improvement with the pro-

Table 7.8. Semi-Lagrangean heuristic results

Problem no	**Opt.**	**LB**	**% Dev.**	**UB**	**% Dev.**	**TET(sec)**
8	793	740	6.68	793	0.00	14
9	9,619	9,551	0.71	9,649	0.31	47
15	3,427	3,348	2.31	3,501	2.16	58
16	259	243	6.18	407	57.14	23
23	238	225	5.46	244	2.52	24
26	284	274	3.52	436	53.52	35
29	729	717	1.65	729	0.00	54
30	745	731	1.88	774	3.89	139
50	62,317	60,328	3.19	62,548	0.37	1,119
51	48,750	45,665	6.33	51,396	5.43	1,998
52	241,564	238,094	1.44	241,564	0.00	4,970
53	44,650	41,884	6.19	50,341	12.75	6,804
54	NA	215,035	NA	222,676	NA	10,501
55	NA	261,384	NA	281,976	NA	23,810
56	NA	133,753	NA	236,743	NA	35,112
Average			3.79		11.51	5,647.20

posed method is less than 1% in terms of lower and upper bounds. It is clear that the results not as good as we have expected. When Table 7.8 is examined, it is noted that the upper bounds are too poor for most of the test problems. A very efficient method in finding an upper bound is the two-phase method [9]. So it was used to improve them starting of the location obtained by the subgradient algorithm.

In table 7.9 the percent deviation of the lower and upper bound from the objective value are given in columns four and six respectively. The last column of the table reports the total execution time spent by the semi-Lagrangean heuristic with the two-phase method to obtain the lower and upper bounds.

Table 7.9. Semi-Lagrangean heuristic with the two-phase method results

Problem no	**Opt.**	**LB**	**%Dev.**	**UB**	**% Dev.**	**TET(sec)**
8	793	741	6.56	793	0.00	26
9	9,619	9,551	0.71	9,634	0.16	104
15	3,427	3,348	2.31	3,501	2.16	31
16	259	243	6.18	259	0.00	43
23	238	225	5.46	238	0.00	28
26	284	271	4.58	284	0.00	53
29	729	717	1.65	729	0.00	72
30	745	732	1.74	746	0.13	462
50	62,317	59,104	5.16	62,317	0.00	2,755
51	48,750	44,996	7.70	53,485	9.71	5,180
52	241,564	237,236	1.79	241,564	0.00	6,429
53	44,650	41,228	7.66	45,035	0.86	16,028
54	NA	214,972	NA	221,420	NA	13,954
55	NA	261,311	NA	280,310	NA	25,187
56	NA	132,076	NA	158,389	NA	59,133
Average			4.29		1.09	8,632.33

When implemented semi-Lagrangean heuristic with the two-phase, this new heuristic produced a very significant enhancement in upper bounds. Although the upper bounds are substantially improved by the two-phase method, these is instance for the percent deviation is high like problem no 51. On the other hand, lower bounds decreased slightly compared with the method where only semi-Lagrangean heuristic was used.

8. CONCLUSIONS

Throughout the study, we are concerned with the discrete rectilinear distance location-allocation problems. Location-allocation problem is a special type of facility location problems which is an example of NP-hard combinatorial problems. Algorithms like branch and bound aiming to solve such problems to optimality with complete enumeration are efficient only for small sized problems because of NP-hard nature of the problem. As a consequence of this fact, there has been an intense interest in developing some heuristics that try to obtain near optimal solution by searching a relatively small subset of whole search space for the last several decades. Advance of techniques in especially area of information technologies made the implementation of the developed heuristics possible and much more efficient. A popular example of heuristics which is known as Lagrangean relaxation helps to obtain an interval in which the optimal solution lies on by relaxing the complicating constraints of the problem. Pioneering work of Cooper [9] which is known as the two-phase algorithm is one of the earliest methods implemented for location-allocation problems.

At the initial phases of the study we have made use of the Lagrangean relaxation with subgradient optimization for our test problems. By comparing our results with the optimal values of the problems which are known to have obtained optimal solutions in the literature we have noticed that although our lower bound results were fairly close to the optimal values, upper bound values were relatively poor. Consequently, we have decided to incorporate with the two-phase algorithm's results to be used as upper bound value through the subgradient procedure. This integrated approach has helped us to obtain good results for both upper and lower bound values with cost of tolerable deterioration in total execution time. However, we have not been able to get rid of duality gap which in turn have led our attention to apply semi-Lagrangean relaxation to RLAP.

