HEURISTIC METHODS FOR CONTINUOUS CAPACITATED LOCATION-ALLOCATION AND LOCATION-ROUTING PROBLEMS

by

Sadettin Yumuşak

B.S. in Industrial Engineering, İstanbul Technical University, 2003

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

2005

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisors, Assoc. Prof. Necati Aras and Prof. İ. Kuban Altınel for teaching me the scientific method, improving my analytical thinking, guiding me to question every assumption and helping me to rebuild my philosophy of life throughout the course of this work.

I would like to thank to Assist. Prof. Deniz Aksen for his serving on my thesis committee.

I thank to my friend Vural Erol for enhancing my interest on scientific research and guiding me to begin this work.

Finally, I give thanks to my parents for their support to my education.

This research has been partially supported by the Boğaziçi University Research Fund Grant No: 03HA302.

ABSTRACT

HEURISTIC METHODS FOR CONTINUOUS CAPACITATED LOCATION-ALLOCATION AND LOCATION-ROUTING PROBLEMS

In this work, we consider two problems and develop heuristic methods for their solution. The first one is the capacitated continuous location-allocation problem and can be defined as locating facilities with capacities in order to satisfy the demands of existing customers at the minimum cost. This cost is a function of the distance between the facilities and customers. Distances can be measured as the ℓ_p, rectilinear, Euclidean or Squared Euclidean distance. In the capacitated location-allocation problem, if the locations of the facilities are given, the problem turns into the transportation problem. The solution to the location-allocation problem always occurs in the basic feasible solution set of the transportation problem. When the flows are fixed, then single facility location problems are obtained each of which can be solved sperately. Using the neighborhood structure developed, simulated annealing, threshold accepting and genetic algorithm heuristics are proposed for the solution of the problem.

Continuous capacitated location-routing problem is the integration of the location-allocation and vehicle routing problems. Vehicles depart from a facility, serve one or more customers and return back to the same facility. The objective is the minimization of the total route lengths. The facilities have limited capacity and can be located anywhere on the continuous plane. In this thesis, a self organizing map heuristic is proposed for this problem.

ÖZET

SÜREKLİ DÜZLEMDE KAPASİTE KISITLI ÇOK TESİSLİ YER SEÇİMİ-PAYLAŞTIRMA PROBLEMİ İLE YER SEÇİMİ-ROTALAMA PROBLEMLERİ İÇİN SEZGİSEL YÖNTEMLER

Bu çalışmada, ele alınan iki problemin çözümleri için sezgisel yöntemler geliştirdik. İlk problem, yer seçimi ve paylaştırma problemi; koordinatları verilen müşteri noktalarının taleplerini karşılamak üzere kapasitesi bilinen tesislerin en az maliyetle yerleştirilmesi olarak tanımlanabilir. Maliyet, tesisler ve müşteriler arasındaki uzaklığın bir fonksiyonudur. Kullanılan uzaklık fonksiyonları ℓ_p, doğrusal, Öklidyen ve karesel Öklidyen uzaklıklardır. Kapasite kısıtlı tesis yerleştirme probleminde arz noktalarının yerleri verildiğinde problem ulaştırma problemine dönüşmektedir. Yer seçimi-paylaştırma problemine en uygun çözüm ise, elde edilen ulaştırma probleminin temel uygun çözüm kümesinde bulunmaktadır. Malzeme akışları sabit tutulduğunda ise, her biri bağımsız çözülebilen tek tesisli yerleştirme problemi elde edilir. Problemin çözümü için, geliştirilen komşuluk yapısını kullanarak tavlama benzetimi, eşik kabul etme ve genetik algoritma sezgiselleri önerildi.

Sürekli düzlemde kapasite kısıtlı rotalama ve yer seçimi problemi; yer seçimi ve paylaştırma problemi ile, araç rotalama probleminin bütünleşmesinden oluşmuştur. Bir tesisten ayrılan araçlar, taleplerini karşılayabildikleri bir veya birden fazla müşterilere uğrayıp aynı tesise geri dönmektedir. Amaç toplam rota uzunluğunu en azlamaktır. Tesisler sınırlı kapasiteye sahip olup, sürekli düzlemde her hangi bir noktaya yerleştirilebilir. Bu tezde, problemin çözümü için ön düzenlemeli harita (SOM) yöntemi önerildi.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

a_i	Activation of node i in ANN
a_{j1}, a_{j2}	Coordinates of customer j
B	Bias term which prevents the overload of vehicle
c	Index of the winner node in SOM
c_{ij}	Unit shipment cost per unit distance from facility i to customer j
c_{gh}	Cost of direct travel from node g to node h
cap	Vehicle capacity
d	Distance function
$dist$	Index distance between neurons in SOM
E	Parameter used for termination of SA and TA
$f(S)$	Objective value of the solution
G	Genrations in GA
$iter$	Iteration
k	Boltzmann constant in the Metropolis algorithm
K	Parameter used for termination of SA and TA
l_{fx}, l_{fy}	Coordinates of the facility f in LRP
L	Number of iterations during which temperature T is left unchanged
m	Number of facilities for LAP
M_α	Learning rate momentum coefficient
M_σ	Neighborhood width momentum coefficient
n	Number of customers
N	Population size in GA
N_c	Neighborhood set of winner node c
N_x	Number of input vectors in SOM
NS	Neigborhood size
o_i	Output vector of node i in ANN
o_d	Desired output in ANN

p	Number of vehicles or probability of acceptance in SA and TA
q_j	Demand of customer j
Q	Subset of V
r	Parameter for setting L
r_j	Index vector of neuron j in SOM
$runs$	Total number of runs for the same number and combination of vehicle assignments
s_i	Capacity of facility i
S	Solution
T	Temperature in SA
Th	Threshold in TA
u_i	Index of node i in ANN
V	Set of customers
v	Coefficient of bias term B
veh_f	Number of vehicles assigned to depot f
w_{ij}	Flow of products from facility i to customer j
w_j	Weight vector of node j in ANN
x_i	Input vector i in SOM
x_{i1}, x_{i2}	Coordinates of facility i in LAP
x_{ghk}	Variable indicating that node g comes before node h on route k
y_{gk}	Variable indicating the assignment of node g to route k
α	Cooling rate in SA and TA, or learning rate in ANN
δE	Energy of magnitude in the Metropolis algorithm
Δ	Magnitude of the deterioration in SA and TA
ℓ_p	A metric measure for pth dimension
Λ	Neighborhood function
Φ	Set of neurons in the input layer
Ψ	Set of neurons in the output layer
σ	Width of the neighborhood function in SOM

ALA	Alternate Location-Allocation
ANN	Artificial Neural Network
CMFWP	Capacitated Multi-Facility Weber Problem
ECMFWP	Euclidean distance Capacitated Multi-Facility Weber Problem
FCTP	Fixed Charged Transportation Problem
GA	Genetic Algorithm
LAP	Location-Allocation Problem
LP	Linear Programming
LpCMFWP	ℓ_p distance Capacitated Multi-Facility Weber Problem
LRP	Location-Routing Problem
LRPSOM	SOM algorithm for LRP
LVQ	Learning Vector Quantization
MDVRP	Multi Depot Vehicle Routing Problem
MFWP	Multi Facility Weber Problem
MTSP	Multiple Traveling Salesman Problem
OR	Operations Research
RCMFWP	Rectilinear distance Capacitated Multi Facility Weber Problem
RLT	Reformulation Linearization Technique
SA	Simulated Annealing
SECMFWP	Squared Euclidean distance Capacitated Multi Facility Weber Problem
SOM	Self Organizing Map
SOMW	Weizsfeld procedure applied after SOM
TS	Tabu Search
VQ	Vector Quantization
VRP	Vehicle Routing Problem

1. INTRODUCTION

In many logistic environments managers have to make decisions on where to place their depots (facilities, factories, warehouses) and how to serve their customers from the depots so as to minimize the total distribution cost. Locating facilities is a common problem that is faced by many disciplines such as architecture, computer science, management science and mathematics. In some distribution systems it is assumed that a tour consists of a visit to a single customer, while in some others more than one customer can be served on a route from the depot. The former system is defined as the Location-Allocation Problem (LAP) while latter is known as the Location-Routing Problem (LRP).

The solution for the LAP is found by determining the closest depot to each customer, while in solving the LRP the aim is to minimize the routes departing from a depot, visiting some customers and returning back to the depot. In the LAP, the facility can be a production plant, a geographical region, a serving point in a city like a fire station, a hospital, a library or a telecommunication service provider. On the other hand, distributions systems such as food and beverage or newspaper delivery are some of the practical settings of the LRP.

In this research we will be interested in the continuous and capacitated versions of the problems explained above: Capacitated Multi-Facility Weber Problem (CMFWP) and Continuous Capacitated Multi-Facility Location-Routing Problem (CCMFLRP).

1.1. Capacitated Multi-Facility Weber Problem

Deterministic LAPs are concerned with locating a set of facilities and allocating their capacity to satisfy the demand of a set of customers with known locations so that the total transportation cost is minimized. Supply centers such as plants and warehouses may constitute the facilities while retailers and dealers may be considered as customers. When the facility locations have to be selected from a given set of

candidate locations, the corresponding location-allocation problem becomes a discrete optimization problem. A continuous LAP is obtained when facilities can be located anywhere in the Euclidean plane. The latter problem is also known as the multi-facility Weber problem (MFWP) [1]. It becomes the so-called single-facility Weber problem if the objective is the determination of an optimal location for a single facility. In some situations, facilities can have capacity constraints, which give rise to the capacitated multi-facility Weber problem (CMFWP). As can be easily observed, in an optimal solution of the uncapacitated problem each customer is served from the nearest facility, which is not true for the more restricted CMFWP because of the capacity constraints. In the CMFWP formulations, the transportation cost is usually assumed to be proportional both to the amount shipped and the distance between the facilities and customers. Then the question becomes how to model the distance mathematically. Different distance functions have been proposed in the literature. Some of them are simple mathematical functions [2] while there exist others which are based on more sophisticated neural network estimators [3], [4]. The ones listed in Table 1.1 are frequently used in location theory. [5]. They all model distances in the two-dimensional Euclidean space but can easily be modified for higher dimensions.

Table 1.1. Most frequently used distance functions in location theory.

Distance	**Formula**
Euclidean	$d_2(\mathbf{x}_1, \mathbf{x}_2) = \left(\lvert x_{11} - x_{21}\rvert^2 + \lvert x_{12} - x_{22}\rvert^2\right)^{1/2}$
Squared Euclidean	$d_2^2(\mathbf{x}_1, \mathbf{x}_2) = \left(\lvert x_{11} - x_{21}\rvert^2 + \lvert x_{12} - x_{22}\rvert^2\right)$
Rectilinear	$d_1(\mathbf{x}_1, \mathbf{x}_2) = \lvert x_{11} - x_{21}\rvert + \lvert x_{12} - x_{22}\rvert$
ℓ_p distance	$d_p(\mathbf{x}_1, \mathbf{x}_2) = (\lvert x_{11} - x_{21}\rvert^p + \lvert x_{12} - x_{22}\rvert^p)^{1/p} \quad p \geq 1$

In most location problems, an explicit function involving an ℓ_p norm is used. A metric measure has properties such as symmetry $d(\mathbf{x}_1, \mathbf{x}_2) = d(\mathbf{x}_2, \mathbf{x}_1)$, nonnegativity $d(\mathbf{x}_1, \mathbf{x}_2) \geq 0$ and identity $d(\mathbf{x}_1, \mathbf{x}_2) = 0 \Leftrightarrow \mathbf{x}_1 = \mathbf{x}_2$. In fact, the Euclidean and rectilinear distances are special cases of the ℓ_p distance. One can obtain the rectilinear distance for $p = 1$ and the Euclidean distance for $p = 2$.

Distance measures used in the LAP differ according to the problem type. In

city centers, the roads are on the edges of rectangular regions that form the city. The distance between any two points in the city can be evaluated by summing the distances of roads on the route between the points. This type of distance is called rectilinear distance. It is also known as Manhattan distance because of the recognition that the street network of Manhattan in New York city is a rectilinear network.

The minimum distance between two points in the plane is a straight line connecting the two points. This form of distance is known as the squared Euclidean distance. Intercity roads may be evaluated closely by calculating the Euclidean distance. It also gives a lower bound between two points.

Real-life location problems cannot be modeled in planar coordinates. In a geographical region there may also be any other constraints rather than capacity of the facilities and demand of the customers. In addition, the location-allocation problem may require three dimensional space to be solved accurately. On the other hand, mathematical modeling of real life problems often requires easier assumptions that enable the problems to be solved in a convenient manner. Hence throughout this study, we make some assumptions for the convenience of the solution. The plane considered is assumed to be an acceptable approximation of the area specifying the problems. Facilities are represented by their planar coordinates and are assumed to occupy no area. Transportation cost of one unit of supply from a facility to a customer is proportional to the distance traveled. More than one facilities can be located on the same point. We assume that there is no fixed cost of opening a facility. The interactions among the facilities and among the customers are ignored. (There is no supply sharing among facilities and no demand sharing among customers.

In this study the capacitated multi-facility Weber problem will be considered. Demands and the planar coordinates of the customers, number and the capacities of the facilities and the transportation costs per unit distance are known. We will try to find out the locations of the facilities and the quantity that will be sent from facilities to the customers.

1.2. Continuous Capacitated Multi-Facility Location-Routing Problem

The LAP ignores serving more than one customer on a route. The error from using moment sum in order to interpret the delivery costs of vehicles is studied by Salhi and Rand [6]. Moment sum is the sum of direct distances between the depot and the customers. It ignores the basic interdependence between location, allocation and routing. The interdependencies between the components of location-routing can be seen in Figure 1.1.

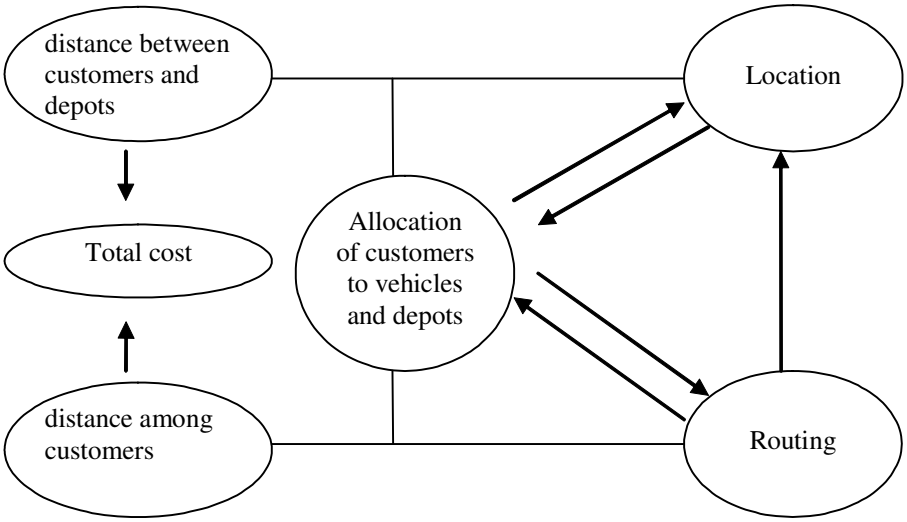

Figure 1.1. LRP Components

Feasible combinations of depot location, routing of vehicles and allocation of customers to depots and vehicles form the feasible solution set of the location-routing problem. The objective is minimizing the total cost of route distances of the vehicles departing from the depots.

LRP aims to find the location of the depots and the routes to the customers from the depots such that the sum of the location and distribution costs is minimized. In addition, capacity constraints for both depots and the vehicles should be considered while minimizing the total cost. It is obvious that, in LRP, the location of depots will influence the choice of customers to be included on specific routes. Thus, LRP may be considered as the combination of two subproblems:

a) The location problem which determines the number and locations of the depots

b) The routing problem which is concerned with the design of routes passing through the customers starting at the depots.

These two subproblems are interdependent, but they are usually treated independently. The interdependence between depot locations and vehicle routes has been recognized by Salhi and Rand [6]. They explain how the solution for the cost function changes if the routes are ignored when locating depots.

In real life problems there are many variations for the location-routing problem proportional to its complexity. In our model we make assumptions different than the ones mostly made in literature. In our model we assume that there is no fixed cost of opening a facility and operating a vehicle. Vehicle capacities are considered instead of the facility capacities. All vehicles are assumed to be identical. Hence, the number of vehicles assigned to the facilities determines the capacity of the facilities.

2. LITERATURE SURVEY

In this chapter, a brief review of the literature on LAP and LRP is given.

2.1. Capacitated Multi-Facility Weber Problem

In his seminal work Cooper [7] considered the Euclidean distance CMFWP (ECMFWP) and proposed an exact solution method that is based on explicit enumeration of all the extreme points of the transportation polyhedron. As the number of extreme points can be very large, this method is useful only for very small instances. As a consequence he suggested two heuristic methods. The first one is called the alternating transportation-location heuristic and is actually very similar in concept to the famous alternating location-allocation (ALA) heuristic [8], which he developed to solve the uncapacitated MFWP with the Euclidean distance. The ALA heuristic makes use of the following characteristic of the uncapacitated problem. Given the locations of the facilities, it is straightforward to determine the allocation of customers to the facilities in which the demand of a customer is served from the closest facility. On the other hand, given the allocation of facilities to the customers, it is possible to find an optimal location for each facility because this phase of the heuristic corresponds to a Euclidean distance single-facility location problem and can be solved to optimality by Weiszfeld's algorithm [9]. Cooper used the same idea to solve the ECMFWP after observing that the CMFWP decomposes into location and allocation (transportation) subproblems. When facilities are located at some points, the resulting transportation problem is solved to determine the corresponding optimal capacity allocations. Then, using these allocations new optimal locations are determined for the facilities. The location and allocation problems are alternately solved until no improvement is possible. It is important to note that the solution method of the single-facility location problems changes with respect to the distance function. For example, Aras et al. use the median location method [5] to solve the rectilinear single-facility location problem in their work on the rectilinear CMFWP (RCMFWP) [10]. It has been shown that Weiszfeld's algorithm can be used to solve optimally the ℓ_p distance single-facility loca-

tion problem for $1 < p < 2$ [11], [12]. As a result, Cooper's alternating two-phase idea can be used to provide approximate solutions for the CMFWP with the ℓ_p distance function (LpCMFWP) for $1 < p < 2$ [10].

