LOCATION AREA MANAGEMENT FOR MOBILE NETWORKS WITH EVOLUTIONARY ALGORITHMS

By

Bahar Karaoğlu

B.S. in Computer Engineering, Marmara University, 2001

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 Computer Engineering
Boğaziçi University

2004

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis supervisors, Assoc. Prof. Haluk Topçuoğlu and Prof. Fikret Gürgen for their patience, suggestions and contribution to this project. Without their directing, I couldn't achieve this thesis. I want to thank Assoc. Prof. Can Özturan, Asst. Prof. Murat Zeren and Asst. Prof. Nilgün Güler for their participation to my thesis jury. Also I want to thank Prof. Cem Ersoy and İlker Demirkol for sharing their knowledge.

I would like to thank to Hüseyin Öner, who is a tolerant boss and also a good friend of mine. My family always encouraged me for this master degree. So I hope I can make them happy. Lastly, I want to thank my friends Özge İnce, Çetin Meriçli and Erdem Aydınlı for their friendship.

ABSTRACT

LOCATION AREA MANAGEMENT FOR MOBILE NETWORKS WITH EVOLUTIONARY ALGORITHMS

Location management is a very important problem in mobile networks. In general, registration and paging costs are associated with tracking the current location of a mobile user. Considering the whole network as a single location area (LA) maximizes the paging cost and minimizes the registration cost. On the other hand considering each cell as a separate LA maximizes the registration cost and minimizes the paging cost. Partitioning the whole network into location areas and assigning base stations to these location areas can minimize the total cost of registration and paging.

In this work, three evolutionary methods for optimizing the tracking cost of a mobile user by finding an optimal network structure are given and their results are compared. Genetic Algorithms, Multi-Objective Genetic Algorithms and Memetic Algorithms are used to partition a given network into optimal location areas. Finding optimal network structure is known to be NP-Complete. Evolutionary algorithms are suitable for optimizations when normal search algorithms are inefficient. This work gives detailed explanation of implementation details for each algorithm and a comparative study about the performances of algorithms on this particular problem is given.

ÖZET

EVRİMSEL YÖNTEMLERLE HAREKETLİ AĞLARDA YERLEŞİM PLANLAMA

Yerleşim planlama hareketli ağlarda çok önemli bir problemdir. Genelde, kayıt ve arama maliyetleri gezgin bir kullanıcının şu anki konumunun takip edilmesi ile bağlantılıdır. Tüm ağı tek bir yerleşim alanı olarak kabul etmek arama maliyetini maksimize ederken ağda tek bir yerleşim alanı bulunduğundan kayıt olma maliyeti sıfır olacaktır. Öte yandan, ağdaki her hücrenin bir yerleşim alanı olarak kabul edilmesi her yerleşim alanında sadece bir hücre bulunacağından arama maliyetini sıfırlarken kayıt maliyeti maksimize olacaktır. Tüm ağı yerleşim ağlarına bölmek ve her baz istasyonunu bu yerleşim ağlarına atamak toplam kayıt ve arama maliyetini minimize edebilir.

Bu çalışmada, optimal ağ yapısını bularak gezgin bir kullanıcıyı takip etmenin maliyetini optimize etmek için üç evrimsel yöntem ve bu yöntemlerin birbirleri ile kıyaslanmaları sunuldu. Verilen bir ağı optimal yerleşim alanlarına ayırmak için Genetik Algoritmalar, Çoklu-Hedefli Genetik Algoritmalar ve Memetic Algoritmalar kullanıldı. Optimal ağ yapısını bulmak NP-Complete olduğu bilinen bir problemdir. Evrimsel algoritmalar normal arama algoritmalarının yetersiz kaldığı problemler için uygundur. Bu çalışma üç algoritmanın gerçekleştirilmesi ile ilgili detaylı bilgi vermesinin yanında bu algoritmaların sözkonusu problem üzerindeki performanslarının kıyaslamasını da sunar.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ ABBREVIATIONS

c_i	Call traffic rate for each cell I
C_j^{BSC}	Maximum call traffic capacity of each BSC
d_i	Peak call attempt rate of cell i per unit time
D_j^{BSC}	Busy hour call attempt capacity of each BSC
l_{in}	A function that returns 1 if if i^{th} base station resides in if n^{th} location area, 0 otherwise
P_i^{BS}	The paging capacity of each base station
r_i	# of TRXs in each cell i
R_j^{BSC}	Maximum number of TRXs for each BSC.,
λ_i	The paging rate per unit time for each base station in network
BHCA	Busy Hour Call Attempt Rate
BSC	Base Station Controller
BSS	Base Station Subsytem
BTS	Base Tranceiver Station
GA	Genetic Algorithm
GSM	Global System for Mobile Communication
HLR	Home Location Register
IMEI	International Mobile Equipment Identity
LA	Location Area
MA	Memetic Algorithm
MOGA	Multi Objective Genetic Algorithm
MS	Mobile Station
MSC	Mobile Services Switching Center
TRX	Base Station Transmitter
VLR	Visitor Location Register

1. INTRODUCTION

Mobile communication becomes more prominent every day as the globalization and the speed of daily life increases. One of the challenges in mobile networks is *Location Management*, which is to be able to track the current locations of the users. In a typical cellular mobile network, the area of coverage is divided into cells and in general, the shapes of these cells are circles; modeled pictorially as hexagons. Each cell is associated with a base station, which is responsible for communicating with the users in its coverage area, i.e. the associated cell. In order to route incoming calls to appropriate base stations, the network should aware of that which cell the user is currently located in.

The operation of locating a user in the network is called *signaling*. Signaling is a bandwidth-consuming process and it is naturally desirable to minimize signaling cost in order to use available bandwidth as much as possible for yielding revenue. There are two sub-operations in the signaling operation: updating the location of the user with actual network cell which the user is currently in (known as *registration*) and searching the network to find out which cell the user is currently in (known as *paging*). The aim of location management is to find a good balance between the paging and registration operations in such a way to minimize overall signaling cost. Both paging and registration have their own costs and the overall signaling cost is calculated simply by taking sum of total paging and registration costs. Since the users are wandering around with different characteristics of movement, it is not trivial to come up with an algorithm for minimizing the signaling cost easily [1,2].