Behind the logic of the semi-Lagrangean relaxation, there lies the idea of relaxing some of the equality constraints of the original problem and iteratively adding them

back in the form of inequalities. We have employed this method and have obtained optimal values by the end of at most three steps. However, average total execution time was very poor compared to ordinary Lagrangean relaxation results. During the analysis of semi-Lagrangean relaxation results we have faced with the surprising fact that the results obtained at the end of the first step in which two of the equality constraints are simultaneously relaxed, is quite poor which is a strong indication of the incorrectness of the relaxing more than one constraint. This is the motivation behind the development of semi-Lagrangean relaxation in which only one of the equality constraints is relaxed and added back to the problem in the following step in the form of inequalities. Obtained results were equal to optimal values and we have also managed to decrease the execution time for all of the test problems. Although the last relaxation have led us to obtain optimal values in tolerable execution time, we have also found out that addition of the relaxed constraints back to the problem causes the total execution time to exponentially grow with increasing problem size. Thus, this new constraint is complicating in the senses that when it is relaxed through the procedure of conventional Lagrangean relaxation, it enables us to obtain lower and upper bounds in which more reasonable execution times comperad with the former relaxations. At this moment we have decided to implement this newly proposed semi-Lagrangean heuristic for all of the test problems. By observing less than 5% deviation from the optimal value for all the problems that have known optimal values, it can be expected to have good solutions for the large problems.

In brief, we have also verified the recognized result of the quality of Lagrangean relaxation and the two-phase method [9] for the location-allocation problem. Furthermore, we have also ended up with the judgment that through little modifications depending on the relevant problem and integration of different algorithms it is possible to obtain hybrid algorithms which enable obtaining better results.

APPENDIX A: TEST PROBLEMS

Problem 8 $m = 4$, $n = 8$

c_{ij} = 4 5 9 12 14 7 8 5
0 5 7 12 13 15 21 8
10 24 25 5 22 12 16 19
18 16 2 12 22 3 21 12

s_i = 26, 23, 1, 15 i = 1,...,4.

d_j = 15, 6, 10, 11, 5, 10, 1, and 7 j = 1,...,8.

(a_j, b_j) = (0,6), (0,10), (0,12), (4,16), (5,9), (1,13), (4,19), and (10,8) j = 1,...,8.

Problem 9 $m = 5$, $n = 15$

c_{ij} = 11 7 5 4 1 8 3 12 10 15 12 6 11 8 16
19 9 14 13 0 19 10 25 13 25 23 25 6 25 20
20 16 12 22 3 20 16 12 1 25 6 3 3 14 9
10 2 7 11 24 21 16 19 4 12 2 25 8 22 13
15 16 15 15 13 25 3 17 0 7 22 7 1 25 20

s_i = 10, 20, 9, 74, 226 i = 1,...,5.

d_j = 36, 41, 11, 26, 11, 12, 41, 3, 10, 26, 25, 8, 45, 42, and 2 j = 1,...,15.

(a_j, b_j) = (0,8), (0,2), (3,4), (4,0), (7,19), (13,11), (5,15), (7,17), (8,2), (0,7), (8,7), (8,14), (6,5), (7,17), and (1,15) j = 1,...,15.

Problem 15 $m = 5$, $n = 10$

c_{ij} = 8 11 9 2 5 8 4 4 2 4
14 14 10 21 18 9 19 12 20 6
1 17 7 21 25 4 3 0 17 12
19 25 11 7 25 16 12 15 20 5
19 0 13 1 5 7 5 20 6 25

s_i = 180, 2, 20, 42, 100 i = 1,...,5.

d_j = 49, 29, 10, 57, 25, 61, 14, 3, 38, and 58 j = 1,...,10.

(a_j,b_j) = (0,12), (0,21), (0,25), (0,20), (1,9), (2,25), (1,22), (6,21), (15,15), and (14,10) j = 1,...,10.

Problem 16 m = 4, n = 10

c_{ij} = 1

s_i = 28, 18, 22, 22 i = 1,...,4.

d_j = 8, 11, 8, 7, 3, 13, 10, 13, 5, and 12 j = 1,...,10.

(a_j,b_j) = (10,4), (2,17), (13,14), (0,16), (17,0), (21,14), (15,13), (10,0), (2,0), and (0,10) j = 1,...,10.

Problem 23 m = 5, n = 8

c_{ij} = 1

s_i = 7, 3, 22, 26, 1 i = 1,...,5.

d_j = 2, 6, 6, 3, 9, 14, 5, and 14 j = 1,...,8.

(a_j,b_j) = (24,23), (5,9), (18,0), (13,15), (20,12), (0,4), (24,5), and (14,8) j = 1,...,8.