Cooper's second heuristic is based on his earlier work [13] and starts by ignoring the capacity constraints and solving the remaining uncapacitated problem. If capacity constraints are violated by the optimal solution of this uncapacitated problem (otherwise, an optimal solution is found), the demand of some customers satisfied by a nonexisting capacity is tried to be met by facilities having extra capacity in such a way that the increase in the objective function is as small as possible. After a feasible allocation is achieved, the procedure is continued with the alternating transportation-location heuristic. Later, he proposed a more efficient heuristic for the ECMFWP [14], which performs a local search over the set of extreme points of the transportation polyhedron. This is done in the neighborhood of a given basic feasible solution where the neighbor solutions are generated by moving to extreme points which are one, two or three steps away from the current one. In other words, when one of the nonbasic variables is exchanged with a basic variable (i.e., a simplex iteration is carried out), an adjacent extreme point is reached. When two nonbasic variables are inserted into the basis simultaneously, then an extreme point adjacent to the immediate neighbor of the current one is obtained. This heuristic can be regarded as an earlier implementation of the variable neighborhood search idea [15].

Apart from Cooper's complete enumeration algorithm [7], there are two more exact methods to solve the ECMFWP for the best of our knowledge: The first one is a biconvex cutting plane procedure and can be found in Selim's unpublished dissertation [16]. Although it is more efficient than Cooper's complete enumeration, this procedure can effectively solve only very small instances. The second one appears in the recent work by Sherali et al. [17], where they design a global optimization procedure for the ECMFWP. This is a branch-and-bound algorithm based on a partitioning of the allocation space, which finitely converges to a global optimum within a specified percentage tolerance. For deriving lower bounds on the subproblems obtained at the nodes of the branch-and-bound tree, the authors use two approaches. The first approach in-

volves computing a lower bound via a projected location space subproblem. In the second approach, a specialized variant of the Reformulation Linearization Technique (RLT) [18] is applied to transform an equivalent representation of the original non-convex problem into a higher dimensional linear programming relaxation. They have shown that the latter approach provides much better lower bounds than the former one. Upper bounds are obtained using the two-phase (alternating location-transportation) heuristic of Cooper [7]. They also generalize this exact method for the solution of the LpCMFWP in the same work [17].

The application of the RLT in the solution of location and allocation problems is not new. Sherali and Tunçbilek [19] apply this approach to solve the squared Euclidean distance CMFWP (SECMFWP). They first set the flow variables to some initial value, and find expressions for the optimal locations in terms of these variables by applying optimality conditions. They then substitute these expressions into the objective function and obtain a minimization problem with a concave quadratic objective function. This is the projection of the original problem onto the space of the flow variables. Finally they use the RLT to generate a linear program which provides a tight lower bound to the objective value of the concave quadratic minimization problem. This linear program is embedded in a branch-and-bound framework which partially enumerates the extreme points of the transportation polyhedron. As a more recent application, Sherali et al. [20] consider the RCMFWP and use the RLT to linearize an equivalent mixed integer nonlinear programming problem of its bilinear programming formulation. This formulation is based on a useful property of the two dimensional rectilinear location problems. Namely, optimal locations always occur within the convex hull of the customer locations and at the intersection points of the vertical and horizontal lines drawn through them [21].

In summary all the exact solution methods use a branch-and-bound approach that is implemented on a transformed version of the problem. Their disadvantage is that they become computationally intensive as the number of variables is increased throughout the procedure. Therefore, efficient heuristic methods are required to solve large-sized instances accurately. In their recent work paper Aras et al. [10] focus on

the ECMFWP, SECMFWP, and LpCMFWP with $1 < p < 2$. They first present a mixed integer linear programming model to approximate these continuous problems. The approximating formulation discretizes continuous problems by using candidate points within the convex hull of customer locations. The number and locations of the candidate points affect the efficiency and quality of the approximation: The larger is the number, the more probable is the set of candidate points include optimal facility locations, but also the higher is the required computational resources. In the luckiest situation candidate points can be exactly optimal facility locations. The authors have proposed a Lagrangean, a p-median and a cellular heuristic to select the set of candidate points and to solve the mixed integer linear approximation. Metaheuristics have been successfully applied to the solution of difficult combinatorial optimization problems during the last decade and for the best of knowledge, besides the early consideration of a variable neighborhood search type strategy by Cooper [14], there is not any systematic application of a metaheuristic strategy to the CMFWP. Motivated by this curious issue we propose a simulated annealing (SA), a threshold accepting (TA) and a genetic algorithm (GA) heuristic in this work. They all make use of the result that an optimal solution occurs at one of the extreme points of the feasible region described by the transportation constraints and explained in the next section. According to the computational results reported in Chapter 5 we can say that they are very accurate.

2.2. Continuous Capacitated Multi-Facility Location-Routing Problem

There are review works on the LRP by Laporte [22], Min et al. [23] and Ahıpağaşaoğlu et al. [24]. The survey of Laporte [22] is the pioneer work in the literature. It is the first comprehensive study of the earlier work on deterministic LRP. Min et al. [23] classify the problem according to some criteria such as depot and vehicle numbers, capacity restrictions, time windows, nature of demand etc. They classify previous research according to those criteria. Ahıpağaşaoğlu et al. [24] provide a more extensive survey on the LRP including all problem variants, computational aspects and research prospects. They categorize the problems, give solution techniques and a good bibliography for the researchers .

The papers examined in this section are the works on deterministic LRP which refers the demands of the customers are deterministic. The LRP is a composition of two subproblems,i.e., the location-allocation problem and the vehicle routing problem, both of which are difficult optimization problems. As these problems cannot be solved on a deterministic sequential machine in an amount of time that is polynomial in the size of the input, it is hard to solve the whole problem by integer programming and researchers propose mostly heuristic algorithms for LRP. While some construction and improvement heuristics specialized to vehicle routing and location problems were offered, there are also meta-heuristics such as simulated annealing, tabu-search and neural networks. Some subproblems may be solved by exact methods. Perl and Daskin [26] propose a three indexed zero-one integer programming model and solves the location-allocation subproblem by implicit enumaration algorithm.

Perl and Daskin [26] formulate an integer programming model for the LRP. They offer a heuristic which decomposes the LRP into three subproblems and solving them in sequential manner. The first phase finds an initial routing solution while all potential depots are opened. The second phase solves the location-allocation subproblem with exact methods and finally the third phase solves the multi-depot routing allocation problem for the results obtained from phase two. The last two phases are iterated until the termination criterion is satisfied. Hansen et al.[29] offer a heuristic method almost identical to Perl and Daskin [26]. They divide the problem into three parts as is done by Perl and Daskin [26]. They propose improvement heuristics for the subproblems.

Laporte et al. [25] describe exact branch and bound algorithms for solving the symmetric and asymmetric version of the MDVRP. By using a suitable graph representation, and then a graph extension, the problems are transformed into equivalent constrained assignment problems. Optimal solutions are then found by means of a branch and bound tree.

Jacobsen and Madsen [27] introduce three heuristics which are Tree-Tour Heuristic, ALA-SAV and SAV-DROP. ALA-SAV uses Cooper's ALA heuristic for allocation and Clarke and Wright's savings heuristic for routing. SAV-DROP combines Clarke

and Wright's [28] savings method with Feldman et al.'s DROP heuristic.

Srivastava and Benton [30] investigate the factors that might affect a distribution system design. They also propose three heuristics for the solution of the problem. Later, Srivastava [31] explains those heuristics in detail. According to the first heuristic SAV1, all depots are opened initially. Then, they are dropped and the customers are assigned to the routes departing from open depots. A savings algorithm is used for the approximation of the routing cost. In contrast to SAV1, the SAV2 heuristic assumes that all depots are closed initially and opened one by one. The third heuristic, CLUST, clusters customers using a density search cluster technique and then refines the clusters further by exchanging customers from their initial clusters if their distance to an open depot is reduced.

The evaluation of the vehicle routes and distances is the most time consuming process in LRP. In order to overcome this problem Chien [32] evaluates feasible location-allocation schemes and uses an approximation method to calculate the route lengths in order to solve discrete uncapacitated location, capacitated routing problem. They propose two estimators which provide accurate approximations to the routing problem instead of performing the routing for every allocation.

Tuzun and Burke [33] use a two-phase tabu-search method for solving LRP. They propose a tabu-search algorithm for routing and location subproblems in a sequence in order to solve discrete multi-depot, multi-vehicle problem. They use add and swap moves for location phase, insert and swap moves and for routing phase. Each time a move is performed on the location phase, the routing phase is started in order to update the routing according to the new depot location configuration.

Wu et al. [34] propose a heuristic method dividing LRP into LAP and VRP. For each subproblem, they offer to treat simulated annealing and some tabu list violation modules. In LAP procedure they improve the results by depot swap moves, customer insertion and swap moves. In VRP they offer some vehicle routing improvement heuristics such as two-opt [35], insertion and swap moves. A tabu list is prepared according

to the moves and the improved results are accepted in Simulated Annealing module.

Albareda-Sambola et al. [36] present a multi-depot LRP problem with one single vehicle associated with each open depot. They apply an LP solution to their model to obtain a first lower bound and a starting solution. Then they use Tabu Search heuristic to improve their results. TS method they use includes a series of intensification phase in which they change the customer allocations by different neighborhood structures, and diversification phase in which they change the depot locations by different neighborhood structures.

Schwardt and Dethloff [37] solve the single depot continuous location-routing problem by using self organizing maps. It is the only work that considers a continuous LRP instance in the literature. Their approach will be discussed in detail in the next sections.

3. SOLVING THE CAPACITATED MULTI-FACILITY WEBER PROBLEM

The mathematical programming formulation of the CMFWP can be stated as follows.

CMFWP:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} w_{ij} d\left(\mathbf{x}_i, \mathbf{a}_j\right) \tag{3.1}$$

s.t.

$$\sum_{i=1}^{m} w_{ij} = q_j \quad j = 1, ..., n \tag{3.2}$$

$$\sum_{j=1}^{n} w_{ij} = s_i \quad i = 1, ..., m \tag{3.3}$$

$$w_{ij} \geq 0 \qquad i = 1, ..., m; j = 1, ..., n \tag{3.4}$$

Here, n is the number of customers and m is the number of facilities to be located. q_j and $\mathbf{a}_j = (a_{j1}, a_{j2})^T$ represent, respectively, the demand and coordinates of customer j. The capacity of facility i is given by s_i and $\mathbf{x}_i = (x_{i1}, x_{i2})^T$ denotes its known coordinates. The allocations w_{ij} are also unknown and represent the amount to be shipped from facility i to customer j with the unit shipment cost per unit distance being c_{ij}. This formulation assumes that the problem is balanced, i.e., the total supply is equal to the total demand. If the total supply is larger than the total demand, the problem can be balanced by adding a dummy customer with zero unit shipment cost. In case the total supply is less than the total demand, there exists no feasible solution. From this formulation it is clear that the demand of a customer can be satisfied from different facilities. In other words, the CMFWP is a multi-source problem. If, due to some additional considerations, every customer should be served by a single facility, the problem is formulated as a single-source CMFWP, which requires the use of additional binary variables and constraints.

Note that when allocations w_{ij} are known, the CMFWP reduces to a pure location problem that is separable into m single-facility location problems, each of which can be solved by Weiszfeld's algorithm [9] and its generalizations (e.g. [38], [12], [11]) On the other hand, when the locations of the facilities are given, the CMFWP becomes the ordinary transportation problem. As a consequence, an optimal solution of the CMFWP always occurs at an extreme point of the transportation polyhedron (3.2)–(3.4), independent of the type of the distance function used. This characteristic of the CMFWP was shown by Cooper [7]. Although pure location and transportation subproblems are easy to solve, the CMFWP belongs to a difficult class of problems. Sherali and Nordai have shown that the CMFWP with the Euclidean distance is NP-hard even if all the customers are located on a straight-line [39].

In this work we propose some heuristic methods based on Cooper's heuristic [7]. We move among the extreme points of the transportation problem and then we obtain the objective value of the problem by solving the single facility location problem for the flows (w_{ij}) of the present state. The single facility location problem can be solved by different ways for the rectilinear, Euclidean, squared Euclidean and ℓ_p distance functions.

In this chapter, we will first discuss the techniques used for solving the single facility location problem and then we will work on the heuristic techniques in order to search the extreme points efficiently.

3.1. Single Facility Location Problems

3.1.1. The Rectilinear Distance Single Facility Location Problem

Rectilinear distance location problem aims to minimize the total rectilinear distance between the demand points and the facility. Single facility rectilinear distance

problem can be stated as:

$$\min f(x_1, x_2) = \sum_{j=1}^{n} w_j \left[|x_1 - a_{j1}| + |x_2 - a_{j2}|\right] \tag{3.5}$$

where w_j is a non-negative parameter representing the flow between the supplier and the customer j, (x_1, x_2) is the coordinate of the supplier and (a_{j1}, a_{j2}) is the coordinate of customer j. Love et al. [40] and Francis et al. [5] describe the solution method for this problem. They decompose the problem $f(x_1, x_2)$ into two subproblems $f_1(x_1)$ and $f_2(x_2)$ which are given as:

$$\min f_1(x_1) = \sum_{j=1}^{n} w_j |x_1 - a_{j1}| \tag{3.6}$$

$$\min f_2(x_2) = \sum_{j=1}^{n} w_j |x_2 - a_{j2}| \tag{3.7}$$

These problems are identical to each other, hence the same approach is used for their solution.

Let the values of a_{j1} for j = 1,...,n be reordered to produce $a_{(1)1} \leq a_{(2)1}..... \leq a_{(n_k)1}$ where $n_k \leq n$ and let $w_{(1)}, w_{(2)}, ..., w_{(n_k)}$ be the corresponding weights. Then objective $f_1(x_1)$ becomes:

$$f_1(x_1) = \sum_{j=1}^{n_k} w_{(j)} \left|x_1 - a_{(j)1}\right| \tag{3.8}$$

The derivative of (3.8) can be taken as follows:

$$f_1'(x_1) = -\sum_{j=1}^{n_k} w_{(j)} \qquad \text{for } x_1 < a_{(1)1} \tag{3.9}$$

$$f_1'(x_1) = \sum_{j=1}^{t} w_{(j)} - \sum_{j=t+1}^{n_k} w_{(j)} \qquad \text{for } a_{(t)1} < x_1 < a_{(t+1)1} \tag{3.10}$$

$$f_1'(x_1) = \sum_{j=1}^{n_k} w_{(j)} \qquad \text{for } x_1 > a_{(n_k)1} \tag{3.11}$$

The minimum for $f_1(x_1)$ exists when the derivative equals to zero or the sign of the derivative is changed. Hence, the conditions for $f_1(x_1)$ to be minimum are:

$$\sum_{j=1}^{t-1} w_{(j)} - \sum_{j=t}^{n_k} w_{(j)} \;<\; 0 \tag{3.12}$$

$$\sum_{j=1}^{t} w_{(j)} - \sum_{j=t+1}^{n_k} w_{(j)} \;\geq\; 0 \tag{3.13}$$

If equation (3.13) is satisfied as equality, then there exist more than one optimum in the interval $a_1^* \in \left[a_{(t^*)1}, a_{(t^*+1)1}\right]$. For convenience equations (3.12) and (3.13) can be rewritten as:

$$-C + 2\sum_{j=1}^{t-1} w_{(j)} \;<\; 0 \tag{3.14}$$

$$-C + 2\sum_{j=1}^{t} w_{(j)} \;\geq\; 0 \tag{3.15}$$

where $C = \sum_{j=1}^{n_k} w_{(j)}$. Thus in order to find the optimum coordinate for the problem defined in (3.5), the coordinates of customers are ordered and the value of C is calculated. Starting from the customer having smallest coordinate, twice the weight of each customer is added to $-C$ until the sum equals to or exceeds zero. This procedure is also applied for the x_2-coordinate.

3.1.2. The Squared Euclidean Distance Single Facility Location Problem

In the Squared Euclidean problem, which is also known as the centroid problem, the objective is minimizing the square of Euclidean distance between the facility and the customer. Single facility squared Euclidean problem can be mathematically formulated as

$$\min\ f(x_1, x_2) = \sum_{j=1}^{n} w_j \left[(x_1 - a_{j1})^2 + (x_2 - a_{j2})^2\right]$$

Similar to the rectilinear case, squared Euclidean problem can be divided into two subproblems

$$\min\ f_1(x_1) = \sum_{j=1}^{n} w_j \left(x_1 - a_{j1}\right)^2 \tag{3.16}$$

$$\min\ f_2(x_2) = \sum_{j=1}^{n} w_j \left(x_1 - a_{j2}\right)^2 \tag{3.17}$$

We will again work on the first subproblem $f_1(x_1)$. The solution procedure is obtained by taking the derivative of this problem as is stated by Francis et al. [5]. The first derivative of $f_1(x_1)$ is

$$f_1^{'}(x_1) = 2\sum_{j=1}^{n} w_j(x_1 - a_{j1}) \tag{3.18}$$

By solving $f_1^{'}(x_1) = 0$ we obtain

$$x_1 = \frac{\sum_{j=1}^{n} w_j a_{j1}}{\sum_{j=1}^{n} w_j}.$$

The optimal value for the x_2-coordinate can be obtained as

$$x_2 = \frac{\sum_{j=1}^{n} w_j a_{j2}}{\sum_{j=1}^{n} w_j}.$$

3.1.3. The ℓ_p Distance Single Facility Location Problem

The solution methods for these problems are iterative and they are not easy to solve. The problem can be stated as

$$\min\ f(x_1, x_2) = \sum_{j=1}^{n} w_j \left[|x_1 - a_{j1}|^p + |x_2 - a_{j2}|^p\right]^{1/p}. \tag{3.19}$$

Taking the derivative of the problem with respect to x_1 and equalizing to zero gives

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = \sum_{j=1}^{n} w_j \left[|x_1 - a_{j1}|^p + |x_2 - a_{j2}|^p\right]^{(1-p)/p} |x_1 - a_{j1}|^{p-1} \tag{3.20}$$

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = \sum_{j=1}^{n} \frac{w_j |x_1 - a_{j1}| |x_1 - a_{j1}|^{p-2}}{\left(\left[|x_1 - a_{j1}|^p + |x_2 - a_{j2}|^p\right]^{1/p}\right)^{p-1}} = 0 \tag{3.21}$$

Leaving x_1 alone on the left side results in

$$x_1 = \frac{\sum_{j=1}^{n} \frac{w_j a_{j1} |x_1 - a_{j1}|^{p-2}}{\left(\left[|x_1 - a_{j1}|^p + |x_2 - a_{j2}|^p\right]^{1/p}\right)^{p-1}}}{\sum_{j=1}^{n} \frac{w_j |x_1 - a_{j1}|^{p-2}}{\left(\left[|x_1 - a_{j1}|^p + |x_2 - a_{j2}|^p\right]^{1/p}\right)^{p-1}}}. \tag{3.22}$$

Since x_1 is not isolated on the left hand side, an iterative procedure is proposed in order to find the minimum value. The iterative formulation for x_1 and x_2 coordinates can be formulated as

$$x_1^{new} = \frac{\sum_{j=1}^{n} \frac{w_j a_{j1} \left| x_1^{old} - a_{j1} \right|^{p-2}}{\ell_p \left[(x_1^{old}, x_2^{old}), (a_{j1}, a_{j2}) \right]^{p-1}}}{\sum_{j=1}^{n} \frac{w_j \left| x_1^{old} - a_{j1} \right|^{p-2}}{\ell_p \left[(x_1^{old}, x_2^{old}), (a_{j1}, a_{j2}) \right]^{p-1}}}, \tag{3.23}$$

and

$$x_2^{new} = \frac{\sum_{j=1}^{n} \frac{w_j a_{j2} \left| x_2^{old} - a_{j2} \right|^{p-2}}{\ell_p \left[(x_1^{old}, x_2^{old}), (a_{j1}, a_{j2}) \right]^{p-1}}}{\sum_{j=1}^{n} \frac{w_j \left| x_2^{old} - a_{j2} \right|^{p-2}}{\ell_p \left[(x_1^{old}, x_2^{old}), (a_{j1}, a_{j2}) \right]^{p-1}}}. \tag{3.24}$$

The iteration is stopped when the difference between successive values of x_1 and x_2 coordinates is less than a predetermined value. Francis et al. [5] claim that any initial position can be acceptable for the procedure, but beginning with the solution of the squared Euclidean distance version is faster for convergence. A perturbation can be applied to the distance function in order to avoid the problem of division by zero when the facility location is equal to a customer location. Love et al. [40] state that for $1 \leq p \leq 2$ the convergence is guaranteed when the perturbation factor is positive.