1.1. A Brief Overview of GSM

GSM stands for Global System for Mobile Telecommunications and has become a worldwide standard. In the cellular telephony architecture, a set of base stations (BS) covers the mobile area. The function of base stations is conveying the incoming calls to the mobile stations within their boundaries. All base stations connect to a mobile switching center (MSC). There are two databases for controlling the roaming. These are visitor

location register (VLR) and home location register (HLR). Figure 1.1 shows a simple representation of cellular telephone architecture.

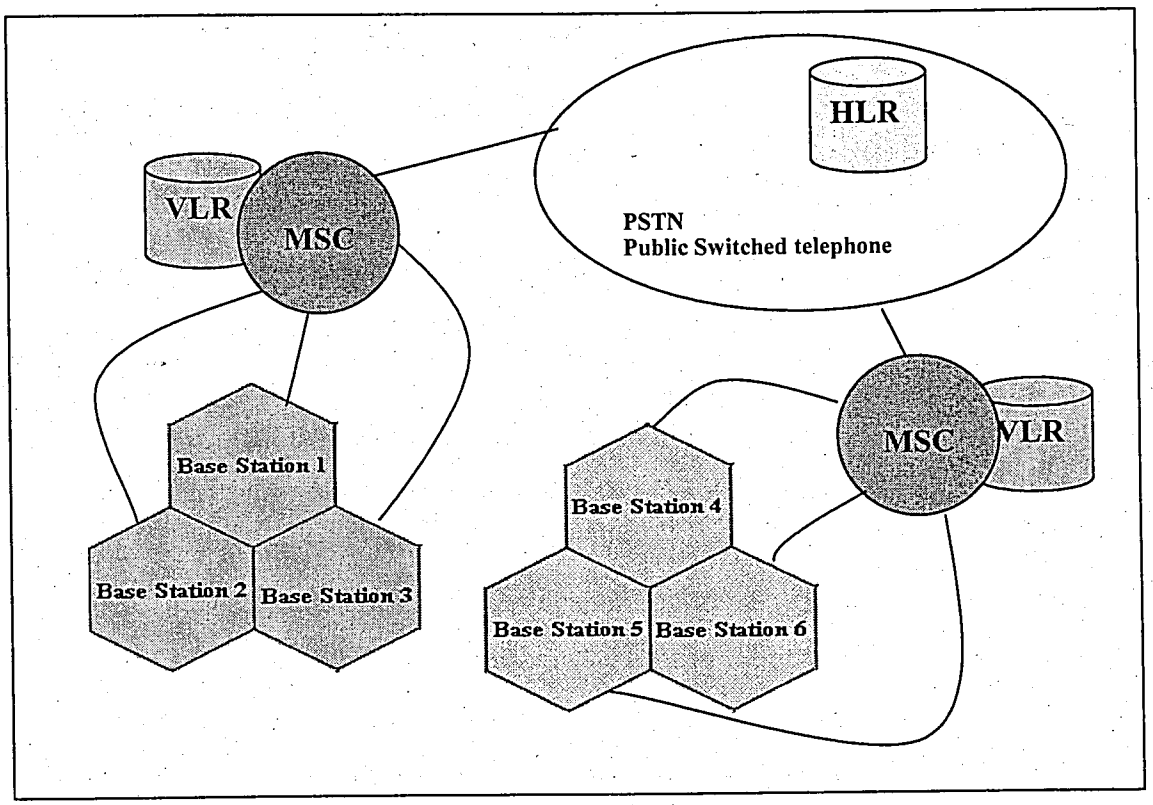

Figure 1.1. A simple representation of cellular telephone architecture

There are three subsystems in a GSM. These are mobile station, base station subsystem and network subsystem. Figure 1.2 shows the architecture of the GSM subsystems. Descriptions of the components of GSM architecture were summarized below [3]:

- **Mobile Station:** Mobile Station is physical equipment that is handheld and portable. A GSM has as many mobile stations as possible.
- **Subscriber Identity Module:** Each MS has a smartcard called subscriber identity module so that mobile stations can have portability. SIM cards can be inserted into other mobile cellular phones so that mobile users can use other cellular phones for communication. SIM has the user information and secret key for identifying the mobile user.

- **Base Transceiver Station:** BTSs provides connection between mobile station and network. There must be many base transceiver stations in a large area. BTSs connect the network with mobile station via um-interface. The communication between BTSs and BSCs is across Abis-interface.
- **Base Station Controller:** BSCs manage the radio resources for one or more BTSs such as radio channel setup, handovers. Base station controllers act as a connector between mobile station and mobile services switching center.
- **Home Location Register:** HLRs stores all the information of the subscribers, along with the current location of the mobile. Usually home location registers are huge databases.
- **Visitor Location Register:** VLRs are similar to the HLRs but they hold only administrative information about the users. VLRs are needed because subscribers can roam in the area, which the VLR is responsible. So when the user leaves the area, the information in the visitor location register is deleted.
- **Mobile Services Switching Center:** The central element of the network subsystem is the MSC. MSCs handle many administrative tasks such as registration, authentication and call routing. Also MSCs are connections between the network and the public fixed networks, PSTN (public switch telecom network).
- **Equipment Identity Register:** EIRs hold information about all the valid equipments in the network. Each mobile station has International Mobile Equipment Identity (IMEI) of its own and it can be tracked by EIR so that if mobile station is stolen it can be reported as invalid by the EIR.