Problem 26 m = 5, n = 12

c_{ij} = 1

s_i = 27, 8, 20, 25, 27 i = 1,...,5.

d_j = 4, 1, 11, 14, 1, 14, 8, 11, 10, 13, 7, and 13 j = 1,...,12.

(a_j,b_j) = (25,10), (6,2), (19,3), (14,3), (21,0), (1,8), (25,6), (15,1), (23,12), (9,21), (0,5), and (15,12) j = 1,...,12.

Problem 29 m = 5, n = 15

c_{ij} = 1

s_i = 30, 37, 15, 48, 2 i = 1,...,5.

d_j = 8, 11, 10, 13, 7, 13, 10, 3, 8, 10, 11, 11, 8, 1, and 8 j = 1,...,15.

(a_j,b_j) = (25,5), (6,2), (19,10), (14,8), (21,3), (1,14), (25,23), (15,7), (23,23), (9,5), (0,1), (15,18), (12,22), (4,0), and (5,15) j = 1,...,15.

Problem 30 m = 5, n = 20

c_{ij} = 1

s_i = 29, 34, 32, 31, 33 i = 1,...,5.

d_j = 11, 11, 8, 1, 8, 9, 11, 5, 15, 1, 3, 5, 14, 3, 12, 8, 15, 15, 2, and 2 j = 1,...,20.

(a_j,b_j) = (25,14), (6,23), (19,7), (14,23), (21,5), (1,1), (25,18), (15,22), (23,0), (9,23), (0,12), (15,17), (12,17), (4,21), (5,11), (5,21), (2,17), (10,4), (8,13), and (3,20) j = 1,...,20.

Problem 50 m = 5, n = 30

c_{ij} = 2 6 9 2 5 1 9 6 0 5 1 1 7 2 0 2 3 3 9 9 4 2 1 1 6 4 4 7 3 6
2 1 5 0 4 9 2 7 3 2 9 6 6 4 0 5 6 9 8 0 5 9 4 6 5 5 4 3 4 2
0 2 9 0 3 3 4 1 4 2 6 5 1 9 6 5 3 1 7 4 7 5 8 0 2 0 1 3 1 0
5 6 5 8 0 1 6 4 3 1 7 9 5 0 7 4 4 4 2 3 0 5 1 9 0 4 2 8 7 2
6 2 0 0 3 7 2 2 4 2 3 0 4 2 8 5 7 9 3 5 0 6 4 9 1 9 6 3 1 4

s_i = 142, 107, 116, 58, 61 i = 1,...,5.

d_j = 21, 0, 21, 23, 20, 1, 11, 24, 22, 10, 27, 24, 2, 26, 10, 15, 21, 2, 16, 13, 8, 17, 18, 7, 8, 23, 23, 16, 27, and 28 j = 1,...,30.

(a_j,b_j) = (646,434), (243,104), (335,153), (695,91), (20,241), (383,646), (376,284), (593,578), (470,505), (697,237), (346,289), (486,125), (296,380), (570,378), (299,356), (159,433), (342,476), (620,259), (211,205), (105,370), (156,409), (254,613), (334,133), (478,523), (429,547), (113,565), (141,669), (46,43), (555,265), and (324,30) j = 1,...,30.

Problem 51 m = 10, n = 30

c_{ij} = 1 9 6 0 5 1 1 7 2 0 2 3 3 9 9 4 2 1 1 6 4 4 7 3 6 2 1 5 0 4

9 2 7 3 2 9 6 6 4 0 5 6 9 8 0 5 9 4 6 5 5 4 3 4 2 0 2 9 0 3

3 4 1 4 2 6 5 1 9 6 5 3 1 7 4 7 5 8 0 2 0 1 3 1 0 5 6 5 8 0

1 6 4 3 1 7 9 5 0 7 4 4 4 2 3 0 5 1 9 0 4 2 8 7 2 6 2 0 0 3

7 2 2 4 2 3 0 4 2 8 5 7 9 3 5 0 6 4 9 1 9 6 3 1 4 2 9 1 0 3

5 9 5 4 8 8 6 7 9 3 4 4 6 1 4 5 8 5 4 5 2 6 4 6 8 3 3 2 1 5

2 5 3 8 4 1 6 7 6 7 1 8 2 9 0 0 7 3 4 1 1 1 0 7 5 5 2 4 7 7

3 9 7 0 6 7 0 4 4 2 9 8 6 4 7 2 3 4 9 1 6 3 1 1 0 4 9 5 4 9

2 3 3 2 5 1 5 9 5 9 6 4 6 0 7 7 4 1 2 3 7 0 5 7 8 3 9 8 6 9

8 4 1 9 7 6 4 0 1 1 5 4 1 2 6 8 4 1 8 3 3 7 2 4 2 8 6 3 3 8

s_i = 142, 107, 116, 58, 61, 155, 3, 153, 163, 142 i = 1,...,10.

d_j = 3, 27, 56, 51, 24, 62, 56, 4, 61, 23, 34, 50, 4, 38, 30, 19, 40, 42, 17, 18, 53, 53, 38, 64, 59, 15, 45, 63, 16, and 35 j = 1,...,30.