3.1.4. The Euclidean Distance Single Facility Location Problem

Euclidean distance is a special case of the ℓ_p distance function: p is set to 2 in the ℓ_p distance. The iterative formulation for the Euclidean distance single facility location

problem proposed by Weiszfeld [9] for the first time is as follows:

$$x_1^{new} = \frac{\sum_{j=1}^{n} \frac{w_j a_{j1}}{\ell_2\left[(x_1^{old}, x_2^{old}), (a_{j1}, a_{j2})\right]}}{\sum_{j=1}^{n} \frac{w_j}{\ell_2\left[(x_1^{old}, x_2^{old}), (a_{j1}, a_{j2})\right]}}, \tag{3.25}$$

$$x_2^{new} = \frac{\sum_{j=1}^{n} \frac{w_j a_{j2}}{\ell_2\left[(x_1^{old}, x_2^{old}), (a_{j1}, a_{j2})\right]}}{\sum_{j=1}^{n} \frac{w_j}{\ell_2\left[(x_1^{old}, x_2^{old}), (a_{j1}, a_{j2})\right]}}. \tag{3.26}$$

3.2. Simulated Annealing Heuristic

Simulated Annealing (SA) is usually said to be the oldest among the metaheuristics and definitely one of the first algorithms that has an explicit strategy to keep away from local minima. SA is based on the idea of annealing of solids, the cooling of material in a heat bath. The solid material is heated past its melting point and then cooled. As the properties of the solid depend on the cooling rate, if the liquid is cooled slow enough then large crystals can be grown. Hence, in the cooling process the temperature can be lowered step by step until a steady frozen state. This form of cooling control was offered by Metropolis's work [41] in statistical thermodynamics. The probability of an increase in energy of magnitude δE at temperature t is

$$p(\delta E) = \exp(-\delta E/kt) \tag{3.27}$$

where k is a physical constant known as Boltzmann's constant

If energy decreases, the system always moves to the new state. In addition, if the energy is increased, the system moves to the new state with probability given in equation (3.27). The process is repeated for a predetermined number of iterations at

each temperature, after which the temperature is decreased until the system freezes into a steady state.

Based on this statistical mechanics approaches, SA algorithm first presented as a search algorithm for combinatorial optimization problems by Kirkpatrick et al. [42]. He mapped the components of the physical annealing process onto the components of the combinatorial optimization problems as follows:

Annealing Process	**Combinatorial Optimization**
System states	Feasible solutions
Energy	Cost
Change of state	Neighboring solution
Temperature	Control parameter
Frozen state	Heuristic solution

Thus, any local optimization problem can be considered as an annealing algorithm if discrete states of the problem and neighborhood of the solutions are defined. Actually in most optimization problems there exist much more local low energy states than an annealing process.

SA can be considered as a form of the heuristic technique called local neighborhood search, hill climbing or iterative improvement. In the iterative improvement method, feasible solutions are searched in the neighborhood of the initial solution. For an optimization problem, the new solution is accepted only if an improvement occurs. However this approach may get trapped in local minima (see Figure **??**). The quality of the solution is highly dependent on the starting point. On the other hand, SA strategy proposes an algorithm which allows diversification (exploration in large search space) together with intensification (concentrating the search on a confined small search space).

Given a combinatorial optimization problem with a finite set of solutions and an objective function, the SA algorithm is characterized by a rule to randomly generate a

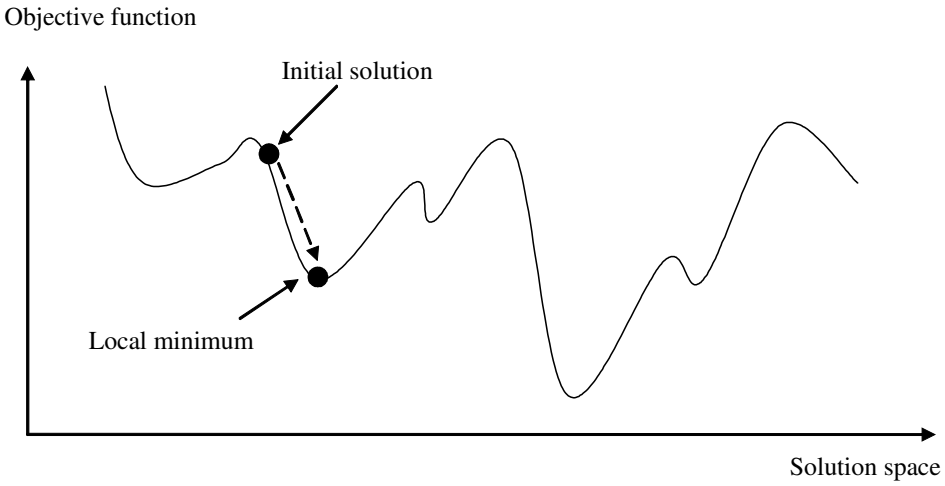

Figure 3.1. Iterative Improvement

new solution in the neighborhood of the current solution. The new solution is accepted if there is an improvement in the objective value. In order to escape the local minima, new solutions with worse objective values are also accepted with a certain probability that depends both on the magnitude of the deterioration Δ and temperature T. The acceptance probability is taken as $e^{-\Delta/T}$ for certain number of iterations L, the temperature T is left unchanged and the temperature is reduced every L iterations by the cooling rate. Thus at the beginning of the search, the probability of admitting uphill moves is high and it slowly decreases, converging to a simple iterative improvement algorithm. Figure 3.2 shows a simple demonstration of SA algorithm. When the algorithm tends to converge to local minimum points, some uphill moves allows the solution to escape from the local minimum.

Considering the search process, the algorithm is the combination of two strategies: random walk and iterative improvement. In the first phase of the search, the bias toward improvements is low and it allows the exploration of the search space; this erratic element is slowly decreased thus leading the search to converge to a (local) minimum. The probability of accepting uphill moves is controlled by two factors: the difference of the objective functions and the temperature. At fixed temperature, the higher is the difference $f(S') - f(S)$, the lower is the probability to accept a move from S to S'. On the other hand, the probability of uphill moves is higher for larger values

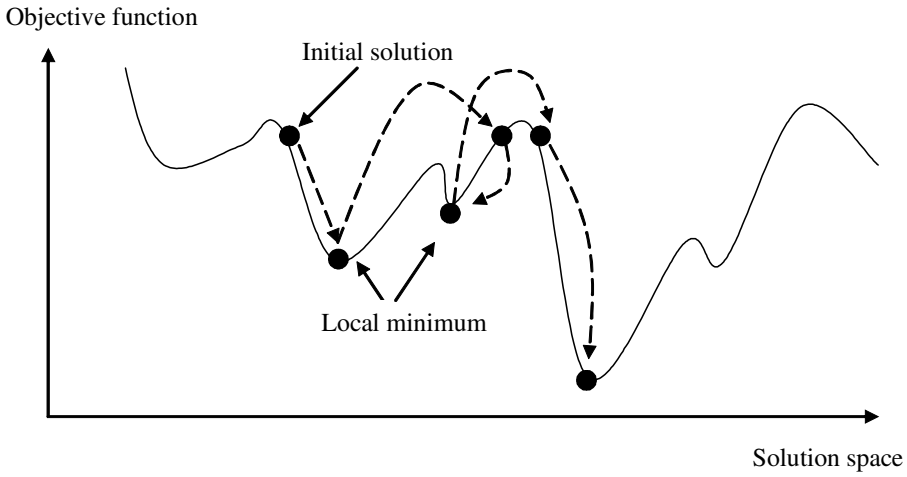

Figure 3.2. Simulated Annealing

of T. Simulated annealing procedure we apply for the CMFWP is shown in Algorithm 1.

It has been shown that the optimal solution occurs at an extreme point of the convex set of the feasible solutions of Capacitated Weber Problem constraint set. Hence, as we perform the search in the space of the extreme points, we should be able to move from one extreme point to another one, which determines the neighborhood structure. It can be easily seen that the constraints of the CMFWP are the same as those of the well-known transportation problem. An important result in linear programming is that each extreme point can be characterized by $m + n - 1$ basic variables and $mn-(m+n-1)$ nonbasic variables if the number of constraints is $m+n$. As a matter of fact, each iteration of the transportation simplex method involves moving from the current extreme point to an adjacent one which results in the largest improvement of the objective value. This is achieved by determining an entering variable among the nonbasic variables and a leaving variable among the basic variables.

Initialization: In order to find a random initial solution we look for a basic feasible solution to a transportation problem whose costs are not yet known. The easiest way of obtaining an extreme point in the transportation problem solution space is applying the Northwest Corner Rule [43]. This constructs a feasible shipment for the transportation

Algorithm 1: *Simulated Annealing for the CMFWP*

1. *Find an initial solution* S *randomly and calculate objective function* $f(S)$.
   Set $S_{min} = S$, $f(S_{min}) = f(S)$. *where* S_{min} *is minimum solution found so far.*
   Assign a value for the initial temperature T_0, *Set* $T = T_0$, *a cooling rate* α
   and iteration length L *at constant temperature.* $Iter = 0$
2. *Select a solution* S' *in the neighborhood of* S.
   Calculate the objective function $f(S')$.
3. *Check whether the new solution is accepted*
   $\Delta = f(S') - f(S)$
   If $\Delta \leq 0$ *then* $S = S'$ *else* $S = S'$ *with probability* $e^{-\Delta/T}$
   If $f(S') < f(S_{min})$ *then* $S_{min} = S'$
   $Iter = Iter + 1$
4. *If* $Iter \geq L$ *then* $T = \alpha T$ *and* $Iter = 0$ *else Goto 2*
5. *If stopping criterion is reached, then stop the algorithm*
   else GoTo 2.

problem, using the cost table as follows. Start at the NW corner (source 1, destination 1), and allocate the most possible: minimum of the supply and demand. If supply exceeds the demand, proceed to the next destination, and continue until all of supply 1 is allocated. Then, go to source 2 and repeat the allocation process, starting with the first (lowest index) destination whose demand has not been fulfilled. If demand exceeds the supply, proceed to the next source, and continue until all of demand 1 is allocated. Then, go to destination 2 and repeat the allocation process. Eventually, all demand must be fulfilled.

After finding the basic vectors for the transportation problem, the costs are evaluated by solving single facility location problem for each facility. The objective value found is taken to be the best solution, basis and flows are taken as best basis and flows.

We set the initial temperature value proportional to the size of the problem. In order to do this, we first find n random pairs of extreme points and after solving the location problem we calculate objective values of these basic vectors found. We find

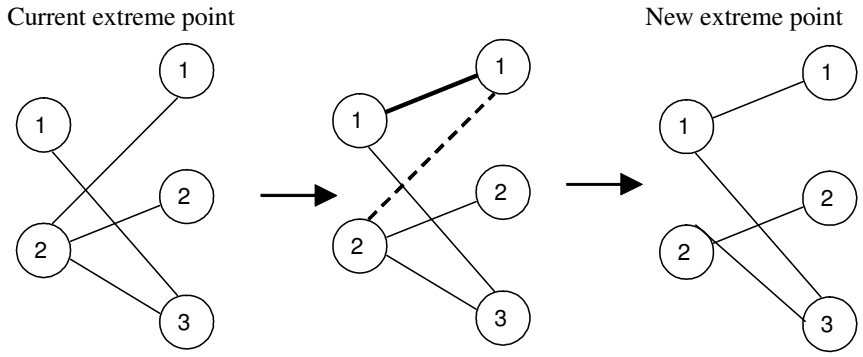

Figure 3.3. SA one-variable exchange

the first basic feasible solution by Northwest Corner Rule. Then we apply one-variable exchange $mn - (m + n - 1)$ times in order to find new random basic vectors. We evaluate the differences of the basic vector pairs. A value $\left|\bar{\Delta}\right|$ is set by taking the average of the differences of the pairs' objective values. The value of T_0 is computed by setting the probability p of accepting an average bad move early in the algorithm. We compute T_0 according to

$$p = e^{-|\bar{\Delta}|/T_0} \tag{3.28}$$

which gives

$$T_0 = -\frac{|\bar{\Delta}|}{\ln(p)} \tag{3.29}$$

Neighborhood Structure: In the SA-based heuristic we use two different neighborhood structures: One-variable exchange and two-variable exchange. Given the basic variables associated with the current solution, we select randomly one of the nonbasic variables and select it as the entering variable. The leaving variable is determined by the stepping stone method. The size of the neighborhood for one-variable exchange is $NS_1 = mn - (m + n - 1)$. Figure 3.3 illustrates the network representation of one-variable exchange. The edges between the nodes represent the existence flow between the depots and customers.

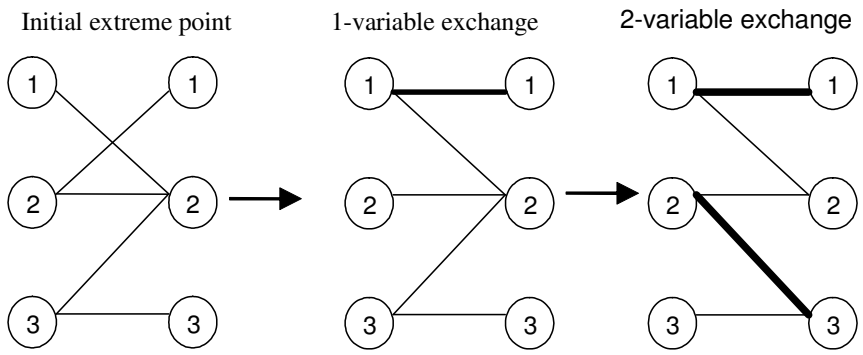

Figure 3.4. SA two-variable exchange

The two-variable exchange is the application of the one-variable exchange twice. Hence, it corresponds to moving from the current extreme point to a nonadjacent extreme point by performing one-variable exchange twice. For two-variable exchange, the neighborhood size can be formulated as $NS_2 = \begin{pmatrix} NS_1 \\ 2 \end{pmatrix}$. An example of two-variable exchange is demonstrated in Figure 3.4. In the example, flow (1,1) is inserted to the basis and corresponding flow (2,1) is deleted from the basis, subsequently (2,3) is inserted and (3,2) is deleted.

New Solution Generation and Acceptance Rule: In order to generate new solutions, we choose a random point from the set of neighborhood we have selected. As the SA heuristic performs the search in the space of flow variables, for each extreme point we have to solve m single-facility location problems in order to compute the objective value. In case of the SECMFWP each location subproblem can be solved analytically by computing the flow-weighted centroid of the customer locations. For the RCMFWP we use the median location method. The ECMFWP is solved via the Weiszfeld procedure [9], while for LpCMFWP we apply the generalizations of the Weiszfeld procedure [5]. If the objective value of the neighbor solution is better than the old solution, we accept the neighbor as our new solution. In other case we accept the neighbor with probability $e^{-\Delta/T}$ where $\Delta = f(S') - f(S)$. If we accept the neighbor as our new basis we update the basis vectors, flows and objective values with the new solution. If the new solution is the best solution from the beginning of the algorithm, we update the best objective value, basis and flows.

Parameter Adjustments: We adjust the temperature when a predefined number of iterations L is reached. The number of iterations L, performed at temperature T is defined as $L = r * NS$ where r is the length coefficient parameter that should be chosen according to the behavior of the problem. NS is taken NS_1 and NS_2 for one-variable and two-variable exchange, respectively. We repeat the edge movement, at a constant temperature for L times. Then we decrease the temperature value geometrically with the formulation: $T_{new} = \alpha * T_{old}$. Increasing the cooling rate causes the temperature decrease slower, so the algorithm searches more space before the stopping conditions are satisfied.

Termination criterion: If the ratio of accepted moves is less than $\%E$ of the total moves at a constant temperature T for K consecutive series of temperatures, we stop the algorithm. We take $K = 5$ and we perform different values for E in order to see the effect of changes in the termination criterion to the final solution. The best objective value, basis and flows found so far are given as the final solution found by the algorithm.