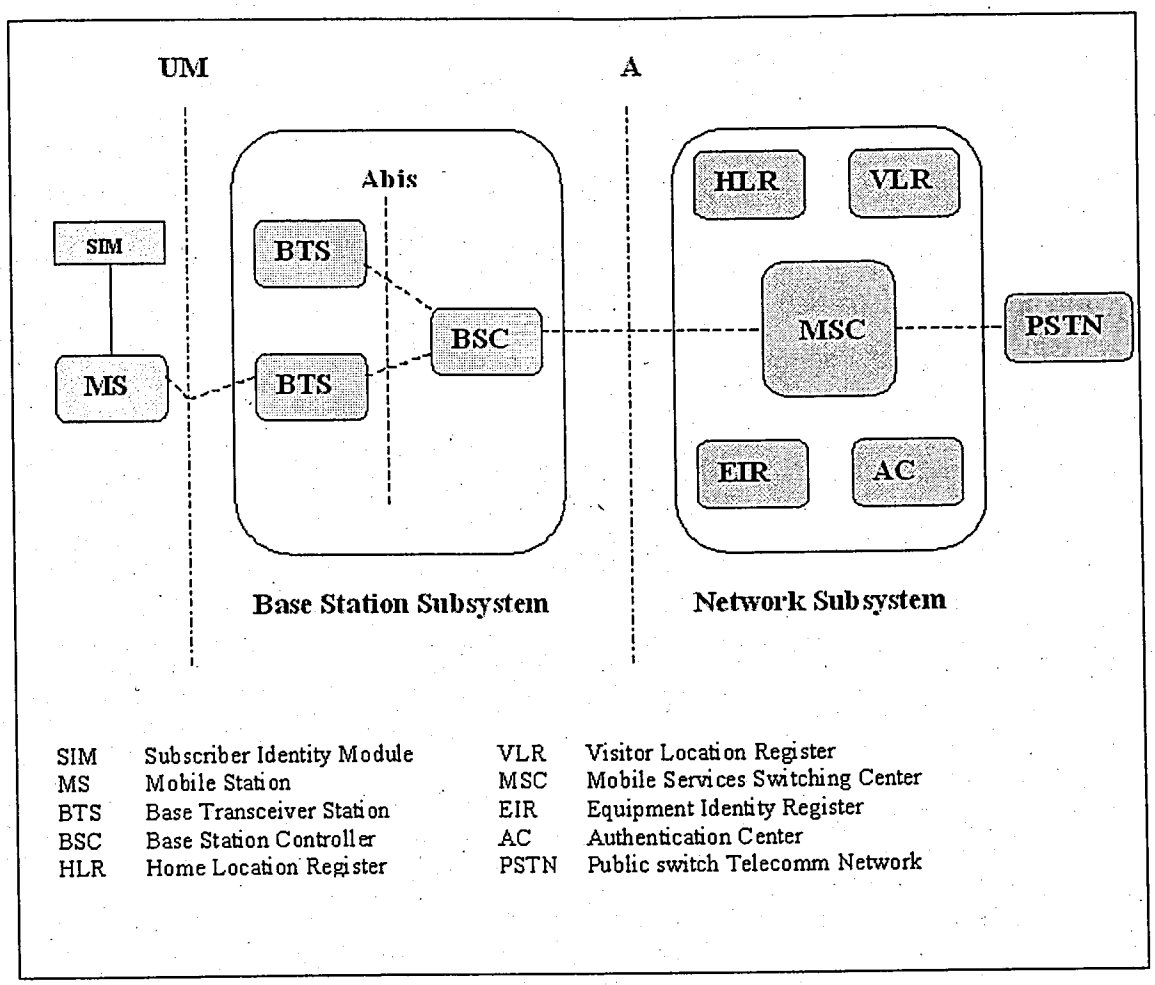

Figure 1.2. Architecture of GSM network [3]

In the mobile network hierarchy, base station controllers are on top of the base stations. Also virtual subsets of cells called *Location Area* (LA) are defined for specifying the location of the user in the network. Whenever user changes his/her location area, he/she is registered to the new location area through a registration operation. In case of an incoming call, the call is directed to the user's last updated location area by related base station controller and the user is searched in the cells of location area through a paging operation. The number of location areas and the number of cells in each of them affects the overall signaling cost. In one of the extreme cases, there can be one location area containing all of the base stations. In such case, every incoming call is directed to that location area and in worst case; all of the cells are searched for finding the appropriate cell where the user is in. Since there is only one location area, user has no chance of transiting to another location area therefore there is no registration operations possible and the paging cost is at its maximum value. This case is known as *never-update scheme* [1,2].

In the other extreme case, the number of location areas can be equal to the number of cells in the network. In such case, each time the user moves to a different cell, a registration operation is performed. Since there is only one cell in each location areas, no paging operation is possible and the registration cost is at its maximum level. This case is known as *always-update scheme*. Our aim in this work is to find such a network structure somewhere in between these two extreme cases that minimizes the total signaling cost.

1.2. Evolutionary Algorithms for Location Area Management

Since the solution space includes all possible network topologies (which are feasible and infeasible), the three assignments (i.e. BS-to-BSC, BSC-to-MSC and BS-to-LA assignments) gives the complexity of the solution. The assignment of cells to switches (i.e. BS-BSC and BSC-MSC assignments) was shown to NP-hard, without considering the location areas [4]. Additionally, minimizing the number of LAs was formulated with bin-packaging problem in [5], which is also NP-hard [6].

Because of the difficulties, in order to satisfy several constraints and its NP-hard nature, we consider techniques that give near optimal solutions. In this study, three different solutions that are in the class of evolutionary algorithms (EAs) are applied for the zone-based location area management problem. They are:

- Genetic Algorithms (GA)
- Multi Objective Genetic Algorithms (MOGA)
- Memetic Algorithms (MA)

Genetic Algorithm (GA) is an evolutionary optimization method based on the Darwinian evolution principles and survival of the fittest rule [7]. By mimicking the populations of living organisms by considering different possible solutions as individuals in these populations and associating fitness to individuals based on their "fitness" to the solution of the problem. By the application of reproduction operators selection, crossover and mutation, the stronger individuals have more chance to reproduce and therefore the individuals in the population becomes more "fit" as the generations passed. We have decided to use GA in this work because our problem is known to be NP-Complete so there

is not a polynomial time algorithm to solve it. Moreover, GA prevents us to get stuck on local minima due to its random exploration in search space.