(a_j,b_j) = (646,434), (243,104), (335,153), (695,91), (20,241), (383,646), (376,284), (593,578), (470,505), (697,237), (346,289), (486,125), (296,380), (570,378), (299,356), (159,433), (342,476), (620,259), (211,205), (105,370), (156,409), (254,613), (334,133), (478,523), (429,547), (113,565), (141,669), (46,43), (555,265), and (324,30) j = 1,...,30.

Problem 52 m = 5, n = 40

c_{ij} = 7 5 7 1 8 8 5 2 3 4 8 1 6 8 9 4 10 9 7 10 9 5 2 10 8 7 5 1 2 2 6 5 2 3 7 9 5 2 9 4

4 8 3 5 3 9 7 4 4 9 6 10 6 4 9 3 3 2 4 1 8 9 3 8 5 9 8 5 1 5 4 8 4 8 2 6 10 3 10 2

10 7 6 10 6 3 9 1 5 8 3 4 4 6 3 6 3 1 9 2 7 10 3 10 9 5 5 8 9 4 2 4 5 2 7 1 3 7 6 2

10 5 1 5 3 8 9 9 2 2 1 2 4 2 10 9 6 8 2 3 1 9 8 8 4 5 6 9 6 3 3 4 6 4 6 5 6 1 6 3

10 10 10 9 5 9 10 3 2 7 1 4 7 2 2 3 4 6 4 3 9 4 4 5 9 5 8 7 5 10 7 6 3 6 8 5 10 3 7 9

s_i = 114, 45, 95, 151, 152 i = 1,...,5.

d_j = 11, 25, 21, 3, 18, 20, 1, 12, 11, 8, 27, 22, 19, 12, 20, 8, 10, 12, 26, 4, 18, 10, 3, 5, 2, 12, 26, 15, 14, 27, 6, 11, 11, 8, 14, 4, 14, 27, 15, and 25 j = 1,...,40.

(a_j,b_j) = (136,228), (289,106), (433,69), (143,484), (353,128), (693,329), (2,298), (202,526), (623,575), (123,76), (7,106), (264,131), (685,610), (407,510), (101,180), (19,579), (548,553), (232,315), (388,562), (394,142), (145,255), (401,240), (369,297), (401,35), (383,144), (673,653), (694,587), (286,563), (692,180), (122,486), (11,237), (484,77),

(76,203), (279,391), (217,206), (565,216), (263,280), (588,286), (507,461), and (284,40) j = 1,...,40.

Problem 53 m = 10, n = 40

c_{ij} = 2 3 3 9 9 4 2 1 1 6 4 4 7 3 6 2 1 5 0 4 9 2 7 3 2 9 6 6 4 0 5 6 9 8 0 5 9 4 6 5
5 4 3 4 2 0 2 9 0 3 3 4 1 4 2 6 5 1 9 6 5 3 1 7 4 7 5 8 0 2 0 1 3 1 0 5 6 5 8 0
1 6 4 3 1 7 9 5 0 7 4 4 4 2 3 0 5 1 9 0 4 2 8 7 2 6 2 0 0 3 7 2 2 4 2 3 0 4 2 8
5 7 9 3 5 0 6 4 9 1 9 6 3 1 4 2 9 1 0 3 5 9 5 4 8 8 6 7 9 3 4 4 6 1 4 5 8 5 4 5
2 6 4 6 8 3 3 2 1 5 2 5 3 8 4 1 6 7 6 7 1 8 2 9 0 0 7 3 4 1 1 1 0 7 5 5 2 4 7 7
3 9 7 0 6 7 0 4 4 2 9 8 6 4 7 2 3 4 9 1 6 3 1 1 0 4 9 5 4 9 2 3 3 2 5 1 5 9 5 9
6 4 6 0 7 7 4 1 2 3 7 0 5 7 8 3 9 8 6 9 8 4 1 9 7 6 4 0 1 1 5 4 1 2 6 8 4 1 8 3
3 7 2 4 2 8 6 3 3 8 5 9 5 3 8 2 2 1 3 0 7 8 2 7 4 8 7 4 0 4 3 7 3 7 1 5 9 2 9 1
9 6 5 9 5 2 8 0 4 7 2 3 3 5 2 5 2 0 8 1 6 9 2 9 8 4 4 7 8 3 1 3 4 1 6 0 2 6 5 1
9 4 0 4 2 7 8 8 1 1 0 1 3 1 9 8 5 7 1 2 0 8 7 7 3 4 5 8 5 2 2 3 5 3 5 4 5 0 5 2

s_i = 142, 107, 116, 58, 61, 155, 3, 153, 163, 142 i = 1,...,10.