3.3. Threshold Accepting Heuristic

Threshold Accepting TA is a local search algorithm similar to SA. It is first introduced by Dueck and Scheuer [44]. Instead of the probabilistic element used in SA, a deterministic threshold is used for the acceptance of the worse solutions. As in SA, for every new solution, TA finds out the difference between the objective value of new solution found and the previous solution accepted. The main difference between SA and TA is the acceptance rule applied. TA admits every new solution if it is not much worse than the previous one. However, SA may admit worse solutions with some small probability. TA is claimed to be simpler and faster than SA, because it does not have to make random decisions and compute probabilities. The TA algorithm applied for the CMFWP is illustrated in Algorithm 2.

The one-variable exchange and two-variable exchange neighborhood structure are also applied in TA. All the other procedures in TA: initialization and update of

Algorithm 2: *Threshold Accepting for the CMFWP*

1. *Find an initial solution* S *randomly and calculate objective function* $f(S)$.
   Set $S_{min} = S$, $f(S_{min}) = f(S)$. *where* S_{min} *is minimum solution found so far.*
   Assign a value for the initial threshold T_{h0}, *Set* $T_h = T_{h0}$, *a cooling rate* α
   and iteration length L *at constant temperature.* $Iter = 0$
2. *Select a solution* S' *in the neighborhood of* S.
   Calculate the objective function $f(S')$.
3. *Check whether the new solution is accepted*
   $\Delta = f(S') - f(S)$
   If $\Delta \leq T_h f(S)$ *then* $S = S'$
   If $f(S') < f(S_{min})$ *then* $S_{min} = S'$
   $Iter = Iter + 1$
4. *If* $Iter \geq L$ *then* $T_h = \alpha T_h$ *and* $Iter = 0$ *else GoTo 2*
5. *If stopping criterion is reached, then stop the algorithm else GoTo 2*

parameters, coefficients (α, r, T, L, K, E) and the stopping criterion is applied similar to SA algorithm proposed.

3.4. Genetic Algorithm Heuristic

Genetic Algorithm (GA) is a metaheuristic search algorithm inspired from the selection mechanism in nature. It is based on the rule "the stronger individuals survive while the weak ones die and vanish". GA was initially treated systematically and fully by Holland [45]. The name GA originates from the similarity of the complex structure of a component vector and the genetic structure of a chromosome. In GA each chromosome encodes a solution to the problem and fitness value of the chromosome associates with the objective function value. It differs from other algorithms with working in a population of solutions rather than a single solution. Information is carried by the whole population to the next generation.

GA generates an initial feasible solution set which forms the population. New individuals are created by combining the parent individuals or changing the initial pop-

ulation or changing the decoding of the parent. Only the ones which have the fittest values among parents and offsprings are able to survive in the next generation. In optimization algorithms the fittest solution represents the strongest individual. Each chromosome (individual) represents a solution to the problem and the fitness is determined by computing the objective function value for this solution. In order to produce new better solutions we will use the most common genetic operators: crossover and mutation. Encoding of the solutions may be done by bit strings, integer/real arrays or more complex tree structures.

Crossover is the exchange of genes of chromosomes. Crossover operator matches two chromosomes and then genes of the chromosomes are combined forming the child. In Figure 3.5 we give a simple representation of crossover proposed in Holland's original GA. The string encoding is used in the example. One index point is selected randomly dividing the chromosomes into two parts. Then the pieces are exchanged in order to produce the offsprings.

Par1	1	0	1		0	0	1	0		Off1	1	0	1	1	0	0	1
				X													
Par2	0	1	1		1	0	0	1		Off2	0	1	1	0	0	1	0

Figure 3.5. One point exchange method for crossover operator

Mutation is a random change in the genes of chromosome. It may be performed to the child obtained from the crossover or to the parents directly. It increases the diversification of the algorithm. Two simple types of mutation are given in Figure 3.6: inversion and insertion. In the inversion method two points are selected randomly on the chromosome and then the gene string between these points are inverted. In the latter method, a gene is selected randomly and then inserted to a new position on the chromosome.

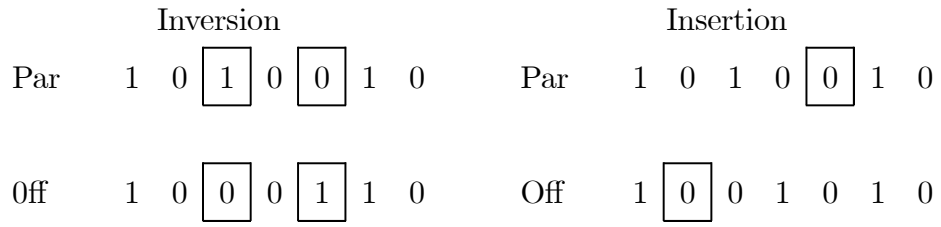

Figure 3.6. Inversion and insertion methods for mutation operator

GAs were proposed in the literature to solve the fixed charge transportation problem (FCTP) which considers not only the linear costs as in the case of the classical transportation problem, but also fixed costs [46], [47]. Note that the FCTP has the same constraints as the CMFWP does. In the literature, different representations were applied to the FCTP such as the Prüfer number representation, matrix representation, permutation representation and direct or edge-based representation [46], [47].

For the Prüfer number and matrix representations, offsrings are not always basic feasible solutions for the FCTP. The offsprings produced by the algorithms based on matrix representation and permutation representation do not show the main characteristics of the parents and do not give better results than generating a new solution randomly. In addition Goetlieb et al. [48] show that Prüfer numbers are very poor representations of spanning trees for Genetic Algorithms because of the low locality and low heritability coding. A representation has high locality if mutating a genotype changes the corresponding phenotype only slightly. Similarly a representation has high heritability, with respect to a crossover operator, if offspring phenotypes consist mostly of substructures of their parents' phenotypes.

Eckert and Gottlieb [46] show that the edge-based representation outperforms the others. The offsprings created in edge based algorithms are always basic feasible solutions. In contrast to other representations, edge based representation has high locality and heritability. These properties bring us to perform edge based representation for Genetic Algorithm approach to the CMFWP.

The starting point of the edge-based representation is that an extreme point with $m+n-1$ basic variables corresponds to a spanning tree with $m+n-1$ edges in the transportation graph. This means that each extreme point can be represented by a chromosome that consists of the set of the edges of the corresponding spanning tree and the associated flow quantities demonstrated in Figure 3.7. The edge set can be written as $E(w) = \{(1,1),(1,2),(2,2),(2,3)\}$.

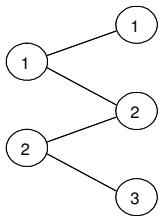

Figure 3.7. Edge Representation for GA

A wide variety of GA procedures exist in the literature that we cannot able to introduce in this work. Population size, selection mechanisms, fitness calculation, chromosome coding and representation, modified types of crossover and mutation operators are the factors that build this variety. In our implementation the fitness function is the objective function which is calculated by solving the location subproblems for each chromosome. The location subproblems are solved exactly by the same methods implemented in SA. At each generation crossover and mutation operators are performed respectively. The GA we propose for the CMFWP is described in Algorithm 3.

Initialization: Initial parents are found with a similar method we have performed in SA. First we determine the population size for the problem, after we find the first parent by performing Northwest Corner Rule. Then we apply one vector exchange $mn-(m+n-1)$ times in order to find new random parent vectors. We do not allow individuals, which has the same basis vector, to take part in the population. For each parent, we calculate the objective function by solving the location subproblem. Parent vectors, flows and fitness functions are stored in the memory in order to represent the parents. Fitness function is the cost calculated in the location subproblem. In our implementation we first apply a crossover and then a mutation operator in each

generation.

Algorithm 3: *Genetic Algorithm*

1. *Define the population size N and generation number G*
   Find initial individuals S_k for $k = 1$ to N randomly.
   Calculate the fitness value $f(S_k)$ for $k = 1$ to N
   Determine the worst individual S_W.Set $Iter = 0$
2. *Obtain temporary offspring by applying crossover*
   operator to two parents selected from the population
3. *Apply mutation to the temporary offspring*
   and get the new individual S_{N+1} and compute $f(S_{N+1})$
4. *Check whether the new solution is accepted*
   If $f(S_{N+1}) < f(S_W)$ and $f(S_{N+1}) \neq f(S_k)$ $k = 1$ to N, then
   the new offspring is inserted into the population
   S_W is deleted. New S_W is determined
   End if
   $Iter = Iter + 1$
5. *If $Iter = G$, then stop the algorithm else GoTo Step 2*

Crossover Operator: We select the parents that will involve into the crossover operator with binary tournament method. While applying this method we do not select a parent twice. The crossover operator is based on the idea that the offspring is formed by the edges of the parents selected and it has to be roughly at the same distance from both parents. Given parents $P1$ and $P2$, about half of the edges of parent $P2$ that do not exist in $P1$ are selected. These edges are then inserted into the chromosome of $P1$ one by one. At each step a selected edge of $P2$ is inserted into $P1$ while one of the edges in $P1$ is deleted by the same process performed for the one-variable exchange in SA. The main advantage of this method is that we always get a basic feasible vector after each move. Crossover operator is illustrated in Figure 3.8. In the example the edge set of P2 differing from P1 is $E(P2) \backslash E(P1) = \{(1,1),(2,3),(3,1)\}$. Edges (1,1) and (2,3) are selected randomly and first the edge (1,1) is inserted from the second parent into the first parent yielding the intermediate offspring. In the second step,

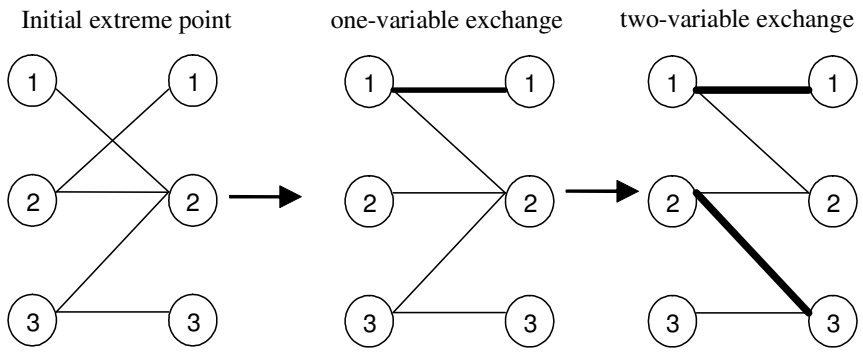

Figure 3.8. GA Crossover Operation

edge (2,3) of second parent is inserted into the first parent yielding the new offspring.

Mutation: We apply mutation operator to the child created by the crossover operator. The mutation operator is defined as follows. A new edge is selected randomly among the edges that do not exist in the chromosome. This creates a unique cycle in the spanning tree as $m+n$ edges in a tree always results in a cycle. In order to find the edge that should be deleted to form a new spanning tree, we apply an approach similar to the stepping stone method. This mutation process is exactly equal to a move of the one-variable exchange neighborhood structure illustrated in Figure 3.3.

Assessing the Child: The location subproblem is solved after the mutation operator and the fitness value is calculated. We use a steady-state replacement scheme where the offspring replaces the worst individual in the population if the following conditions are satisfied. First, the fitness of the offspring should be lower than that of the worst individual. Second, the offspring should not have duplicates in the population. The new parent configuration is carried to the next generation.

Termination: We define generation as performing crossover and mutation operators respectively. Increasing the number of generations may cause an improvement in the objective value however it is expected to increase computation time linearly, so we have to select the generation number considering computation time and problem accuracy. The algorithm is terminated when determined number of generations or computation time is reached.

4. SOLVING THE CONTINUOUS CAPACITATED MULTI-FACILITY LOCATION ROUTING PROBLEM

In this chapter, we will define the LRP and its components in detail. In the LRP, the locations and the demand of n customers are given. m depots have to be located to satisfy the customer demand. The depots can be placed anywhere in the continuous plane. The deliveries from the depots to the customers are performed by a set of vehicles. Vehicles visit the customers on a route beginning and ending at one of the depots. The problem is to minimize the total distance of routes traveled by the vehicles. In our study, we ignore the fixed costs for depots or vehicles for the location-routing problem.

In the context of the LRP, the system cost is equivalent to the transportation cost of the vehicles. The fixed costs of truck or depots, packaging or inventory costs are ignored. The objective function minimizes the total cost of the vehicle tours in terms of the Euclidean distance. Transportation or delivery cost may be divided into two major elements: stem distance and the variable distance. The stem distance is the sum of the distances from the depot to the first customer and from the last customer back to the depot. The stem distance reflects the effect of the location of the depot on the total distance traveled. The variable running distance is the result of the decision made in order to select the next destination of the vehicle after visiting a customer. The variable distance represents the effect of the sequence of customers. Stem distance and variable distance are illustrated in Figure 4.1. Clearly the location of the depot has a secondary effect on variable running distance except changing first and last customers on the route. The mathematical model of our continuous capacitated multi-facility location-routing problem (CCMFLRP) is presented next.

$$Minimize \sum_{i=m+1}^{n+m} \sum_{\substack{j=m+1 \\ j\neq i}}^{n+m} c_{ij} \sum_{k=1}^{p} x_{ijk} \tag{4.1}$$
$$+ \sum_{f=1}^{m} \left[\sum_{j=m+1}^{n+m} \left(c_{fj}\left(l_{fx}, l_{fy}\right) \sum_{k=1}^{p} x_{fjk} \right) + \sum_{i=m+1}^{n+m} \left(c_{if}\left(l_{fx}, l_{fy}\right) \sum_{k=1}^{p} x_{ifk} \right) \right]$$

$$\sum_{j=m+1}^{n+m} q_j y_{jk} \leq cap \quad k = 1, ..., p \tag{4.2}$$

$$\sum_{k=1}^{p} y_{jk} = 1 \quad j = m+1, ..., n+m \tag{4.3}$$

$$\sum_{\substack{g=1 \\ g\neq h}}^{n+m} x_{ghk} = y_{hk} \quad h = 1, ..., n+m;\ k = 1, ..., p \tag{4.4}$$

$$\sum_{\substack{h=1 \\ h\neq g}}^{n+m} x_{ghk} = y_{gk} \quad g = 1, ..., n+m;\ k = 1, ..., p \tag{4.5}$$

$$\sum_{i\in Q} \sum_{j\in Q} x_{ijk} \leq |Q| - 1, \forall Q \subseteq V, 2 \leq |Q| \leq n-1 \quad k = 1, ..., p \tag{4.6}$$

$$\sum_{f=1}^{m} y_{fk} = 1 \quad k = 1, ..., p \tag{4.7}$$

$$\sum_{k=1}^{p} y_{fk} = veh_f \quad f = 1, ..., m \tag{4.8}$$

$$y_{gk} \in \{0, 1\} \quad g = 1, ..., n+m;\ k = 1, ..., p \tag{4.9}$$

$$x_{ghk} \in \{0, 1\} \quad g, h = 1, ..., n+m;\ g \neq h;\ k = 1, ..., p \tag{4.10}$$

Here, n is the number of customers, m is the number of depots to be located and p is the number of vehicles. The indices are i and j for customers, f for depots, g and h for nodes (customers+depots), and finally k for vehicles. q_j represents the demand of customer j. Cost of direct travel from node g to node h is given by c_{gh}. The capacity of vehicles is given by cap. veh_f represents the number of vehicles assigned to depot f. V is the set of customers while Q is a subset of V. If node g precedes node h on route k, binary variable x_{ghk} is set to 1, otherwise it is set to 0. Similarly, if node g is assigned to vehicle k, binary variable y_{gk} is set to 1, otherwise it is set to 0. Variables

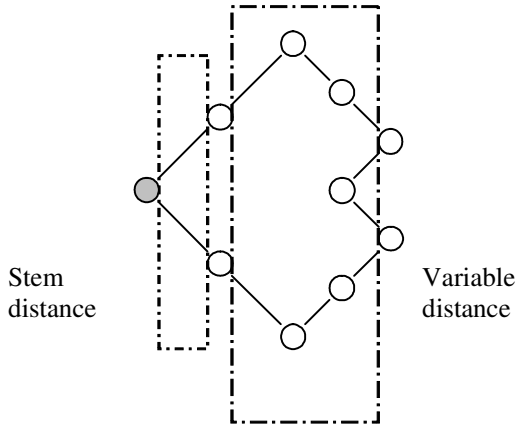

Figure 4.1. Distance types in LRP

l_{fx} and l_{fy} represent, respectively, the x-coordinate and y-coordinate of depot f.

The constraint in equation (4.2) states that the demand of the customer allocated to a single vehicle must not exceed the vehicle capacity, while (4.3) ensures that each customer is assigned to exactly one vehicle. The following two constraints (4.4) and (4.5) are given in order to assign the customers to the vehicle routes in a sequence. In equation (4.4) it is stated that if node h is assigned to vehicle k, vehicle k must visit node h exactly once, else the vehicle cannot visit node h. Similarly equation (4.5) commands that if node g is assigned to vehicle k, vehicle k must leave node g exactly once, else the vehicle cannot leave node g. Equation (4.6) is the subtour elimination constraint for the vehicle routes. Constraints (4.7) and (4.8) organize the relations and assignments between the vehicles and the depots. Constraint (4.7) says that each vehicle is assigned to exactly one depot while (4.8) ensures that each depot is assigned to a predetermined number of vehicles.

In this chapter we will introduce a neural network, referred to as the self organizing map algorithm (SOM) for solving the CCMFLRP. However, first we will give a brief introduction to artificial neural networks in order to understand the SOM structure better.

4.1. Artificial Neural Networks

Artificial Neural Networks (ANN) are computational prototypes that learn from experience instead of being expressly programmed with rules as in conventional artificial intelligence. They are formed by small and simple processing elements linked by weighted connections. These computation units (neurons) compute from local information stored and transmitted via connections. The system learning is obtained by the connectivity among these units and parallel computation is occurred. The major components forming neural networks are discussed in this section.

All of the processings in ANN are carried out by a set of processing units (neurons). A unit's job is simply to receive input from its neighbor's and to compute an output value which it sends to its neighbors. There are three types of units: input units which receive inputs from the external system, output units which send signals out of the system and finally hidden units whose inputs and outputs are within the system. The state of activation a_i represents of the state of the unit u_i at time t. They can be continuous or discrete; bounded or unbounded. Related with each unit there is an output function $f_i(a_i(t))$ which maps the current state of activation $a_i(t)$ to an output signal $o_i(t)$ where $o_i(t) = f_i(a_i(t))$

Pattern of connectivity of the network constitutes what the system knows and determines, how it will respond to any arbitrary input. The total pattern of connectivity can be represented by merely specifying the weights for each of the connections in the system. w_{ij} represents the strength and the sense of the connection from unit u_j to unit u_i. We can represent the pattern of connectivity by a set of connectivity matrices W_i for complex connectivity for each type of connections. We also need a rule which takes the output vector $o(t)$ and combines it with the connectivity matrices to produce a net input for each type input into unit. We can write $S(t) = Wo(t)$ to represent the net input vector for inputs of unit i.