Multi Objective Genetic Algorithm (MOGA) is a variation of GA [8]. Instead of using a multi-attribute utility function as the cost function for more than one objective, we run the algorithm with cost functions containing different objective, on different populations for each objective and after a certain number of generations, merge them and continue running. This provides variability in population and reproduction of individuals specialized on maximizing only one objective. The offspring of reproduction of two individuals evolved to maximize different objectives probably have a tendency to maximize both of the objectives.

Memetic Algorithm (MA) is another variation of GA that uses a local search operation in the mutation operator [9]. The resultant individual is the one that gives maximum fitness in the local search operation. This approach provides more smooth exploration in solution space and gives opportunity to reach more optimal results by searching the neighborhood of the solution.

In the experimental phase, we ran a set of experiments with different values of the network characteristics (such as BS, BSC, MSC sizes) and solution parameters (such as local search number, crossover rate, population size, etc.), compared the performance of the proposed solutions. MA gave the best performance followed by MOGA and GA. Due to strict constraints on the network configurations, running times of the algorithms are long, which is between 12 to 24 hours on a P4 2.00GHz machine. MA was the most CPU consuming algorithm but since our study is an offline study (we run the algorithm once and then construct the network), this consumption is tolerable.

In our next chapter, we introduce the general concept of location area management, which also includes the related work. The detailed information on evolutionary algorithms is given in Chapter 3. Chapter 4 provides our framework for designing and implementing EA-based solutions, for location area management problem. The structure, definitions and pseudo codes are given in the same chapter.

2. LOCATION AREA MANAGEMENT

Location management plays an important role in cellular networks and it deals with following a mobile user in a network [10]. The two main operations of location management are registration and paging. Cellular network performs the paging operation. When an incoming call arrives to mobile station, the network searches the mobile station in all possible cells to find out the cell on which the mobile station is located, so the call can be routed to the corresponding base station. This search operation is called paging. The registration operation is performed by the mobile station. When a mobile station changes its location, it updates its location information on the system. The number of all possible cells to be paged depends on the performance of the registration of mobile stations.

So in a mobile communication system, on the arrival of an incoming call to a wireless customer, the system searches the customer's mobile station by paging it from different base stations following the strategy of looking up the current location information of the mobile. This information is managed by a registration procedure in which mobile stations report their presence whenever entering a new location area. A location area is a set of base stations, which are paged simultaneously, when a mobile station receives an arriving call. Location areas can be stored in databases and this information is used whenever an incoming call arrives to a mobile. The size of location areas is important because of the cost of paging and registration.

In Figure 2.1 a 4X4 network is divided into five location areas. First location area has two cells, second one has five cells, third location area has three cells, fourth has four cells and the last location area has two cells. This configuration is one of the alternatives from the solution space. Both number of location areas and the number of cells in one location area are changeable.

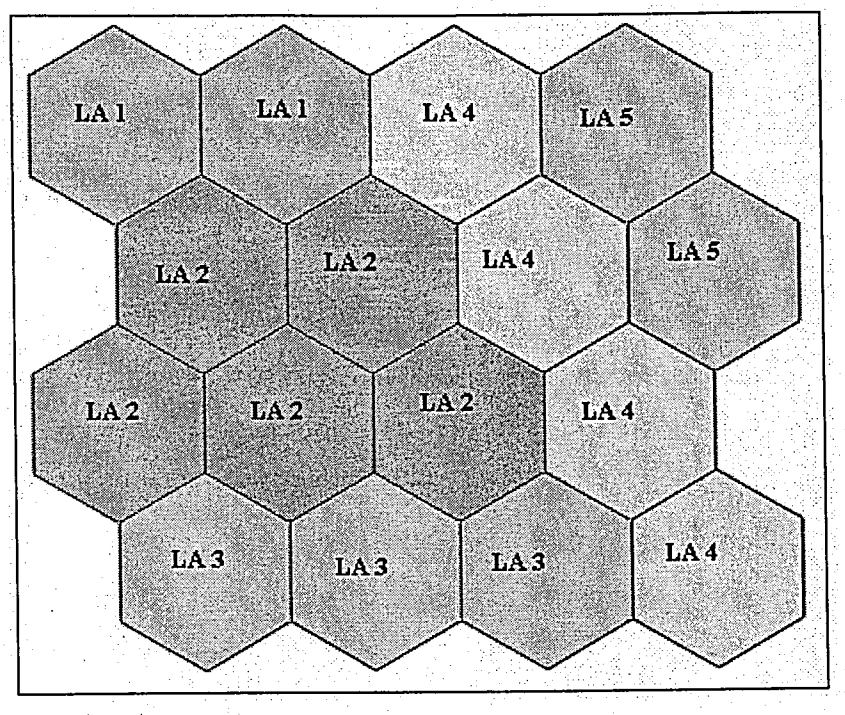

Figure 2.1. A sample network with 16 base stations divided into location areas

The schemes for location area management are categorized in the following five main groups [11,12,13,14,15,16]:

- **Time-Based Scheme:** In the time-based scheme, the location update is performed according to an elapsed time from the last location update. When a mobile station is needed to be searched, all possible cells, which the user can move within the elapsed time, are tracked. The disadvantage of this scheme is missing some cells according to the threshold time.
- **Movement-Based Scheme:** In the movement-based scheme, the number of cell boundaries that the mobile station crosses is counted as a measure of location update. The disadvantage of this scheme is the unnecessary updates, even though the location of the mobile user does not change.
- **Distance-Based Scheme:** In the distance-based scheme, the distance from the last updated location is measured and a new update is applied if a specified threshold is exceeded. The drawback of this scheme is that although a user does not receive an incoming call, and if he crosses the specified distance, the update is performed.

- **Profile-Based Scheme:** In the profile-based scheme, some specific criteria decide the location update. An example of these specific criteria is users' (mobile stations') characteristics.
- **Zone-Based Scheme:** Zone-based scheme, improves the drawbacks by reducing the redundant updates. It is based on updating the location if the mobile user enters an area (location area) where there is a high residing probability. In zone-based scheme the cellular network is divided into location areas. If the mobile station changes its location area, the registration operation is applied. Whenever an incoming call arrives, the mobile user is paged in its current location area. Zone-based schemes are widely used in Global System for Mobile Communication (GSM). There are two simple location area techniques. These are always-update strategy and never-update strategy. In the always-update strategy, each base station is a location area. So the mobile stations perform a registration whenever they enter a new cell. In this condition the registration cost is maximum but there is no need to a paging operation when a new call arrives.