d_j = 3, 22, 46, 42, 20, 51, 46, 3, 50, 19, 28, 41, 3, 31, 25, 16, 33, 34, 14, 15, 44, 44, 31, 52, 48, 12, 37, 52, 13, 28, 6, 53, 36, 1, 31, 6, 6, 42, 15, and 1 j = 1,...,40.

(a_j,b_j) = (136,228), (289,106), (433,69), (143,484), (353,128), (693,329), (2,298), (202,526), (623,575), (123,76), (7,106), (264,131), (685,610), (407,510), (101,180), (19,579), (548,553), (232,315), (388,562), (394,142), (145,255), (401,240), (369,297), (401,35), (383,144), (673,653), (694,587), (286,563), (692,180), (122,486), (11,237), (484,77), (76,203), (279,391), (217,206), (565,216), (263,280), (588,286), (507,461), and (284,40) j = 1,...,40.

Problem 54 m = 5, n = 50

c_{ij} = 5 3 2 2 7 5 5 8 4 7 3 2 6 1 5 10 3 8 4 3 10 7 7 5 1 6 7 10 9 1 6 10 5 7 6 6 5 4 5 3
1 3 10 1 4 4 5 2 5 3
7 6 2 10 7 6 4 2 8 5 8 6 9 1 3 1 2 4 2 1 6 7 6 9 1 2 7 5 4 2 8 10 6 1 8 5 5 5 3 4 1
6 2 10 1 5 3 9 8 3
7 3 1 1 4 8 3 3 5 3 4 1 5 3 9 6 8 10 4 6 1 7 5 10 2 10 7 4 2 5 3 10 2 1 4 6 10 6 5

9 9 7 8 10 4 5 5 7 2 5

6 9 6 5 6 3 7 5 7 9 4 4 3 2 6 3 6 4 9 5 2 7 8 7 8 2 9 3 10 1 1 8 4 5 2 2 2 1 8 6 6 3 5 8 8 4 10 8 1 7

8 1 5 5 3 10 9 7 5 8 3 4 5 10 2 7 4 2 2 1 5 10 6 5 10 3 4 4 3 6 2 6 10 6 10 7 5 7 1 8 8 5 2 3 4 8 1 6 8 9

s_i = 151, 116, 125, 67, 70 i = 1,...,5.

d_j = 15, 1, 15, 16, 14, 2, 8, 17, 15, 8, 18, 17, 2, 18, 7, 10, 15, 2, 12, 9, 6, 12, 13, 6, 6, 16, 16, 12, 19, 17, 5, 14, 19, 5, 11, 3, 19, 13, 1, 11, 3, 3, 15, 6, 2, 6, 8, 6, 18, and 17 j = 1,...,50.

(a_j,b_j) = (451,361), (156,407), (523,319), (633,198), (467,623), (196,257), (9,206), (688,532), (406,423), (187,355), (555,600), (65,379), (290,51), (315,489), (349,284), (661,689), (414,372), (430,62), (129,198), (13,106), (294,383), (390,546), (30,284), (41,218), (422,362), (417,230), (636,539), (403,302), (512,114), (41,170), (531,665), (241,478), (554,24), (555,338), (144,421), (636,661), (457,10), (463,675), (6,667), (51,205), (407,565), (28,38), (395,220), (288,357), (513,315), (225,51), (286,401), (534,406), (456,565), and (291,50) j = 1,...,50.

Problem 55 m = 10, n = 50

c_{ij} = 6 10 3 10 2 10 7 6 10 6 3 9 1 5 8 3 4 4 6 3 6 3 1 9 2 7 10 3 10 9 5 5 8 9 4 2 4 5 2 7 1 3 7 6 2 10 5 1 5 3

8 9 9 2 2 1 2 4 2 10 9 6 8 2 3 1 9 8 8 4 5 6 9 6 3 3 4 6 4 6 5 6 1 6 3 10 10 10 9 5 9 10 3 2 7 1 4 7 2 2

3 4 6 4 3 9 4 4 5 9 5 8 7 5 10 7 6 3 6 8 5 10 3 7 9 3 4 1 3 10 8 6 7 3 6 8 9 1 6 5 1 5 7 5 5 10 10 6 6 7