We need a function f which takes $a(t)$ and the vectors $S_i(t)$ producing a new state of activation. In general we have $a(t+1) = f(a(t), S(t))$. Some activation functions used

commonly are threshold, linear ($f(S) = S$), and sigmoid ($F(S) = (1 + e^{-S})^{-1}$) functions. We also need a learning rule. Changing the processing or knowledge structure involves modifying the patterns of interconnectivity. These can be done by development of new connections, the loss of existing connections or the modification of strengths of connections. Learning algorithms in ANN can be divided into two: supervised and unsupervised. In supervised learning, the desired outputs o_d are known priori and are used during training so that neurons can adjust their weights by matching their outputs to the desired outputs. The weights are usually synthesized gradually, and at each step of the learning process they are updated in order to minimize the error between the network's output and a corresponding desired target. After training, the neural network is tested by presenting some input values, and determining how close it approximates to outputting the correct target values. In unsupervised learning, the neural network is not given a target value during training. Unsupervised neural networks usually perform some kind of data compression, such as dimensionality reduction or clustering.

For instance, a single layer, supervised model of neural networks (Perceptron) will be introduced. Perceptron model is based on a neural unit which takes more than one input and produces an output. The output is a Boolean function which is 0 or 1. Threshold Activation function is used to compute the output. Original perceptron convergence procedure is given in Algorithm 4 and illustrated in Figure 4.2.

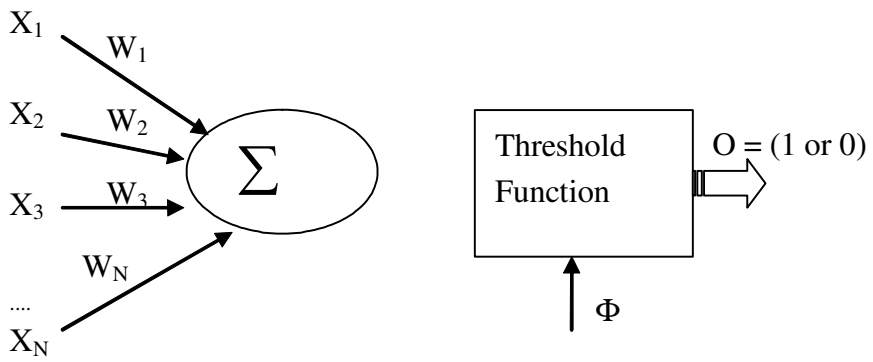

Figure 4.2. Perceptron Model

Artificial neural network architectures are grouped into three classes: feedforward,

Algorithm 4: *Perceptron Model*

1. *Initialize weights and threshold* $w_i(0)$ *and* θ
2. *Present new input and desired output* $X=(x_1, x_2, \ldots, x_n)$ *and* $o_d(t)$
3. *Net input is computed:*
   $S = w_i(t)x_i(t) - \theta$
4. *Actual Output is calculated*
   $o(t) = f(S) : if\ S > 0\ o(t)=1\ else\ o(t) = 0$
5. *Adapt weights*
   $w_i(t + 1) = w_i(t) + \alpha(o_d(t) - o(t))x_i(t)$

feedback and competitive networks. Feedforward Networks canalize sets of input signals into output signals. The desired input output transformation is usually determined by external, supervised adjustment of the system parameters. Each input pattern received from the environment is associated with a specific desired target pattern. Feedforward networks do not have a cyclic connectivity pattern. Also the connectivity pattern may be cyclic in a network which we call feedback networks. These networks may have bidirectional connections, self connections recurrent connections. In feedback networks the input information specifies the initial activity state of a feedback system, after state transitions the final state is obtained as the outcome. In competitive networks cells compete in their activities by means of mutual lateral interactions and develop adaptively into specific detectors of different signal patterns. The aim is to classify data into a number of groups without the aid of training groups. The weights and the outputs are usually expected to converge to representations that capture the statistical regulatories of the input data.

A well known competitive network type called Learning Vector Quantization (LVQ) uses unsupervised learning strategy. If the input vectors are to be classified into a finite number of clusters then several codebook (neuron weight) vectors are usually made to represent each class. The goal is to locate the codebook vectors such that they directly define near optimal class boundaries among the classes. In other words, LVQ attempts to design codebook vectors which can discriminate the classes instead of minimizing an error functional. In the algorithm the inputs are shown sequentially

and the location of the closest neuron to the input is updated. This closest vector, named winner codebook, is moved to the input. After a sufficient number of iterations the codebook vector begins to win for certain input vectors. The codebook vector represents the group of these input vectors.

In order to solve LRP we will use a prototype of competitive neural network structure with lateral excitation: Self Organizing Maps. Unlike the vector quantization, not only the winner but also the neighboring neurons are updated. In other words, the winner and a group of neurons in the neighborhood of the winner are moved towards the current input.

4.2. Self Organizing Maps

A Self Organizing Map (SOM) is a neural network consisting of two connected layers of neurons as shown in Figure 4.3.

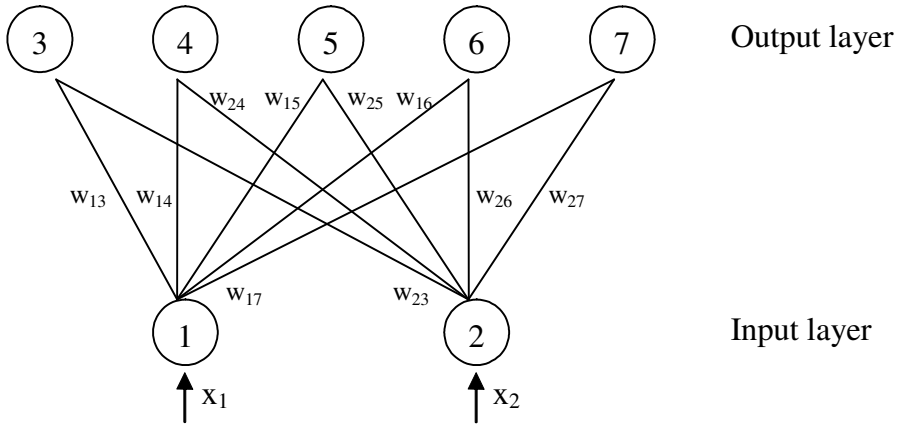

Figure 4.3. SOM

The neurons in the first layer are used to pass the information to the neurons of the second layer. Weights are assigned to the connection between the neurons of the first and second layers. For a given data, the output units compete to be the winner according to a specified criterion. The weight vectors are updated in the neighborhood of the winner in order to get closer to the input data. The training of the network is continued until the winner output unit associated with each input data is not changed

for some number of iterations.

Kohonen [50] experiments with various architectures and system equations which produces globally well organized maps. It is an effective tool for the mapping of complex high dimensional data onto a low dimensional data having simple relations. In contrast to vector quantization (VQ), the codebook vectors do not act independently in SOM. As in VQ, the closest codebook vector becomes the winner; but the update function differs from VQ. A neighborhood set N_c is defined around the winner cell c. At each learning step all the cells within N_c are updated, while cells outside of N_c are unchanged. The width of radius N_c is time variable. It will be advantageous to set N_c very wide in the beginning and then shrink. It is possible to end the process with $N_c = c$. The SOM algorithm used in this research is depicted in Algorithm 5.

In Algorithm 5, $\mathbf{x}_i$ represents the input vector i and N_x is the number of input vectors. *dist* represents the index distance between neurons, while d represents the Euclidean distance between the input vector and weight vector. α and M_α represent, respectively, the learning rate factor and the momentum of learning rate factor. Similarly σ and M_σ represent, respectively, width of neighborhood and momentum of width of neighborhood. $\mathbf{w}_j$ is the weight vector of neuron j, c is the index of the winner weight and vector *iter* is the iteration counter. Λ_{jc} represents the neighborhood function value between the winner node and node j. Φ and Ψ represent, respectively, the set of neurons of the input layer and the second layer.

The input vectors in location problem are the coordinates representing the two dimensions of the demand points. Hence, we need two input neurons in order to represent the dimensions of the demand points and also weight vectors. Weight vectors are also called codebook vectors in the literature. The number of the output neurons determines the number of the codebook vectors which will represent the input data.

$\sigma(t)$, the width of N_c at time t, is initially taken a large value and decreased until the end of the iterations. From the formula we can see that as the codebook gets closer to the winner codebook, $\Lambda_{jc}(t)$ increases to 1. On the other hand, as $\sigma(t)$ decreases to

0, $\Lambda_{jc}(t)$ decreases to 0.

Algorithm 5: *Self Organizing Map*

1. *Initialization*

   Define the width of neighborhood $\sigma(0)$

   Define the learning rate $0 < \alpha(0) < 1$

   Define weight vector $\mathbf{w}_j = (w_{1j}, ..., w_{nj})^T$ *for each neuron* $j \in \Psi$

   The weights of the input neurons $i \in \Phi$ *are set to one*

   Define maximum number of iterations

   Set $Iter = 0$

2. $t = 0, Iter = Iter + 1$
3. $t = t + 1$
4. *Determine a random order of input vector selection*
5. *Select the input vector* $\mathbf{x}_i = (x_{i1}, x_{i2})^T$ *in the sequence*
6. *Compute the distance* d_j *between the input vector* $\mathbf{x}_i$ *and the weight vector* $\mathbf{w}_j(t)$

   $\forall j \in \Psi$ *compute* $d_j(\mathbf{x}_i, \mathbf{w}_j(t))$
7. *Select neuron* j^* *satisfying* $d_j^{\min} = \min\{d_j(\mathbf{x}_i, \mathbf{w}_j(t))\}$ *as the center c(t)*
8. *Adapt the weight vectors* $\forall j \in \Psi$ *using the formula*

   $\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \alpha(t)\Lambda_{jc}(t)(\mathbf{x}_i - \mathbf{w}_j(t))$

   where $\Lambda_{jc}(t) = \exp\left(-\frac{dist_{cj}^2}{\sigma^2(t)}\right)$
9. *Update* α *and* σ; $\alpha(t+1) = \alpha(t)$, $\sigma(t+1) = \sigma(t)$

   If $t = N_x$ *Then GoTo 10 Else GoTo Step 3*
10. *Update* α *and* σ; $\alpha(1) = \alpha(N_x)M_\alpha$, $\sigma(1) = \sigma(N_x)M_\sigma$

    If $Iter = Itermax$ *Then stop the algorithm Else GoTo Step 2*

The adaptation of the codebook vectors is depicted in Figure 4.4. The distances between the codebook vectors and the input vector are computed. In the figure codebook vector 5 is the closest vector to the input vector and is declared as the winner. All codebook vectors are updated and come closer to the input vector according to the neighborhood function with respect to the winner. From the figure it can be observed

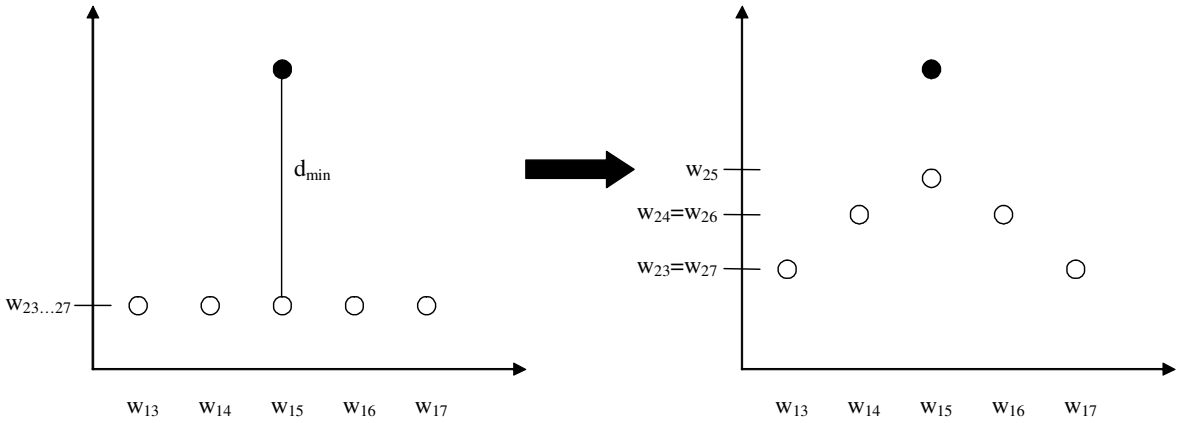

Figure 4.4. Adaptation of codebooks in the neigborhood of the center.

that coordinates of winner codebook vector and the vectors closer to the center are moved to a larger extent.

In SOM, weight vectors approximate the density of the input vectors. The neighborhood of the weight vectors may be organized as a linear array in one dimension or as a grid in two dimensions. If one dimension is considered, as shown in Figure 4.5, the distance is simply $|r_j - r_c|$ where r_j denotes the index vector of codebook vector j. If we take two dimensional map, as shown in Figure 4.6, the Euclidean distance is considered $\|r_j - r_c\|$. Two dimensional maps are suitable to represent a plane or a solid in three dimensional space.

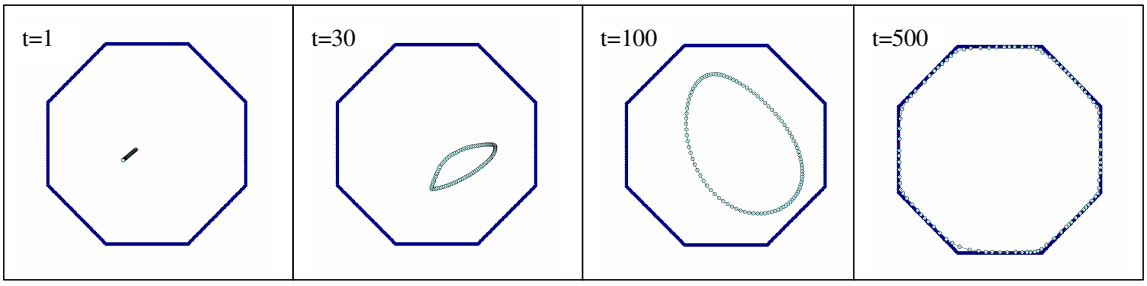

Figure 4.5. One dimensional SOM

One dimensional self organizing maps are mostly considered in routing problems for the reason that the input vectors in routing problems are needed to be arranged on

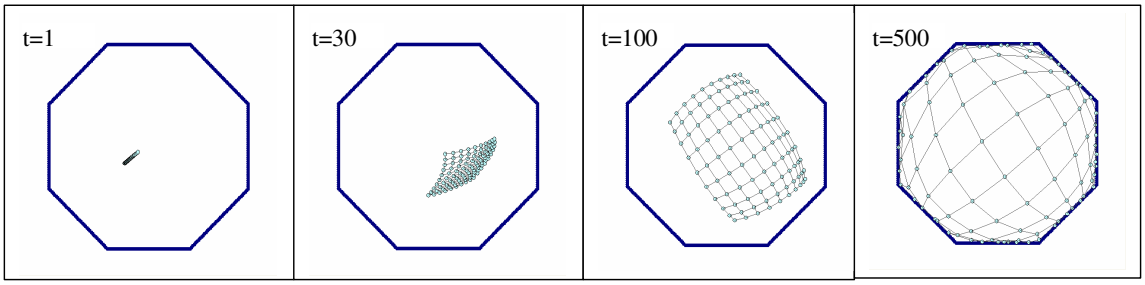

Figure 4.6. Two dimensional SOM

a line. The idea of using Self Organizing Maps for routing problems are inspired from the elastic net approach to TSP proposed by Durbin and Willshaw [49]. Elastic net is a geometrically designed algorithm starting with M neurons lying on an imaginary elastic band originally located at the center of the inputs where M is larger than the number of inputs N. The neurons are then moved in the Euclidean plane so that the elastic band is gradually stretched until it passes sufficiently near each input to define a tour. During that process two forces are applied to the band: one for minimizing the length of the band and the other one for minimizing the distance between the inputs and the neurons on the band. It can be observed that these two forces are applied also in SOM.

Angeniol et al. [51] are among the first authors using SOM on the TSP. Modares et al. [52] use SOM for multiple traveling salesman problem (MTSP). They consider more than one ring which are connected to a central depot. Meanwhile Somhom et al. [53] work on MTSP with a minmax objective. Schwardt and Dethloff [37] showed that SOM can be considered for solving the continuous location-routing problem. On the other hand, they only offer a solution method for single depot case, and do not consider multi-depot LRP. They use several rings of neurons each representing the tour of a vehicle connected to a single codebook which represents the depot. By doing this, movement of the depot in the plane is allowed.

The experiments show that SOM is successful especially with the location problems in continuous space. Also there have been many SOM based researches on location-allocation and routing problems which are the two subproblems of the

location-routing problem. In this research an approach is developed which integrates the location-allocation and vehicle routing subproblems in order to solve location-routing problem.

4.3. Self Organizing Map Algorithm for Continuous Capacitated Multi-Facility Location-Routing Problem

By the success of the SOM algorithm on TSP, the researchers pay their attention on multi vehicle routing problems. However, there are some additional conditions of VRP that cause SOM hard to implement. First, the problem has multi-routes similar to MTSP. Somhom et al. [53] have solved this problem by running the SOM algorithm for each route simultaneously. The routes are represented by one dimensional codebook array similar to the implementation for TSP. The depot is set at the center of the rings. The starting and the end points of the codebook array is linked to the depot but the depot's location is not changed.

Another difficulty is the capacity constraint which is a considerable obstacle for the algorithm's convergence. Capacity constraints hinder the freedom of the codebook movements. Modares et al. [52] propose a mechanism in the winner selection rule not to cause overloaded vehicles. They modify the winner selection criterion by including a bias term which reflects the load rate of each vehicle. This bias term prevents the nodes of the overloaded vehicles to be the winner more frequently and increase the probability of a node of vehicle that has not enough change to be the winner. The winner node is selected according to the following formula

$$c = argmin_j \{|\mathbf{x}_i - \mathbf{w}_j| + vB\}$$

where B is the bias term and is given as $\sum(load/capacity)$. *load* refers to the load of the vehicle and *capacity* refers to the capacity of the vehicle. v is a parameter that determines the effect of the bias to the whole selection algorithm that reconciles between the quality and the feasibility of the solution.