In the never-update strategy, there is only one location area for the entire network, so there won't be a registration cost. Although the registration cost is minimum in never-update strategy, when a call arrives the mobile user is searched in all the cells. This maximizes the paging cost. These two strategies are the two extremes of location area management. Most of the existing cellular systems use the combining of these two simple techniques for minimizing their total cost. Figure 2.2 shows a 3X3 network, which uses always-update strategy, where each base station is a location area. Figure 2.3 shows same network with never-update strategy, where all base stations forms a single location area.

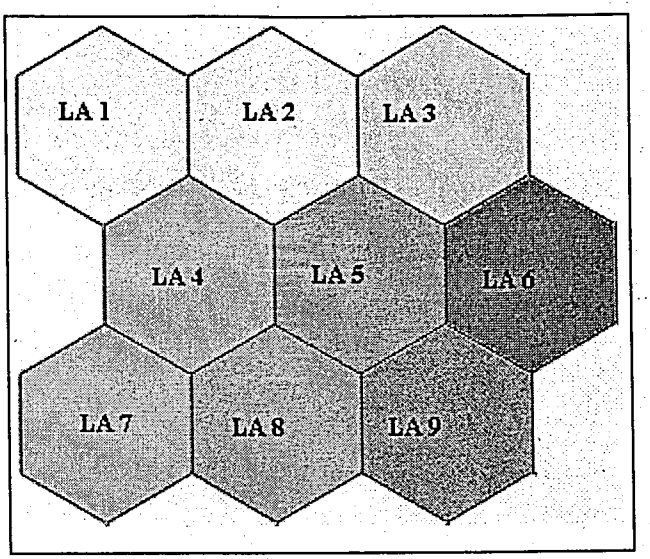

Figure 2.2. A network which uses always-update strategy

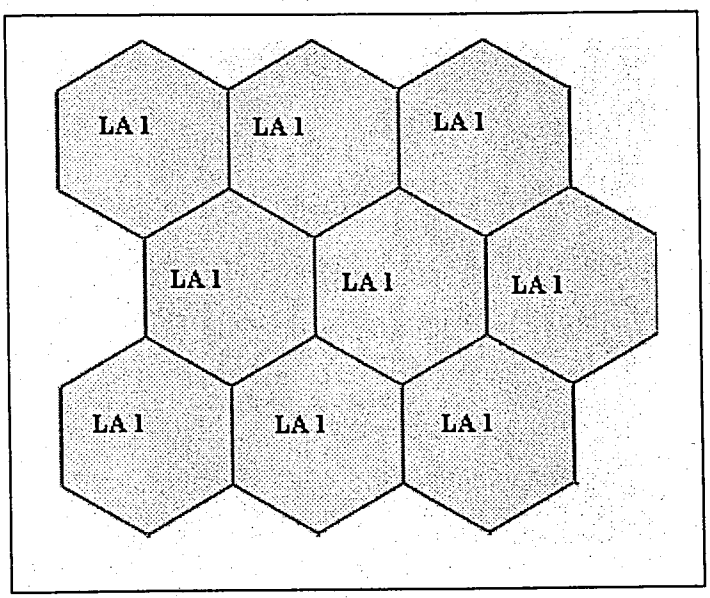

Figure 2.3. A network which uses never-update strategy

2.1. Problem Specification

Purpose of this thesis is providing a system for locating given number of base stations on a network. This network contains mobile services switching centers for managing base station controllers, base station controllers for managing location areas and these location areas contain any number of base stations. The main objective is reducing the cost of paging and registration signaling and the goal is to place base stations in optimum location areas, to place location areas into optimum base station controllers and

to place base station controllers into optimum mobile services switching centers. So the objective function requires addition of these two costs: registration and paging.

The network is partitioned into small areas called cells and these cells are combined to create a location area. Each location area is related to a base station controller and each base station controller is related to a mobile services switching center. So when a call arrives to a mobile services switching center, MSC knows the BSC information of the mobile user and BSC knows the location area information of the mobile user and BSC searches the mobile user in the base stations, which belongs to that location area. So partitioning a network into location areas results in paging fewer base stations. Also, when mobile users pass from one location area to another location area, their location information must be updated. If there is no location area in a network, each boundary passing from a base station to another base station causes a location update. But if there are location areas, the location update occurs whenever the location area information of a mobile user changes.

There are some constraints for assigning base stations to location areas, location areas to base station controllers and base station controllers to mobile services switching centers. All base stations in a location area must be adjacent to each other. Also the location areas in base station controllers must be adjacent to each other. The limitations of the number of location areas and base station controllers are other constraints. There must be at least one location area in a given network (which contains all the cells). On the other hand, the maximum number of location areas that can exist in a given network is equal to the number of cells in that network. The same constraints are applicable to base station controllers. The maximum number of base station controllers that can exist in a network is equal to the number of location areas in a network. The minimum number of base station controllers is equal to 1. Lastly, maximum number of mobile services switching centers that can exist in a network is equal to the number of base station controllers in a network. The minimum number of mobile services switching center equals to 1.

An example network structure is given in Figure 2.4. In this network there is a 4X4 network, with 16 base stations. The network is designed to have 5 location areas and 3 base station controllers and 1 MSC. First and fifth location area has two base stations, second

location area has five base stations, third location area has three base stations and fourth location area has four base stations. Two base station controllers (first and second BSCs) have two location areas. Third base station controller has only one location area.