1 2 3 1 2 5 6 6 8 1 5 1 4 7 6 6 4 10 8 6 5 8 2 1 3 8 10 4 7 8 1 8 5 3 7 10 10 7 1 7 10 1 10 1 3 6 9 1 1 6

4 5 6 8 5 4 1 5 6 8 6 7 9 5 8 8 7 8 2 8 1 3 10 9 3 2 3 10 3 3 8 7 8 8 4 9 8 9 3 4 6 4 3 6 7 2 1 7 9 7

5 2 5 10 8 4 5 6 7 1 7 9 4 2 5 8 4 5 8 5 8 6 2 3 1 10 3 7 4 9 4 8 2 4 1 5 5 6 1 4 2 4 8 7 10 9 6 7 2 10

4 7 6 8 8 7 8 3 6 6 6 3 5 4 10 6 6 5 1 4 2 7 2 2 4 1 7 8 5 5 4 8 3 1 9 1 5 8 8 9 3 5

9 9 5 5 4 3 9 6

6 7 1 8 3 1 3 10 4 7 4 4 4 3 3 5 7 3 7 6 7 1 1 2 3 4 6 3 9 9 6 4 1 3 2 7 10 4 3 9 2 6 7 7 7 9 1 4 9 3

7 4 9 5 9 3 8 9 3 10 10 9 6 7 5 2 7 7 9 4 1 6 1 4 5 3 10 1 7 7 6 1 8 4 10 3 4 4 4 3 6 8 3 6 6 10 10 3 9 8

5 7 5 5 2 10 2 6 7 2 7 8 8 5 1 5 6 1 5 2 2 4 10 7 3 5 5 1 5 5 10 3 2 6 4 6 4 3 1 7 4 6 9 5 6 4 1 2 2 2

s_i = 75, 204, 170, 144, 191, 185, 93, 34, 200, 169 i = 1,...,10.

d_j = 42, 26, 4, 12, 12, 32, 26, 9, 19, 39, 51, 31, 13, 54, 24, 21, 47, 15, 28, 16, 53, 38, 24, 25, 52, 36, 56, 32, 22, 52, 17, 17, 13, 22, 3, 45, 51, 19, 43, 26, 49, 45, 28, 7, 26, 22, 43, 21, 48, and 9 j = 1,...,50.

(a_j,b_j) = (451,361), (156,407), (523,319), (633,198), (467,623), (196,257), (9,206), (688,532), (406,423), (187,355), (555,600), (65,379), (290,51), (315,489), (349,284), (661,689), (414,372), (430,62), (129,198), (13,106), (294,383), (390,546), (30,284), (41,218), (422,362), (417,230), (636,539), (403,302), (512,114), (41,170), (531,665), (241,478), (554,24), (555,338), (144,421), (636,661), (457,10), (463,675), (6,667), (51,205), (407,565), (28,38), (395,220), (288,357), (513,315), (225,51), (286,401), (534,406), (456,565), and (291,50) j = 1,...,50.

Problem 56 m = 20, n = 50

c_{ij} = 9 9 10 3 7 7 10 5 1 9 2 3 7 4 6 10 6 10 8 5 8 6 5 1 1 4 5 4 7 2 5 2 10 3 1 9 8 3 8 9 10 3 5 10 10 6 9 6 5 9

8 9 2 8 7 10 9 8 8 5 8 6 2 4 5 9 9 4 10 6 10 5 8 4 1 9 6 9 9 8 10 1 9 3 3 1 10 10 3 6 4 3 3 2 7 6 9 1 3 7

5 3 9 4 10 4 8 9 7 4 6 7 6 1 9 1 2 4 5 5 9 1 9 3 1 8 1 2 7 3 9 3 3 6 7 7 2 10 4 4 1 9 5 5 10 3 6 1 9 5

6 4 2 8 3 3 4 5 8 2 5 9 6 8 3 1 7 6 7 4 7 2 10 4 9 2 5 1 8 9 3 5 6 1 7 10 1 8 1 10 10 7 7 2 2 10 7 9 2 2

3 1 4 10 3 8 9 1 9 9 3 2 7 1 7 5 10 2 3 7 3 4 1 1 10 8 5 5 5 9 5 8 4 4 3 3 1 8 9 4 5 10 7 5 9 4 2 10 6 8

7 4 10 4 4 3 8 9 2 10 7 9 8 4 6 9 2 8 5 6 1 1 9 10 1 7 10 2 4 5 10 2 9 7 3 1 10 5 7