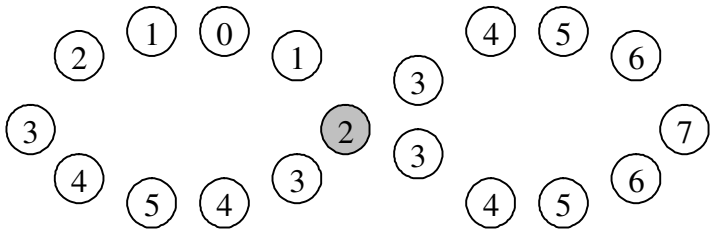

Figure 4.7. Neighborhood of the nodes

Schwardt and Dethloff [37] propose a tabu search approach in order to prevent the excess load to the vehicle. In contrast to Modares et al.'s work [52], they consider the depot as a codebook vector connecting all rings and it can be updated as the other codebook vectors on the ring. Customers are assigned to the rings by assigning to a codebook vector on the ring. If the capacity of a vehicle is exceeded, then a customer which is closest to another ring is dropped from that vehicle and is given a tabu status to prevent the dropped customer to be assigned back to the depot for a number of iterations.

In our algorithm, the neuron representing the depot's location is not fixed. The codebook representing the depot's location is included in the neighborhood of the winner codebook. The codebook for the depot may also be the winner neuron. Codebooks which are on different routes but connected to the same depot are also in the neighborhood of the winner neuron. On the other hand, the codebooks which are on different depots are independent from each other. Figure 4.7 shows neighborhood structure of the nodes connected to the same depot. In the figure, the winner neuron is indexed by zero. The number on the other nodes gives the index distance between the winner and the other nodes. The winner node may be any node including the depot.

The index values are given to the nodes in the order in their respective rings. For instance, the nodes which are on the starting position of the ring take the index value 1. The index distance between a codebook and the winner node is calculated in Algorithm 6. *ncod* is the number of codebooks (nodes) on a ring. r_j is the index of the node j on its defined vehicle. c represents the winner node. $dist_{cj}$ is the index distance

between node j and winner node c, fac_j is the index of facility assigned to node j and veh_j is the index of vehicle assigned to node j.

Algorithm 6: Index distance calculation in SOM

If $fac_j = fac_c$

  If $veh_j = veh_c$

    If $|r_c - r_j| \leq ncod - |r_c - r_j|$ then

      $dist_{cj} = |r_c - r_j|$

    Else

      $dist_{cj} = ncod - |r_c - r_j|$

    End If

  ElseIf $veh_j \neq veh_c$

    If $r_j \leq ncod - r_j$ and $r_c \leq ncod - r_c$ Then

      $dist_{cj} = r_j + r_c$

    ElseIf $r_j \leq ncod - r_j$ and $r_c > ncod - r_c$ Then

      $dist_{cj} = r_j + ncod - r_c$

    ElseIf $r_j > ncod - r_j$ and $r_c \leq ncod - r_c$ Then

      $dist_{cj} = r_c + ncod - r_j$

    ElseIf $r_j > ncod - r_j$ and $r_c > ncod - r_c$ Then

      $dist_{cj} = ncod - r_j + ncod - r_c$

    End If

  End If

  $\Lambda_{jc} = \exp\left(-\frac{dist_{cj}^2}{\sigma^2}\right)$

ElseIf $fac_j \neq fac_i$ then

$\Lambda_{jc} = 0$

End If

Algorithm 7 represents the SOM algorithm we propose for the Multi Depot LRP (LRPSOM) based on the work of Schwardt and Dethloff [37]. In the algorithm all the vehicles are assigned to the depots initially. In other words the depots and the number of vehicles are defined at the beginning of the algorithm.

Algorithm 7: *SOM algorithm for LRP (LRPSOM)*

1. *Initialize the algorithm*

   Define the number of depots, the number of vehicles assigned to the depot and the capacities of the vehicles.

   Define the number of codebooks, initial neuron weights, learning rate, neighborhood function, learning rate momentum and neighborhood momentum, termination criterion and a tabu counter.

2. *Increase the iteration counter by one*

   Determine a random representation sequence of input vectors for the initial iteration

3. *Select the input vector in the sequence*

   IF input vector is assigned to a vehicle Then

   Disconnect the input vector

   Update the vehicle's load

   END IF

4. *Calculate the Euclidean distance between the input selected and each codebook that is not on a ring with a tabu counter>0.*

   If tabu counter>0, then the distance is set to a big real value M.

5. *Select the codebook vector, having the minimum distance to the input vector as the center.*

6. *Assign the input vector to the center codebook.*

   (It is also assigned to the vehicle and the depot to which the codebook belongs)

7. *Update the load of the center's ring*

8. *Reduce the tabu counter by 1 for each vehicle with a tabu counter>0*

9. *Adapt the weight vectors*

10. *Update the center's vehicle load if capacity excess occurs*

    10.1. *Compute the distances between the customers and node's of vehicles for which tabu counter = 0. If tabu counter>0 then set a big real value M for the distance*

    10.2. *Disconnect the looser input vector which is closest to another vehicle's node*

10.3. Set a tabu counter>0 for the input vector and the ring

10.4. Update the load of the center's ring

If center's vehicle load still exceeds the capacity Then GoTo Step 10.1

Else GoTo Step11

11. If all input vectors presented once in the random order then GoTo Step 12

Else GoTo Step 3

12. Update the learning rate, neighborhood width and the iteration counter

If iteration counter reaches to a predefined maximum value

Then stop the algorithm Else GoTo Step 2

The consequent coordinates of the depot's locations are defined by the codebook coordinates representing the depots. Setting the first and last customers of the routes as fixed points for a Weber problem, the coordinates of the depot locations can be redefined. According to the approach called SOMW proposed by Schwardt and Dethloff [37], customer demands directly connected to the depots are set to one and the Euclidean distance single facility location problem is solved for the depot connected to these customers.

It can be now controlled that if all the constraints of the LRP formulation are satisfied by Algorithm 7. In order to satisfy the constraint (4.2) we apply a mechanism as follows. We do not connect the presented input to a vehicle if the tabu counter between the input and the vehicle is bigger than 0. However we can connect the input to the nearest node of a ring without considering if the connected vehicle exceeds its capacity. If the capacity constraint is violated after the connection we have to choose an input to disconnect from the vehicle. In order to select the suitable customer, we compute the distances between the customers of the winner vehicle with other vehicles by finding the minimum distance between the ring nodes and the customer locations. The closest customer to another vehicle (the tabu counter should be equal to 0 between this input and the vehicle closest to it) is selected as the looser input to be disconnected from the vehicle.

Other constraints are easily satisfied because of the suitable structure of the

SOM algorithm for the routing problems. Each customer is assigned to a single neuron and the vehicle it presents. This is the same expression declared by the constraint (4.3). The vehicle routes are represented by one dimensional rings. These rings visit each customer only once, because the customer is assigned to a single neuron on this circular path. Hence this condition satisfies the constraints (4.4) and (4.5). Besides the ring structure of the SOM already satisfies the constraint (4.6).

In the SOM algorithm we initially assign a certain number of vehicles to a depot. The vehicles are assumed to have equal capacities and equal properties in our problem. All vehicles have to be assigned to exactly one depot at the beginning of the SOM algorithm. Hence, constraints (4.7) and (4.8) have to be ensured at the beginning in order to provide the initial conditions to run the SOM algorithm. The number of the vehicles assigned to the depots cannot be changed while running the SOM algorithm. Hence, the number of vehicles assigned to a depot is predefined.

In Figure 4.8, the LRPSOM algorithm we propose is illustrated. Four depots are opened. Each depot has four vehicles represented by the four rings around the depots. The small dots represent the customer locations. At the beginning of the algorithm, the depots are placed on randomly selected customer locations. The rings are connected to the depots, resembling the leaves of flowers. At each iteration, all customers are presented exactly once in the algorithm and assigned to the vehicles. SOMW is applied after the last iteration, determining the final coordinates of the depots.

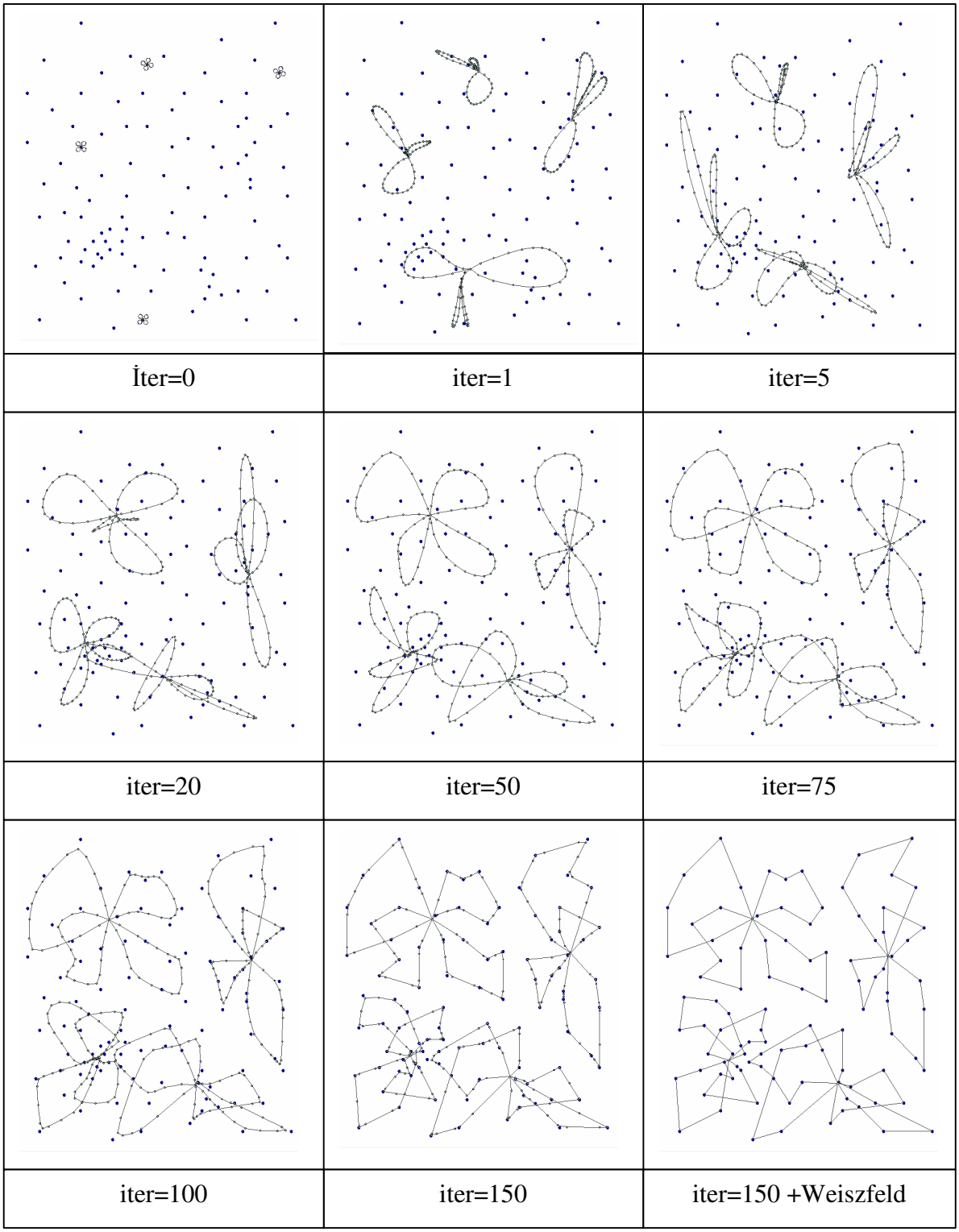

Figure 4.8. Formation of the solution in LRPSOM

5. COMPUTATIONAL RESULTS

In this chapter we present the experimental results for the CMFWP and the CCMFLRP by using the heuristic algorithms that we introduced in the previous chapters. First, we solve the CMFWP problem and present the results obtained by the various metaheuristics and the neighborhood structures we proposed. Then we present the results for CCMFLRP which are attained by the SOM based method.

5.1. Results for the Capacitated Multi-Facility Weber Problem

In this section we compare the new heuristics with the existing methods both in terms of solution quality and computation time on a number of test instances. Test problems are collected from different sources. The instances are grouped in five sets. The first set RCMFWP includes the rectilinear distance problems. Five problems (R16, R23, R26, R29, R30) are obtained from Sherali et al. [20] and three problems (R8, R9, R15) are given in Sherali et al. [17]. Sixteen Euclidean problems used in this research are obtained from two sources: ten instances (E2-E11) from [54] and the remaining six instances (E15–E20) from Sherali et al. [17]. We divided these sixteen instances into two sets according to the problem sizes. Set E1CMFWP includes the seven instances (E2-E8). Remaining nine instances form the set E2CMFWP. Fourth set SECMFWP including the squared Euclidean test instances can be found in Sherali and Tunçbilek [19]. Finally, LpCMFWP instances obtained from [17] form the last set; parameter p of the ℓ_p-distance is set to the following values: $p = 1.25$, $p = 1.5$, $p = 1.75$ respectively.

The size of these test instances and the best known solutions obtained so far are give in Tables 5.1, 5.2, 5.3, 5.4 and 5.5. For each instance we provide the number of facilities to be located (m), the number of customers (n), the best objective values and the corresponding CPU times.

The computation times required to solve the RCMFWP (R16, R23, R26, R29, R30) implemented by Sherali et al. [20] and the squared Euclidean distance problems

Table 5.1. Rectilinear distance test instances for the CMFWP

Instance	(m, n)	BK/opt	CPU time
R8	(4,8)	793	107.30 sec
R9	(5,15)	9619	419.13 sec
R15	(5,10)	3427	158.00 sec
R16	(4,10)	259	28.43 sec
R23	(5,8)	238	26.34 sec
R26	(5,12)	284	203.08 sec
R29	(5,15)	729	310.28 sec
R30	(5,20)	746	35.02 sec

Table 5.2. Euclidean distance test instances for the CMFWP 1

Instance	(m, n)	BK/opt	CPU time
E2	(2,4)	247.28	0.20 sec
E3	(2,4)	214.34	0.90 sec
E4	(3,5)	24.00	2.30 sec
E5	(3,5)	73.96	2.00 sec
E6	(3,9)	221.40	66.40 sec
E7	(3,9)	871.62	42.20 sec
E8	(4,8)	609.23	6 min

Table 5.3. Euclidean distance test instances for the CMFWP 2

Instance	(m, n)	BK/opt	CPU time
E9	(5,15)	8169.79	23 min
E10	(5,20)	12846.87	134 min
E11	(5,20)	1107.18	73 min
E15	(5,10)	2595.47	8 min
E16	(6,10)	7797.21	9 min
E17	(7,10)	6967.90	315 min
E18	(8,10)	1564.46	468 min
E19	(9,10)	3250.68	12 min
E20	(10,10)	7719.00	462 min

Table 5.4. Squared Euclidean distance test instances for the CMFWP

Instance	(m, n)	BK/opt	CPU time
SE9	(4,8)	875.34	227.09
SE16	(4,15)	3800.54	14.75
SE21	(4,24)	6805.43	98.02

Table 5.5. ℓ_p distance test instances for the CMFWP

Instance	(m, n)	BK/opt	CPU time
Lp8, p=1.25	(4,8)	710.20	114.49
Lp8, p=1.50	(4,8)	661.90	126.71
Lp8, p=1.75	(4,8)	630.72	141.94
Lp9, p=1.25	(5,15)	8998.93	998.44
Lp9, p=1.50	(5,15)	8646.61	588.50
Lp9, p=1.75	(5,15)	8350.95	806.43
Lp15, p=1.25	(5,10)	3046.07	254.37
Lp15, p=1.50	(5,10)	2827.55	294.39
Lp15, p=1.75	(5,10)	2689.12	215.42

are run by Sherali and Tuncbilek [19] were experimented by using an IBM 3090 machine. The CPU times of the RCMFWP (R8, R9, R15), E1CMFWP, E2CMFWP and LpCMFWP instances were collected on a Sun Ultra 1 workstation having 256 Mbytes of RAM.

Orbay [55] has improved the best solutions for problems R30, SE16 and Lp9(p=1.50). We adjust the best known solutions of these problems presented in Table 5.6.

We coded the new heuristics in Microsoft Visual Basic 6.0 and ran the programs

Table 5.6. Improved test instances

Instance	(m, n)	Best known
R30	(5,20)	745
SE16	(4,15)	3591.53
Lp9, p=1.50	(5,15)	8609.12

on a Laptop computer with 1.7 GHz Pentium Centrino processor having 256 MByte RAM.

5.1.1. Simulated Annealing Heuristic

In the implementation of the SA-based heuristic the initial extreme point is found by applying the northwest corner rule [43]. The value of T_0 is chosen such that the probability of accepting an average bad move early in the algorithm is $p = 0.95$. Thus, T_0 is computed by the formula

$$T_0 = -\frac{|\bar{\Delta}|}{\ln(0.95)} \tag{5.1}$$

The number of iterations L performed at any temperature T is set to four times the neighborhood size ($r = 4$) for one-variable exchange. r is assigned a value of one for two-variable exchange because of the large neighborhood size associated with the two-variable exchange. We evaluate the computation time and accuracy of the SA-based heuristic at different values of cooling rate α and set its value equal to 0.9 independent of the neighborhood structure. The algorithm is terminated when the percentage of accepted solutions during five consecutive cycles (each temperature update is referred to as a cycle) is less than 5%.

The best solutions and the computation time obtained by the one-variable exchange for rectilinear instances RCMFWP are satisfactory as indicated in Table 5.7; however solutions are not accurate for all instances such as for R16, R26 and R29. The solutions for the two-variable exchange rectilinear cases in the same table are obtained within a reasonable amount of time. Best solutions are found for all instances except the instance R-30 resulting the best deviation to be 0.10 %. The average deviation is 1.12% that is considerably accurate. The average worst deviation for the problem set is 3.85%.

For small Euclidean distance problems E1CMFWP presented in Table 5.8, the

accuracy and the computation time of the SA-based heuristic are considerably well enough for one-variable exchange. For Euclidean instances solved by two-variable exchange, optimal solutions are found for most of the instances in the set with some exceptions.