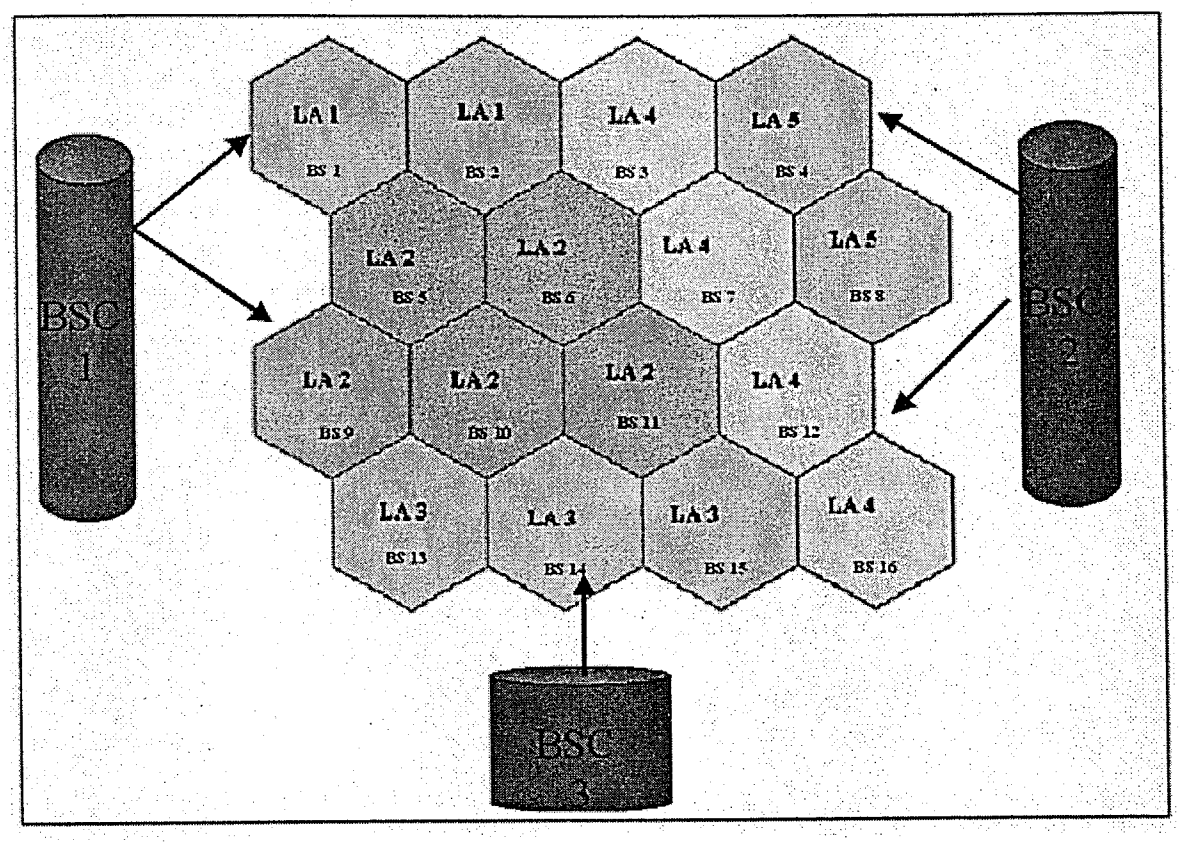

Figure 2.4. Example network structure of the problem

2.2. Formulations of the Problem

Our model for GSM consists of base stations (BSs), base station controllers (BSCs) and mobile services switching centers (MSCs). Base stations are grouped into location areas (LAs) and assigned to switches (BSCs and MSCs). The objective function and constraints given below are based on the model presented in [5].

The main difference in our model is to provide an objective function which requires the addition of two distinct cost types: paging cost and registration cost. Because of this addition, multi-objective genetic algorithms (MOGA) can be applied to this problem easily.

As part of the constraints and objective function, the following network parameters are used in our model.

λ_i: The paging rate per unit time for each base station in network.

c_i : call traffic rate for each cell i.

d_i : peak call attempt rate of cell i per unit time.

r_i : #of TRXs in each cell i.

The parameters used for representing the capacities of BS, BSC and MSC are the followings:

P_i^{BS}: The paging capacity of each base station.

P_i^{BSC}: The paging capacity of each base station controller.

C_j^{BSC} : Maximum call traffic capacity of each BSC.

C_k^{MSC} : Maximum call traffic capacity of each MSC.

D_j^{BSC} : Busy hour call attempt capacity of each BSC.

D_k^{MSC} : Busy hour call attempt capacity of each MSC.

R_j^{BSC} : Maximum number of TRXs for each BSC.

R_k^{MSC} : Mmaximum number of TRXs for each MSC.

The following design variables are used in our formulations:

x_{ij}: x_{ij} returns 1 if i^{th} cell is a member of j^{th} BSC, otherwise 0.

y_{jk}: y_{jk} returns 1 if j^{th} BSC is a member of k^{th} MSC, otherwise 0.

d_{is}: A function that returns 1 if i^{th} base station and s^{th} base station resides in the same base station. 0 otherwise.

l_{in} : A function that returns 1 if if i^{th} base station resides in if n^{th} location area, 0 otherwise.

As mentioned before, our objective function is given with the addition of paging and registration costs with the corresponding weights. In other words:

$$C(x) = P.C_p(x) + R.C_r(x) \quad (2.1)$$

where $C_p(x)$ is the paging cost and $C_r(x)$ is the registration cost. P and R are the coefficients of the costs which are equal to 10 and 1 in the experiments, respectively.