1 1 10 3 9 5 2 3 8 9 3

4 9 9 5 6 2 10 3 1 4 7 4 7 8 8 2 1 3 2 10 2 7 10 3 3 8 8 4 2 6 2 3 9 7 5 6 3 1 3 10 1 9 1 8 3 10 1 6 5 9

5 9 9 1 6 6 8 5 10 2 7 7 8 5 10 9 2 7 3 9 8 4 1 7 2 7 4 1 7 6 6 9 10 3 2 8 9 4 9 8 4 6 4 6 5 6 8 4 3 9

3 2 2 3 2 6 10 4 3 7 3 7 4 1 10 3 7 10 4 4 5 8 3 9 8 5 2 2 3 10 1 4 8 3 4 2 4 5 10 1 8 1 7 7 8 5 8 2 10 4

1 3 7 2 9 2 5 10 5 10 5 8 8 3 2 7 10 8 6 4 3 8 3 7 7 2 10 9 7 1 5 1 7 8 5 1 6 7 9 4 3 7 4 6 7 7 2 7 10 8

10 2 4 8 3 10 10 9 10 2 10 2 8 8 10 5 10 2 5 4 6 1 4 8 1 3 1 9 3 8 10 8 5 7 3 1 4 3 5 4 7 5 9 5 5 7 7 10 1 6

1 7 2 5 5 10 6 8 3 9 9 4 9 6 3 9 5 10 1 10 9 3 3 5 7 3 10 2 7 1 9 1 5 5 2 1 4 1 10 10 6 1 7 8 4 6 2 7 3 6

9 6 10 4 7 2 8 8 9 9 7 3 5 6 1 2 10 3 9 6 2 9 3 4 1 5 4 7 4 10 8 1 4 4 6 3 10 5 3 10 8 10 5 4 9 1 1 8 3 4

8 3 4 3 5 10 7 9 5 4 9 6 4 6 1 9 5 6 2 2 7 1 10 10 9 9 10 4 9 4 5 9 9 7 5 2 7 2 1 4 6 4 7 6 3 3 5 5 4 10

4 4 10 3 3 3 7 6 10 5 6 4 2 1 9 5 5 4 1 10 6 3 5 4 9 8 9 3 2 5 2 2 5 9 7 8 1 3 6 10 8 2 7 7 10 6 6 4 5 9

10 7 10 2 9 10 4 10 5 7 6 4 5 8 7 6 1 7 9 4 5 10 6 5 2 6 3 6 8 1 6 5 6 3 7 4 7 9 4 3 2 9 4 6 3 4 9 4 3 1

4 10 1 10 2 3 7 2 7 6 3 2 7 2 2 1 2 3 5 7 8 5 4 8 10 5 2 8 2 2 7 3 9 5 2 8 9 4 2 7 7 9 9 3 9 9 2 1 1 3

3 1 6 5 3 7 1 5 1 6 6 10 5 3 10 4 5 10 5 8 2 3 9 1 3 1 3 4 8 6 9 8 7 8 10 3 4 6 10 10 8 4 2 6 1 3 4 10 9 3

1 4 8 10 7 5 3 3 8 8 2 8 2 3 5 4 9 7 7 9 4 4 9 9 9 9 8 6 4 7 2 3 8 1 4 1 8 5 8 5 5 8 8 6 4 1 6 5 4 7

7 6 7 1 4 7 8 9 7 1 7 10 10 7 6 8 3 7 5 6 4 4 3 5 9 4 1 7 7 8 5 1 5 8 6 6 9 6 9 4 2 2 8 2 3 1 5 4 7 9

s_i = 162, 62, 37, 18, 148, 178, 113, 170, 206, 114, 23, 135, 75, 65, 146, 90, 86, 51, 185, 40 i = 1,...,20.

d_j = 36, 52, 48, 62, 14, 30, 4, 66, 77, 28, 22, 80, 19, 37, 33, 30, 82, 50, 46, 14, 75, 6,

67, 61, 23, 79, 41, 61, 36, 22, 52, 48, 41, 55, 65, 9, 16, 12, 8, 48, 80, 22, 32, 75, 42, 22, 71, 9, 14, and 82 j = 1,...,50.

(a_j,b_j) = (451,361), (156,407), (523,319), (633,198), (467,623), (196,257), (9,206), (688,532), (406,423), (187,355), (555,600), (65,379), (290,51), (315,489), (349,284), (661,689), (414,372), (430,62), (129,198), (13,106), (294,383), (390,546), (30,284), (41,218), (422,362), (417,230), (636,539), (403,302), (512,114), (41,170), (531,665), (241,478), (554,24), (555,338), (144,421), (636,661), (457,10), (463,675), (6,667), (51,205), (407,565), (28,38), (395,220), (288,357), (513,315), (225,51), (286,401), (534,406), (456,565), and (291,50) j = 1,...,50.