The results for the E2CMFWP instances are shown in Table 5.9. One variable exchange method is not adequate for all instances. Average of the runs is over 10% and even best solution for E19 is over 10%. The reason for this is that the solution cannot search a large portion of the solution space with one-variable exchange neighborhood structure. However, best solutions are equal to best known values for all instances when the two-variable exchange neighborhood structure is employed. We observe that all the solutions are not accurate enough as it can be observed from the average and worst solution values of some instances like E11 and E19.

For SECMFWP instances indicated in Table 5.10, the accuracy is not satisfactory for all runs although the best solutions are found for one-variable neighborhood structure. For the two-variable exchange structure the results are satisfactory except the instance SE21

LpCMFWP instances presented in Table 5.11 have longer computation times because of the longer location subproblem solution. The best results for the instances solved by one-variable exchange neighborhood structure are satisfactory similar to the Euclidean instances which have the same customer and facility set (E8, E9, E15). With two-variable exchange, Lp distance instances are solved in longer computation times, however the best solutions are very accurate.

In conclusion, we can say that, the accuracy of the two-variable neighborhood is satisfactory while the computation time is higher than the one-variable exchange because of the neighborhood size.

Table 5.7. RCMFWP runs with SA

Instance	One variable exchange				Two variable exchange			
	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)
R8	0.00	0.21	2.14	6	0.00	0.21	2.14	21
R9	0.00	0.05	0.16	31	0.00	0.00	0.02	257
R15	0.00	3.87	4.61	18	0.00	1.74	4.26	114
R16	0.00	8.26	35.52	10	0.00	1.24	9.27	43
R23	0.00	0.00	0.00	10	0.00	0.00	0.00	46
R26	1.41	10.00	26.06	19	0.00	2.54	7.75	135
R29	0.96	6.30	11.52	27	0.00	1.04	3.43	270
R30	1.07	2.78	4.16	48	0.81	2.16	3.89	605
Average	0.43	3.93	10.52	21	0.10	1.12	3.85	186

Table 5.8. E1CMFWP runs with SA

Instance	One variable exchange				Two variable exchange			
	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)
E2	0.00	0.00	0.00	1	0.00	0.00	0.00	1
E3	0.00	0.00	0.00	2	0.00	0.00	0.00	1
E4	0.00	0.00	0.00	8	0.00	0.01	0.06	10
E5	0.00	0.00	0.00	11	0.00	0.00	0.00	13
E6	0.00	0.01	0.06	16	0.00	0.00	0.00	95
E7	0.00	0.00	0.00	23	0.00	0.00	0.00	81
E8	0.00	1.98	8.15	51	0.00	0.21	1.34	163
Average	0.00	0.28	1.17	16	0.00	0.03	0.20	52

Table 5.9. E2CMFWP runs with SA

Instance	One variable exchange				Two variable exchange			
	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)
E9	0.00	0.58	3.39	525	0.00	0.00	0.99	3823
E10	0.00	0.06	0.10	413	0.00	0.00	0.60	5479
E11	0.00	20.31	41.19	605	0.00	4.01	25.37	8217
E15	0.00	8.17	24.32	110	0.00	0.24	1.22	647
E16	0.00	5.56	9.41	153	0.00	4.04	11.07	1147
E17	3.61	10.18	15.39	184	0.00	2.47	4.22	1782
E18	2.45	38.83	89.60	408	0.00	3.38	9.24	3287
E19	11.55	20.11	43.59	451	0.00	8.08	15.35	4778
E20	0.01	3.65	14.55	724	0.01	0.01	0.01	10087
Average	1.96	11.94	26.84	397	0.00	2.47	10.05	4361

Table 5.10. SECMFWP runs with SA

Instance	One variable exchange				Two variable exchange			
	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)
SE9	0.00	1.75	15.03	8	0.00	0.00	0.00	26
SE16	0.00	0.69	6.33	24	0.00	0.16	0.92	164
SE21	0.00	11.13	21.31	55	4.72	8.14	10.74	590
Average	0.00	4.52	14.22	29	1.57	2.77	3.89	260

Table 5.11. LpCMFWP runs with SA

Instance	One variable exchange				Two variable exchange			
	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)
Lp8, p=1.25	0.00	3.84	7.41	127	0.00	0.48	2.52	493
Lp8, p=1.50	0.00	2.16	7.36	133	0.00	0.00	0.01	412
Lp8, p=1.75	0.00	1.46	8.85	111	0.00	0.12	1.15	335
Lp9, p=1.25	0.01	0.17	0.73	1933	0.01	0.01	0.03	15771
Lp9, p=1.50	0.00	0.26	1.05	1721	0.00	0.00	0.00	11171
Lp9, p=1.75	0.00	0.09	0.42	1382	0.00	0.00	0.01	11300
Lp15, p=1.25	0.00	4.58	6.69	260	0.00	1.77	11.00	1634
Lp15, p=1.50	0.00	7.05	8.61	288	0.00	1.77	15.30	1768
Lp15, p=1.75	1.21	8.72	10.68	251	0.00	1.22	9.78	1799
Average	0.14	3.15	5.76	690	0.00	0.60	4.42	4965

5.1.2. Threshold Accepting Heuristic

In the threshold accepting heuristic, the initial extreme solution is found by northwest corner rule. We take the probability of acceptance of accepting bad solutions to be approximately 0.95 at the beginning of the algorithm. Considering the acceptance criterion in TA we compute the initial threshold T_{h0} as follows. We select n random pairs of solutions as we applied in SA by Northwest Corner Rule and compute T_h for all pairs according to the equation with equation

$$f(S_1) - f(S_2) = T_h f(S_2) \text{ for } \; f(S_1) \geq f(S_2) \tag{5.2}$$

which gives

$$T_h = f(S_1)/f(S_2) - 1 \qquad \text{for } \; f(S_1) \geq f(S_2). \tag{5.3}$$

We suppose the T_h values are normally distributed and calculate T_{h0} such as $T_{h0} = \mu + 2\sigma$ where μ is the mean and σ is the standard deviation of T_h values of the pairs.

Table 5.12. RCMFWP runs with TA

Instance	One variable exchange				Two variable exchange			
	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)
R8	0.00	3.33	14.00	11	0.00	0.00	0.00	26
R9	0.00	0.08	0.31	79	0.00	0.00	0.00	306
R15	0.00	4.29	9.69	27	0.00	2.45	4.26	93
R16	0.00	11.97	33.98	13	0.00	4.32	17.76	42
R23	0.00	0.25	2.52	35	0.00	0.00	0.00	122
R26	0.00	19.51	28.52	60	0.00	3.17	17.61	285
R29	0.00	4.09	11.52	84	0.00	1.18	4.25	482
R30	0.00	1.91	2.68	145	0.00	2.11	4.56	1240
Average	0.00	5.68	12.90	57	0.00	1.65	6.06	325

The number of iterations L performed at any threshold T_h is determined to be equal to four times the neighborhood size ($r = 4$) for one-variable exchange. L is taken to be equal to the neighborhood size ($r = 1$) for two-variable exchange. The value of α is set equal to 0.9. When the percentage of accepted solutions during five consecutive cycles (each temperature update is referred to as a cycle) is less than 5%, we terminate the runs.

The computational results for the one variable exchange and two variable exchange are presented in Tables 5.12, 5.13, 5.14, 5.15, 5.16 for RCMFWP, E1CMFWP, E2CMFWP, SECMFWP, LpCMFWP problem sets respectively. Studying on the results obtained, similar comments with SA can be made for each problem set and neighborhood structure. We conclude that there is not a significant difference between the TA and SA methods with respect to the solution quality and the efficiency of the problem.

Table 5.13. E1CMFWP runs with TA

Instance	One variable exchange				Two variable exchange			
	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)
E2	0.00	0.00	0.00	1	0.00	0.00	0.00	1
E3	0.00	0.00	0.00	1	0.00	0.02	0.21	1
E4	0.00	0.00	0.00	15	0.00	0.00	0.00	17
E5	0.00	0.00	0.00	10	0.00	0.00	0.00	12
E6	0.00	0.01	0.08	12	0.00	0.00	0.00	94
E7	0.00	4.98	13.75	32	0.00	0.00	0.00	80
E8	0.00	5.44	31.72	49	0.00	0.14	1.40	141
Average	0.00	1.49	6.51	17	0.00	0.02	0.23	49

Table 5.14. E2CMFWP runs with TA

Instance	One variable exchange				Two variable exchange			
	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)
E9	0.00	3.66	34.31	566	0.00	0.06	0.56	3611
E10	0.00	0.07	0.10	179	0.00	0.00	0.01	2657
E11	0.00	29.41	62.08	423	0.00	7.62	25.37	5310
E15	0.00	14.12	23.53	100	0.00	0.28	1.22	561
E16	0.00	5.98	15.43	119	3.09	4.59	6.33	1001
E17	0.00	1.75	6.80	227	0.00	2.39	10.66	1865
E18	0.00	6.27	30.99	350	0.00	2.91	19.01	2897
E19	11.70	20.62	54.51	511	0.00	3.80	13.30	5415
E20	0.01	5.25	33.30	1372	0.01	0.07	0.61	15121
Average	1.30	9.68	29.01	427	0.34	2.41	8.56	4271

Table 5.15. SECMFWP runs with TA

Instance	One variable exchange				Two variable exchange			
	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)
SE9	0.00	9.93	34.61	20	0.00	0.00	0.00	66
SE16	0.00	10.93	64.63	71	0.00	1.27	6.33	390
SE21	3.04	13.98	16.64	126	0.00	7.01	15.55	1170
Average	1.01	11.61	38.63	72	0.00	2.76	7.29	542

Table 5.16. LpCMFWP runs with TA

Instance	One variable exchange				Two variable exchange			
	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)
Lp8, p=1.25	0.00	0.31	3.03	94	0.00	0.44	2.23	298
Lp8, p=1.50	0.00	3.42	9.69	112	0.00	0.00	0.03	277
Lp8, p=1.75	0.00	4.23	21.48	90	0.00	0.41	4.06	199
Lp9, p=1.25	0.01	0.10	0.91	1580	0.01	0.01	0.03	11821
Lp9, p=1.50	0.00	0.22	0.44	1926	0.00	0.04	0.44	8504
Lp9, p=1.75	0.00	7.54	37.19	1266	0.00	0.00	0.01	8356
Lp15, p=1.25	0.00	8.55	11.05	176	0.00	3.54	11.05	730
Lp15, p=1.50	0.00	6.73	16.26	189	0.00	1.60	15.95	845
Lp15, p=1.75	0.00	13.61	20.33	169	0.00	0.47	2.29	955
Average	0.00	4.97	13.38	622	0.00	0.72	4.01	3554

5.1.3. Genetic Algorithm Heuristic

In the GA-based heuristic, the population size is equal to the lower bound on the number of extreme points which is given in [14] as $n!/(n-m+1)!$ If this value is higher than 100, it is taken as 100. Parent selection is performed by the binary tournament method. Crossover and mutation probabilities are equal to one. We use a steady-state replacement scheme where the offspring replaces the worst individual in the population if the following conditions are satisfied. First, the fitness of the offspring should be lower than that of the worst individual. Second, the offspring should not have duplicates in the population.

The number of generations is the crucial point for the accuracy and the efficiency of the problem. In the first solution method, we run each instance with a fixed number of generations G=1000. On the basis of the results we draw the following conclusions: Results for the RCMFWP instances are presented in Table 5.17. The solutions have considerably low computation times. Besides, we are able to find best known solutions for six of the problems. As indicated in Table 5.18, the results for E1CMFWP instances are also very good. Best known solutions are found for each problem except E8. For the larger Euclidean instances shown in Table 5.19 we could not able to find the best known results for all instances (e.g. E11 and E19). However total run time is clearly much more less than the SA and TA algorithms proposed. Squared Euclidean instances are shown in Table 5.20. Larger problem SE21 behaves different from the other instances as was observed in SA and TA solutions before. For LpCMFWP instances presented in Table 5.21 we are able to find near optimal solutions.

In the second solution method, in order to have a better comparison between the two heuristics GA and SA, the computation time of the GA-based heuristic for each instance is limited by the amount of time that is required by the SA-based heuristic with two-variable-exchange for the same instance. Therefore, the number of generations turns out to be different for each instance. Best solutions found are close to the best known values for RCMFWP instances presented in Table 5.17. However, the results are not as well as two-variable simulated annealing results. For E1CMFWP instances,

Table 5.17. RCMFWP runs with GA

Instance	G=1000				CPU time of two-variable SA			
	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)	Best % Dev.	Avg % Dev.	Worst % Dev.	No of G
R8	0.00	1.61	14.00	6	0.00	0.00	0.00	3658
R9	0.02	0.67	2.19	10	0.00	0.05	0.16	25383
R15	0.00	4.70	16.92	7	0.00	2.98	4.26	15200
R16	0.00	11.43	33.98	6	0.00	7.80	18.53	6880
R23	0.00	0.00	0.00	6	0.00	0.00	0.00	8000
R26	0.00	7.92	24.30	8	0.00	4.08	22.89	16615
R29	0.00	9.05	17.42	11	0.00	4.80	11.25	25414
R30	9.13	17.26	28.32	14	2.01	3.07	11.41	42461
Average	1.14	6.58	17.14	9	0.25	2.85	8.56	17951

presented in Table 5.18, the results are not accurate for problems E4 and E8. For E1CMFWP instances, presented in Table 5.19, again the results are not improved compared with low generation results with G=1000. The average deviation of the runs is 9.16% while near best known solutions can be achieved except the problem E11. The results are very bad compared to SA2 with the same computation time. Squared Euclidean results in Table 5.20, are not improved significantly in spite of the high generation number. Among LpCMFWP instances shown in Table 5.21, only Lp8 instances can be improved because of the high number of generations. As a result, increasing the generation number does not give the expected results compared to the SA algorithm with two-variable exchange. However, GA can find satisfactory results in a small time period as tried with G=1000.

Table 5.18. E1CMFWP runs with GA

Instance	G=1000				CPU time of two-variable SA			
	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)	Best % Dev.	Avg % Dev.	Worst % Dev.	No of G
E2	0.00	0.00	0.00	2	0.00	0.00	0.00	531
E3	0.00	0.00	0.00	4	0.00	0.00	0.00	263
E4	0.00	19.43	29.69	6	0.00	19.35	29.68	1709
E5	0.00	0.00	0.00	6	0.00	0.00	0.00	2152
E6	0.00	0.04	0.10	18	0.00	0.01	0.07	5315
E7	0.00	0.38	3.80	8	0.00	0.00	0.00	9971
E8	0.03	3.33	8.19	18	0.00	1.56	7.11	8936
Average	0.00	3.31	5.97	9	0.00	2.99	5.27	4125

Table 5.19. E2CMFWP runs with GA

Instance	G=1000				CPU time of two-variable SA			
	Best % Dev.	Avg % Dev.	Worst % Dev.	CPU time (s)	Best % Dev.	Avg % Dev.	Worst % Dev.	No of G
E9	0.49	1.83	1.65	47	0.00	0.00	0.00	82217
E10	0.73	3.76	1.32	37	0.00	0.04	0.09	148591
E11	25.63	48.58	84.88	31	16.96	28.63	42.79	261901
E15	0.00	17.40	27.79	15	0.00	8.47	23.53	43850
E16	0.87	7.28	10.73	24	0.00	4.82	10.70	48809
E17	0.52	7.25	25.61	28	0.00	4.64	15.39	69881
E18	3.36	33.86	62.01	28	0.00	19.64	33.93	119525
E19	17.59	26.10	44.22	34	0.07	14.33	16.34	142629
E20	0.61	4.15	9.38	35	0.01	1.90	9.30	286132
Average	5.53	16.69	29.73	31	1.89	9.16	16.90	133726

Table 5.20. SECMFWP runs with GA

Instance	G=1000				CPU time of two-variable SA			
	Best	Avg	Worst	CPU	Best	Avg	Worst	No of
	% Dev.	% Dev.	% Dev.	time (s)	% Dev.	% Dev.	% Dev.	G
SE9	0.00	3.05	0.00	5	0.00	0.00	0.00	4960
SE16	0.00	3.49	15.34	9	0.00	1.90	6.33	18480
SE21	5.49	22.43	47.08	14	2.36	13.21	21.24	42910
Average	1.83	9.66	20.81	9	0.79	5.04	9.19	22117

Table 5.21. LpCMFWP runs with GA

Instance	G=1000				CPU time of two-variable SA			
	Best	Avg	Worst	CPU	Best	Avg	Worst	No of
	% Dev.	% Dev.	% Dev.	time (s)	% Dev.	% Dev.	% Dev.	G
Lp8, p=1.25	0.02	6.09	23.84	918	0.00	0.00	0.01	14880
Lp8, p=1.50	0.05	2.65	9.81	870	0.00	0.00	0.01	16913
Lp8, p=1.75	0.01	1.92	9.39	728	0.00	0.83	8.26	17385
Lp9, p=1.25	0.03	1.39	4.80	1982	0.02	0.09	0.34	1005
Lp9, p=1.50	0.46	1.86	4.28	2117	0.01	0.01	0.01	1517
Lp9, p=1.75	0.51	1.49	1.97	1725	0.00	0.14	1.05	1221
Lp15, p=1.25	0.00	8.88	14.49	443	0.00	7.62	11.05	2169
Lp15, p=1.50	0.00	12.76	27.29	417	0.00	11.38	16.26	1891
Lp15, p=1.75	0.00	14.14	31.78	411	0.00	12.20	20.33	1828
Average	0.12	5.69	14.18	1068	0.00	3.59	6.37	6534

5.2. Results for the Continuous Capacitated Multi-Facility Location-Routing Problem

Location-Routing problem models presented in the literature have several different problem perspectives. The authors propose different objective function values and constraints to LRP. Perl's instances [56] are the only instances for LRP defined. However, the LRP model proposed by Perl differs from our model. In our model we do not consider the fixed cost of opening a facility and operating a vehicle. In addition Perl's model is discrete and our model is continuous.

The structure of the Multi Depot Vehicle Routing Problem (MDVRP), a variant of the classical Vehicle Routing Problem (VRP) indicated in the work of Cordeau et al. [57] is very similar to our LRP model. The only difference is that in our model the locations of the facilities are not fixed. The MDVRP instances can also be obtained from the Internet at http://neo.lcc.uma.es/radi-aeb/WebVRP/Problem_Instances /MDVRPInstances.html. We apply our SOM based algorithm to these instances ignoring the depot location given in the problem data.