The paging cost on a cell is total of paging rates of the cells that belong to the same location area with that cell. This value is calculated by:

$$C_p(x) = P. \sum_i \lambda(1 - d_{is}) \quad (2.2)$$

The registration events are happened upon boundary crossing. Each time a user leaves a location area and enters a new one, it sends a registration message on the uplink control channel. Therefore the rate at which mobile users cross a boundary determines the cost of registration signaling. This cost is calculated by:

$$C_r(x) = R. \sum_i q(i) \quad (2.3)$$

Where

q(i): rate at which users pass by location i. (handover rate)

Therefore, the objective function is re-written as:

$$\min_i \; P. \sum_i \lambda(1 - dis) + R. \sum_i q(i) \quad (2.4)$$

The following set of constraints completes the model for location area management problem considered in our algorithms:

- **Paging Capacity of BS:** Maximum number of paging for one base station in one time slot.

$$\lambda_i < P_i^{BS}, \forall I \quad (2.5)$$

- **Paging Capacity of BSC:** Maximum number of paging for one base station controller in one time slot.

$$\sum_i x_{ij}\lambda_i < P_j^{BSC}, \forall j \tag{2.6}$$

- **Call Traffic Capacity of BSC:** Maximum number of calls for one base station controller in one time slot.

$$\sum_i x_{ij}c_i < C_j^{BSC}, \forall j \tag{2.7}$$

- **Call Traffic Capacity of MSC:** Maximum number of calls for one mobile services switching center in one time slot.

$$\sum_j \sum_i x_{ij}y_{jk}c_i < C_k^{MSC}, \forall k \tag{2.8}$$

- **BHCA Capacity of BSC:** BHCA stands for busy hour call attempt rate. In order to have a feasible cellular network, the limited call processing capability of base station controllers may create a limit on the peak call arrival rate. This call arrival rate includes not only established connections but also the failed attempts.

$$\sum_i x_{ij}d_i < D_j^{BSC}, \forall j \tag{2.9}$$

- **BHCA Capacity of MSC:** In order to have a feasible cellular network, the limited call processing capability mobile services switching centers may create a limit on the peak call arrival rate. This call arrival rate includes not only established connections but also the failed attempts.

$$\sum_j \sum_i x_{ij}.y_{jk}d_i < D_k^{MSC}, \forall k \tag{2.10}$$

- **TRX Capacity of BSC:** TRX stands for transmitters. Each base station controller has finite number of transmitters, which defines the number of channels used in that cell.

$$\sum_i x_{ij} r_i < R_j^{BSC}, \forall j \tag{2.11}$$

- **TRX Capacity of MSC:** Each mobile service switching center has finite number of transmitters, which defines the number of channels used in that cell.

$$\sum_j \sum_i x_{ij} y_{jk} r_i < R_k^{MSC}, \forall k \tag{2.12}$$

- Each BS should be assigned to exactly one BSC. So for the i^{th} cell, BS-BSC topology matrix should have only one entry in the i^{th} row that has value 1 and others must be 0.

$$\sum_j x_{ij} = 1, \forall i \tag{2.13}$$

- Each BSC should be assigned to exactly one MSC. So, for j_{th} BSC, BSC-MSC topology matrix should have only one entry in the j_{th} row that has value 1 and others must be 0.

$$\sum_k y_{jk} = 1, \forall j \tag{2.14}$$

- Each BS should be assigned to exactly one LA. So, for the i_{th} cell, BS-LA topology matrix should have only one entry in the i_{th} row that has value 1 and others must be 0.

$$\sum_n l_{in} = 1, \forall i \tag{2.15}$$

2.3. Related Work

Optimal location area planning for zone-based schemas is not a widely studied problem. There are only a few important research works, which are briefly summarized in the following paragraphs:

Saraydar *et al.* [17] has proposed one-dimensional location area design. In their work, they consider a linear service area like a highway, which joins cities or railway lines. In this kind of question they aim to divide the highway into location areas, along its length so that paging and registration cost is minimized. In their problem, the mobile stations are vehicles or pedestrians. Their purpose was to find the location area boundaries that minimize a specific objective function of the paging and registration traffic. Paging cost for a location area in their proposal equals to the sum of costs of roaming rate of each user in that location area. There can be one or more cells in a location area and a paging request causes a paging signal to spread to each cell, signaling cost is proportional to the number of cells. So the paging cost is proportional to the cost of each unit of paging signal, average call arrival rate of each user, total number of mobile users in one area and overall signaling in areas. Cost of registration is related to the boundary crossings. When a mobile user leaves a location area and enters the new location area, the mobile user sends a registration message. The registration cost is calculated by using the rate at which mobile users cross a boundary. So rate at which users pass by location x and cost of each registration signal are used to calculate total cost of registration. They use three approaches, which are direct approach, average local lengths as a heuristic method and asymptotic location area density.

Demirkol *et al.* [5,12,18] have proposed an algorithm which uses a Simulated Annealing based approach to find optimal network structure in between the two extreme cases which are zero paging cost and zero registration cost cases. They inherited the Mobile Switching Center (MSC), Base Station Controller (BSC), and Base Station (BS) architecture from the Global System for the Mobile communication (GSM) standards. Also they have included the constraints such as call arrival rate, number of transmitters per base station, etc. In their experimental phase, they have compared the performance of solutions generated by simulated annealing approach with pure randomly generated solutions and solutions found by performing a greedy search. Finally they have concluded that the

simulated annealing approach gives more appropriate network structures than those given by greedy search and random generation.

Another approach to location area management has been proposed by Subrata and Zomaya [19]. In their work, they proposed three different techniques for determining the optimal reporting cell locations efficiently. Reporting cells are used to keep track of location of mobile user in the network. In case of an incoming call to the mobile station, system searches the neighborhood of the reporting cell that the user last reported. Finding the optimal reporting cell locations in a given network is also known to be NP-complete. Three methods, which they used in this research are genetic algorithms, tabu search and ant colony optimization. The brief explanations of them are given below:

- **Genetic Algorithms:** In their study, a population of 100 chromosomes is used. Each chromosome consists of binary string representing a reporting cell configuration. For an 8*8 network, the chromosome will have length of 64. Each binary gene, which represents a cell in the network, can have a value of either 1 or 0 representing reporting and non-reporting cells. In initialization phase, the configuration for each chromosome is randomly generated. The constraints for stopping criteria are max computation time, chromosomes' similarity and number of iterations/ generations.
- **Tabu Search:** A neighborhood structure is defined in this method. The neighborhood of a solution is defined to consist of a "0 to 1" move and a "1 to 0" move. If a cell is at state "0" then the only allowed move for that cell "0 to 1", or if a cell is at state "1" the allowed move is "1 to 0".
- **Ant Colony Optimization:** In ant colony optimization, they initialize population and pheromone, then for each ant a solution is generated and the pheromone is laid. The pheromone is updated until the escape condition is met. According to their experimental study, the tabu search shows a better performance than the rest.