REFERENCES

1. Al-Loughani, L., *Algorithmic approaches for solving the Euclidian distance location-allocation problems*, Ph.D. Dissertation, Industrial and System Engineering, Virginia Polytechnic Institue and State University, Blacksburgh, Virginia, 1997.

2. Aras, N., İ.K. Altınel, and M. Orbay, *New heuristic methods for the capacitated multi-facility Weber problem*, Research Paper Series No: FBE-IE-04/2005-05, Boğaziçi University, Bebek, İstanbul, 2005.

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

4. Beasley, J.E., *Modern heuristic techniques for combinatorial problems*, Ed.by C.R. Reeves, Wiley Publishing, New York, 1993.

5. Beltran, C., C. Tadonki, and J. Vial, "Semi-Lagrangean relaxation", http://www.unige.ch/hec/logilab/template/papiers/papier77OptOnLine.pdf, (last accessed May 15, 2006).

6. Bongartz, I., P.H. Calamai and A.R. Conn, "A projection method l_p norm location-allocation problems", *Mathematical Programing,* Vol. 66, pp. 283-312, 1994.

7. Cooper, L., "Location-allocation problems", *Operations Research*, Vol. 11, pp.130-145, 1963.

8. Cooper, L., "Heuristic method for location-allocation Problem", *SIAM Review*, Vol. 6, pp.37-52, 1964.

9. Cooper, L., "The transportation location problem", *Operational Research*, Vol. 20, pp.94-108, 1972.

10. *Cplex solver manuel,* http://www.gams.com/dd/docs/solvers/cplex.pdf , (last accessed May 15, 2006).

11. Hansen, P., J. Perreur, and F. Thisse, "Location theory, dominance and convexity: Some further results", *Operational Research* Vol. 18, pp.1241-1250, 1980.

12. Held, M., P.Wolfe, and H.P. Crowder, "Validation of subgradient optimization ", *Mathematical Programing*, Vol. 6, pp. 63-88, 1974.

13. Francis, R. L., "A note on the optimum location of new machines in existing plant location", AIIE Transactions, Vol. 14, pp. 57-59, 1963.

14. Fisher M., "The Lagrangean relaxation method for solving integer programing problems", *Management Science*, Vol.27, No.1, 1981.

15. *Gams user guide,* http://www.gams.com/docs/gams/GAMSUsersGuide.pdf, (last accessed May 15, 2006).

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

17. Graham, R.L., "An efficient algorithm for determining the convex hull of a finite planer set", *Inform process lett.* Vol. 7, pp. 175-180, 1972.

18. Lorena, Luiz A.N. and Edson L.F. Senne, "A column generation approach to capacitated p-median problems", *Computers and Operations Research*, Article in Press, http://elsevier.com.

19. Love, R.F., H. Juel.,"Properties and solution method for large location-allocation problems", *Journal of Operational Research Society*, Vol. 33, pp. 443-452, 1982.

20. Lozano, S.,F.Guerreno., "Kohonen Maps for solving a class of location allocation problems", *European Journal of Operational Research*,Vol. 108, pp. 106-107

21. Murtagh, B.A. and S.R. Niwattisyawong, "An efficient method for the multi-depot location-allocation problem", *Journal of the Operational Research Society*, Vol. 33, pp. 629-634, 1982.

22. *Operations Research Labratory,* http://www.orlab.org (last accessed May 15, 2006).

23. Salsi, S. and M.D.H. Gamal, "A celluler heuristic for the multisource Weber problem", *Computers and Operations Research*, Vol. 30, pp. 1609-1624, 2003.

24. Shapiro J.F., "A survey of Lagrangean techniques for discrete optimization", *Analysis of Discrete Mathematics*,Vol. 5, pp. 113-138, 1979.

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

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

27. Sherali, H.D., I. Al-Loughani and S.Subramanian, "Global optimization procedures for the capacitated Euclidean and l_p distance multifacility location allocation problems", *Operational Research*,Vol. 50, pp. 443-448, 2002.

28. Taha, Hamdy A., *Introduction to Operations Research*, 6th ed., Prentice-Hall Inc., New Jersey, 1997.

29. *XA solver manuel,* http://www.gams.com/dd/docs/solvers/xa.pdf, (last accessed May 15, 2006).

30. Wendell, R.E. and A.P. Hunter, "Location Theory, dominance and convexity", *Operations Research*, Vol. 21, pp. 314-320, 1973.