Some instances have the same customer location and demand structure. They differ from each other only with the depot location. Hence, for LRP these instances are equal. The similar instances can be grouped as p12-p13-p14, p15-p16-p17, p18-p19-p20, p21-p22-p23. We choose p13, p16, p19 and p22 for our test set and ignore the other eight problems. In total, we have 25 test instances in order to test our LRP algorithm.

The instances are presented in Table 5.22. In column 2 the number of depots and the customers are given while in column 3 the number of vehicles assigned to each depot is indicated. The number of vehicles are taken from the best known solutions for MDVRP. In the fourth column best known solutions for MDVRP are given. The last column includes the improved results obtained by changing the locations of the facilities. routes. In order to improve the best known solutions, we update the locations of the depots. The consequent coordinates of the depot's locations are defined by the

codebook coordinates representing the depots. Setting the first and last customers of the routes as fixed points for a Weber problem, the coordinates of the depot locations can be redefined. Customer demands directly connected to the depots are set to one and the Euclidean distance single facility location problem is solved for the depot connected to these customers. Briefly, the improvement is obtained by decreasing the stem distances.

Initial width of the neighborhood σ is taken equal to 4, initial learning rate α is taken equal to 0.3. The decreasing rate of α at each iteration (learning rate momentum) M_α is set 0.995 and the decreasing rate of σ (neighborhood width momentum) M_σ is taken 0.99. Tabu counter for the vehicles that are restricted to connect to a customer is initially taken 10. This means that a vehicle cannot be reassigned to a customer for 10 runs if it is disconnected from the customer because of the under capacity. The number of codebooks on each vehicle is computed with the formula $\left\lceil 3\frac{\#\text{ of } customers}{\#\text{ of } vehicles} \right\rceil$. Finally the number of iterations is taken 150. At each iteration all the customers are presented once. The initial depot locations are set on the coordinates of the customers which are randomly selected.

The new heuristics were coded in Microsoft Visual Basic 6.0 and the programs were run on a laptop computer with 1.7 GHz Pentium Centrino processor having 256 MByte RAM. The solutions are presented in Table 5.23. Number of runs for each problem and the number of feasible solutions obtained from the runs are presented in columns 2 and 3, respectively. In columns 4 and 5 of Table 5.22, respectively, percent deviations of the best and average solutions from the best known MDVRP results given in column 4 of Table 5.23 are reported. On the other hand, in columns 6 and 7 of Table 5.23, the solutions are compared with the improved solutions given in column 5 of Table 5.22.

Although we get better results than the improved results for some problems, our algorithm is not satisfactory. For problems p04, p08, p11 and pr05 even feasible solutions cannot be obtained. There are two reasons for the infeasibility. First, total demand of the customers are very close to the total capacity of the vehicles. Second,

customers become dense in some regions and the rings cannot be assigned to these customers. However, better solutions than the MDVRP best knowns can be obtained for problems p01, p03, p07, pr01.

Table 5.22. Test instances for the CCMFLRP

Instance	depot,customer (m, n)	vehicle (k)	BK MDVRP	BK MDVRP + Location
(1)	(2)	(3)	(4)	(5)
p01	(4,50)	3,4,2,2	576.87	565.17
p02	(4,50)	1,1,2,1	473.53	471.23
p03	(5,75)	2,2,3,2,2	641.19	634.8
p04	(2,100)	8,7	1001.59	992.68
p05	(2,100)	4,4	750.03	747.41
p06	(3,100)	6,4,6	876.50	869.45
p07	(4,100)	4,4,4,4	885.80	871.73
p08	(2,249)	12,13	4437.68	4407.68
p09	(3,249)	11,9,6	3900.22	3854.89
p10	(4,249)	7,7,6,6	3663.02	3609.6
p11	(5,249)	6,5,5,6,4	3554.18	3487.64
p13	(2,80)	4,4	1318.95	1318.95
p16	(4,160)	4,4,4,4	2572.23	2572.23
p19	(6,240)	4,4,4,4,4,4	3827.06	3827.06
p22	(9,360)	4,4,4,4,4,4,4,4,4	5702.16	5702.16
pr01	(4,48)	1,1,1,1	861.32	855.19
pr02	(4,96)	2,2,2,2	1307.61	1284.68
pr03	(4,144)	3,3,3,2	1806.6	1793.24
pr04	(4,192)	3,3,4,4	2072.52	2047.60
pr05	(4,240)	5,4,5,5	2385.77	2348.13
pr06	(4,288)	5,6,6,6	2723.27	2700.32
pr07	(6,72)	1,1,1,1,1,1	1089.56	1083.52
pr08	(6,144)	2,2,2,2,2,2	1666.6	1621.14
pr09	(6,216)	3,3,3,3,3,2	2153.1	2124.57
pr10	(6,288)	4,4,4,4,4,4	2921.85	2873.93

Table 5.23. CCMFLRP Solutions with SOM Algorithm

Inst (1)	No of runs (2)	Feasible (3)	Best %dev mdvrp (4)	Avg %dev mdvrp (5)	Best %dev mdvrp+loc (6)	Avg %dev mdvrp+loc (7)	CPU time (min) (8)
p01	500	464	-1.45	13.09	0.59	15.43	116
p02	500	283	0.24	17.82	0.73	18.40	65
p03	500	489	-1.00	8.14	0.00	9.23	174
p04	500	0	Inf	Inf	Inf	Inf	355
p05	500	327	0.08	6.25	0.43	6.62	257
p06	500	244	4.33	13.62	5.18	14.55	336
p07	500	151	-4.06	3.48	-2.51	5.15	284
p08	100	0	Inf	Inf	Inf	Inf	499
p09	100	1	10.05	10.05	11.35	11.35	435
p10	100	4	2.68	7.18	4.20	8.76	406
p11	100	0	Inf	Inf	Inf	Inf	405
p13	100	88	6.83	15.82	6.83	15.82	40
p16	100	61	9.14	18.78	9.14	18.78	149
p19	100	34	11.64	21.63	11.64	21.63	336
p22	100	7	13.44	19.42	13.44	19.42	781
pr01	100	46	-1.12	7.48	-0.41	8.25	12
pr02	100	48	0.93	8.73	2.73	10.67	53
pr03	100	40	5.44	15.89	6.22	16.75	122
pr04	100	15	7.58	14.30	8.89	15.69	308
pr05	100	0	Inf	Inf	Inf	Inf	489
pr06	100	10	7.16	10.78	8.07	11.72	656
pr07	100	81	0.96	9.17	1.53	9.78	26
pr08	100	34	3.99	10.71	6.90	13.81	126
pr09	100	11	3.88	9.68	5.28	11.15	285
pr10	100	1	6.26	6.26	8.03	8.03	675

6. CONCLUSIONS

CMFWP has a nonconvex objective function and is very difficult to solve to optimality. Various representation techniques and neighborhood structure is introduced for the metaheuristics in order to solve the CMFWP. This research uses the techniques of SA, TA and GA to develop a heuristic for solving the capacitated multi-facility Weber problem efficiently with the rectilinear, Euclidean, squared Euclidean and ℓ_p distances .

The metaheuristics are tested in terms of both their solution quality and computation time on benchmark instances available in the literature. The results are very accurate for the SA-based and TA-based heuristics with two-variable exchange neighborhood structure. On the other hand, one-variable exchange results are less time consuming while has low solution quality. The reason for one-variable exchange to give worse solutions is that it gets stuck near a local minimum and does not have ability to escape from the loop of solutions near this local minimum. In addition, it is observed that there is not an evident difference for the accuracy and computation time between SA and TA algorithms.

In order to make a comparison between the GA and the probabilistic local search algorithms, GA-based heuristic for each instance is run with the amount of time used to solve SA-based heuristic with two-variable exchange. After some number of generations it is observed that the structure of individuals in the population becomes very similar. Hence, this results the solution set in the population to stuck in an area near a local minimum and the algorithm cannot search a wider space in the overall solution set. As a result the solution quality is lower than SA-based and TA-based heuristics proposed in this paper. As a conclusion we can assert that SA-based heuristic provides superior solutions to the CMFWP than the GA-based heuristic.

In CCMFLRP, the determination of locations is as important as determination of routes. We are coerced to integrate the well known location-allocation and vehicle

routing problems. Continuous depot location variables make the problem harder to solve with exact methods. Thus, we propose a meta-heuristic method, SOM based heuristic, for location-routing problem.

In SOM algorithm modified for location-routing problem, for each depot we establish a multi codebook rings which represent the vehicles. These rings are connected with a single codebook which acts as the depot. The capacity constraints of the vehicles are satisfied by a tabu search based method we proposed.

Unfortunately, the results we obtain were worse than the best known MDVRP solutions for the majority of the test instances we used. However, this study is important, being the first attempt in order to solve the continuous multi-facility LRP. For the future research, heuristics other than SOM can be tried for the solution of the problem. Variants of the problems can be considered; for instance vehicle acquisition and depot opening costs can be added to the objective.

REFERENCES

1. Wesolowsky, G., "The Weber problem: history and perspectives", *Location Science,* Vol 1, No. 1, pp. 5–23, 1993.

2. Brimberg, J. and R.F. Love, "Estimating distances," Facility location: A survey of applications and methods, Z. Drezner (Editor), *Spring Series in Operations Research*, Springer-Verlag, New York, pp. 9–32, 1995.

3. Alpaydın, E., İ.K. Altınel and N. Aras, "Parametric distance function versus non-parametric neural networks for estimating road travel distance functions", *European Journal of Operational Research,* Vol. 92, pp. 230–243, 1996.

4. Altınel, İ.K., B.J. Oommen and N. Aras, "Vector quantization for arbitrary distance function estimation", *INFORMS Journal on Computing,* Vol. 9, No. 4, pp. 439–451, 1997.

5. Francis, R.L., L.F. McGinnis, and J.A. White, *Facility layout and location: An analytical approach*, 2nd edition, Prentice Hall, Upper Saddle River, NJ, 1992.

6. Salhi, S. and G.K. Rand, "The effect of ignoring routes when locating depots", *European Journal of Operational Research, Vol. 38, pp.* 150-156, 1989.

7. Cooper, L., "The transportation-location problem", *Operations Research,* Vol. 20, pp. 94–108, 1972.

8. Cooper, L., "Heuristic methods for location-allocation problems", *SIAM Review,* Vol. 6, No.1, pp. 37–52, 1964.

9. Weiszfeld, E., "Sur le point pour lequel la somme des distances de n points donnés est minimum", *Tôhoku Mathematics Journal,* Vol. 43, pp. 355–386, 1937.

10. Aras, N., İ.K. Altınel and M. Orbay, *Efficient heuristics for the Rectilinear dis-*

tance capacitated multi-facility Weber problem, Research Paper Series No: FBE-IE-03/2005-04, Boğaziçi University, Bebek, İstanbul, 2005.

11. Brimberg, J., R. Chen, and D. Chen, "Accelerating convergence in the Fermat–Weber location problem", *Operations Research Letters,* Vol. 22, pp. 151–157, 1998.

12. Frenk, J.B.G., M.T. Melo and S. Zhang, "A Weiszfeld method for a generalized l_p distance minisum location model in continuous space", *Location Science,* Vol. 2, pp. 111–117, 1994.

13. Cooper, L., "Solutions of generalized locational equilibrium models", *Journal of Regional Science,* Vol. 7, pp. 1–18, 1967.

14. Cooper, L., "An efficient heuristic algorithm for the transportation-location problem", *Journal of Regional Science,* Vol. 16, pp. 309–315, 1976.

15. Hansen, P. and N. Mladenović, "Variable neighborhood search: Principles and applications", *European Journal of Operational Research,* Vol. 130, pp. 449–467, 2001.

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

17. Sherali, H.D., I. Al-Loughani, and S. Subramanian, "Global optimization procedures for the capacitated Euclidean and ℓ_p distance multifacility location-allocation problems", *Operations Research,* Vol. 50, pp. 433–448, 2002.

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

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

20. 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.

21. Hansen, P., J. Perreur, and F. Thisse, "Location theory, dominance and convexity: Some further results", *Operations Research,* Vol. 28, pp. 1241–1250, 1980.

22. Laporte, G. "Location-Routing Problems", *Vehicle Routing: Methods and Studies*, B.L. Golden, A.A. Assad (eds), North-Holland, Amsterdam, pp. 163-197, 1988.

23. Min, H., V. Jayaraman and R. Srivastava, "Combined location-routing problems: A synthesis and future research directions", *European Journal of Operational Research*, Vol. 108, No. 1, pp. 1-15, 1998.

24. Ahıpaşaoğlu, S.D., G. Erdoğan and B. Tansel, *Location-routing problems: a review and assessment of research directions*, Working Paper, Department of Industrial Engineering, Bilkent University: Ankara, Türkiye, 2004.

25. Laporte, G., Y. Nobert and S. Taillefer, "Solving a family of multi-depot vehicle routing and location-routing problems", *Transportation Science,* Vol. 22, pp. 161-172, 1988.

26. Perl, J., and M.S. Daskin, "A warehouse location-routing problem", *Transportation Research Part B*, Vol. 19 No. 5, pp. 381-396, 1985.

27. Jacobsen, S.K. and O.B. Madsen, "A comparative study of heuristics for a two-level routing-location problem", *European Journal of Operational Research*, Vol. 5, pp. 378-87, 1980.

28. Clarke, G. and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points", *Operations Research*, Vol. 12, pp. 568-581, 1964.

29. Hansen, P.H., B. Hegedahl, S. Hjortkjaer, and Obel, B., "A heuristic solution to the warehouse location-routing problem", *European Journal of Operational Research*,

Vol. 76, No. 1, pp. 111-127, 1994.

30. Srivastava, R. and W.C. Benton, "The location-routing problem : considerations in physical distribution system design", *Computers and Operations Research*, Vol. 17, pp. 427-435, 1990.

31. Srivastava, R., "Alternate solution procedures for the location-routing problem", *Omega International Journal of Management Science*, Vol. 21, No. 4, pp. 497-506, 1993.

32. Chien T.W., "Heuristic procedures for practical-sized uncapacitated location-capacitated routing problems", *Decision Sciences*, Vol. 24, pp. 995-1021, 1993.

33. Tuzun, D., and L. Burke, "A two-phase tabu search approach to the location routing problem", *European Journal of Operational Research*, Vol. 116, No. 1, pp. 87-99, 1999.

34. Wu, Tai-Hsi, Chinyao Low, Jiunn-Wei Bai, "Heuristic solutions to multi-depot location-routing problems", *Computers and Operations Research*, V.29 No.10, pp.1393-1415, 2002.

35. Lin, S., "Computer solutions of the traveling salesman problem", *Bell System Technology Journal*, Vol. 44, 2245-2269, 1965.

36. Albareda-Sambola, M., J.A. Diaz and E. Fernandez, "A compact model and tight bounds for a Combined Location-Routing Problem", *Computers and Operations Research*, Vol. 32, No. 3, pp. 407-428, 2005.

37. Schwardt M. and J. Dethloff, "Solving a continuous location-routing problem by use of a self-organizing map", *International Journal of Physical Distribution and Logistics Management*, Vol. 35, No. 6, pp. 390-408., 2005.

38. Brimberg, J. and R.F. Love, "Global convergence of a generalized iterative procedure for the minisum location problem with l_p distances", *Operations Research,*

Vol. 41, pp. 1153–1163, 1993.

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

40. Love, R.F., J.G. Morris, George Wesolowsky, *Facilities Location Models and Methods*. Elsevier Science Publishing, New York, USA, 1988.

41. Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. . Teller and E. Teller. "Equations of state calculations by fast computing machines", Journal of Chemical Physics, Vol. 21, pp. 1087–1091, 1953.

42. Kirckpatrick, S., C. D. Gelatt, and M. P. Vecchi. "Optimization by simulated annealing", *Science*, Vol. 220(4598), pp. 671-680, 1983.

43. Bazaraa, M.S., J.J. Jarvis, H.D. Sherali, *Linear Programming and Network Flows*. John Wiley and Sons Inc., Singapore, 1990.

44. Dueck, G. and T. Scheuer, "Threshold Accepting: A general purpose optimization algorithm appearing superior to simulated annealing", *Journal of Computational Physics*, Vol. 90, pp. 161-175, 1990.

45. Holland, J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

46. Eckert, C. and Gottlieb, J., "Direct representation and variation operators for the fixed charge transportation problem". *Proceedings of 7th International Conference on Parallel Problem Solving from Nature – PPSN VII*, Granada, Spain, pp. 345–354, 2002.

47. Gottlieb, J., and L. Paulmann, "Genetic algorithms for the fixed charge transportation problem", *Proceedings of the 5th IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, pp. 330-335, 1998.

48. Gottlieb, J, B. Julstrom, G. Raidl, F. Rothlauf, "Prüfer Numbers: A poor representation of spanning trees for evolutionary search", *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, Morgan Kaufmann Publishers. San Francisco, California, 2001.

49. Durbin, R. and D. Willshaw, "An analogue approach to the travelling salesman problem using an elastic net method", *Nature*, Vol. 326, pp. 689-691, 1987.

50. Kohonen,T., "The Self-Organizing Map", *Proceedings of the IEEE*; Vol. 78, No. 9, pp. 1464-1477, 1990.

51. Angeniol B., G. De-La-Croix and J.Y. Le-Texier, "Self-organizing feature maps and the traveling salesman problem", *Neural Networks*, Vol. 1, pp. 289-293, 1988.

52. Modares, A., S. Somhom, and T. Enkawa, "A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems", *International Transactions in Operational Research*, Vol. 6, pp. 591-606, 1999.

53. Somhom, S., A. Modares, T and Enkava, "Competition-based neural network for the multiple traveling salesman problem with minimax objective", *Computers and Operations Research*, Vol. 26, pp. 395-407,1999.

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

55. Orbay, M., *A mixed integer linear programming approximation based heuristic for continuous capacitated location-allocation problem*, M.S. Dissertation, Boğaziçi University, 2004.

56. Perl, J., "A unified location-routing analysis", *UMI Dissertation Information Service*, June 1983.

57. Cordeau, J. F., M. Gendreau and G. Laporte, "A tabu search heuristic for periodic

and multi-depot vehicle routing problems", *Networks*, Vol. 30, pp. 105-119, 1997.