Another approach for location area management from Subrata and Zomaya is evolving cellular automata [20]. They also approached to the problem from the reporting cell point of view. They used cellular automata combined with genetic algorithms in order to come up with an evolving parallel reporting cells planning algorithm. In their approach,

cells in the network are mapped to cellular units of the cellular automata and a genetic algorithm is used to achieve efficient cellular automata transition rules in a selected cellular automata neighborhood. A fixed-length cellular automata with assumption that it is wrapped around the edges, is used. Each unit in the cellular automata is associated with a cell in the network. Since it is assumed that the cells in the network are hexagonal, each cell can have a maximum of six neighbors. Also, each cell can be either a reporting cell or not; so, each automaton has two states (either 1 or 0). For the cases that a cell has less than six neighbors, dummy neighbors are added to fix the number of neighbors. Then, suitable cellular automata rules are discovered by using a genetic algorithm. They concluded that the genetic algorithms can be effectively used to discover suitable cellular automata rules for reporting cells problem.

Quintero and Pierre have proposed a multi-population memetic algorithm for assigning cells to switches in a mobile network [21,22]. For the cost calculations, two types of handoffs are defined. A *handoff* is defined as transferring a call from one radio channel to another if the user moves from one cell to another since adjacent cells uses different radio frequencies for communication. If a handoff occurs between two cells belonging to same base station controller, it is called as a s*imple handoff*. On the other hand, if the handoff occurs between two cells belonging to different base station controllers, it is called as a *complex handoff*. The cost function consists of total simple handoff cost, total complex handoff cost and the total link cost. They used a multi population memetic algorithm with migration and two local refinement algorithms, tabu search and simulated annealing as local refinement algorithms. The results presented in their paper confirmed the efficiency of multi population memetic algorithms for large sized cellular networks.

3. EVOLUTIONARY ALGORITHMS

3.1. Genetic Algorithms

Genetic Algorithms (GA) [7,23,24,25] are random search techniques, which are based on random natural selection and genetics. They find efficient solutions by combining survival of the fittest among string structures. String structures represent different solutions and in every generation, a new set of strings is created by using the old generations and also bits for finding the best solution in the generation. In genetic algorithms, historical information is used to find better results. The solution space, in which the genetic algorithms are used are called "populations" and each solution is defined as an "individual". A "chromosome" represents a solution in a solution space. The goal of genetic algorithms is optimizing a function or a process.

Genetic Algorithms perform well in many different types of problems. There are several difference, between genetic algorithms and the other search techniques. These are [7]:

- Genetic algorithms do not use the parameters but they use the coding of parameters.
- Genetic algorithms do not search a single point but they search a population of points.
- Genetic algorithms do not use derivatives but they use objective functions.
- Genetic algorithms do not use deterministic rules but they use probabilistic transition rules.

A genetic algorithm requires three main operators, selection, crossover and mutation, which are explained below:

- **Selection:** Individual strings are allowed to pass to the next generation according to the results of selection operation [26]. String's fitness value, which is calculated from its fitness function affects this operation. Fitness function is the measure of quality. The strings are placed in a pool and the reproduction operator chooses the string for the next generation from this pool.

There can be different types of selection operators. An easy way for reproduction operator is always selecting the fittest strings and ignoring the weakest strings. There are so many researches, which use this method. It can give good results according to the problem domain [27]. The roulette wheel method is one of the mostly used selection operator. The roulette wheel method uses a statistical approach based on strings' relative fitness values. In roulette wheel selection, highly fit individuals have a higher probability of being selected for further processing.

- **Crossover:** As in the biological meaning, the crossover operator blends the chromosomes (solutions) from the parents to produce new chromosomes for the offspring. The genetic algorithm selects two strings according to selection rules. The strings, which are selected, can be different or identical. Then it is decided that if there will be a crossover or not according to a parameter called crossover probability. This can be a simple probability value and this value can be set by the user. If the genetic algorithm decides not to perform a crossover by controlling the crossover probability parameter, two selected strings are copied to the next generation. If the crossover is decided to perform, traditionally, crossover operator randomly chooses a crossover cite and cuts the strings into two substrings, and swaps the tail substrings to produce new chromosomes.
- **Mutation:** After selection and crossover, different strings can be generated, but there may not be enough variety of strings to ensure the genetic algorithm sees the entire problem space according to the initial population. Another possibility is, genetic algorithms can converge to strings which are not close to the optimum solutions. Usually, the mutation operation is used to prevent being stuck on a local optima. The probability of occurrence of mutation operation is determined by the mutation probability parameter. There are various different and problem specific mutation methods. The most widely used mutation method is single bit mutation, where a randomly selected bit of the string is mutated by taking one's complement of the bit (the bit is set to 0 if it is currently 1, and is set to 1 if it is currently 0). An example pseudo code of genetic algorithm is given Figure 3.1.
-

```
Procedure Genetic Algorithm( )
{
t = 0
initialize P(t)
evaluate P(t)
While (termination condition false)  do {
                    select P( t+1)
                    recombine P(t)
                    evaluate P(t+1)
                    t=t + 1  }
}
```

Figure 3.1. Structure of genetic algorithms

3.2. Multi Objective Genetic Algorithms

In multi objective genetic algorithm approach, one population of solutions is created for each objective. Blickle [8] suggested the main principles on multi objective optimization. The population is divided into disjoint sub-populations, where each population is sub-population optimized with its own objective. Then, best results from sub-populations are selected for genetic algorithm step and this elite population is optimized for objective set.

If there are n objectives in a problem, first, GA is applied to n different populations for their different objective functions. After sub-populations create their best results, an elitist strategy is applied and best resultants of n populations create one population for applying genetic algorithm. For our problem, there are two objectives: Reducing the registration cost and reducing the paging cost. So that GA runs for these costs first.

3.3. Memetic Algorithms

Memetic algorithms [9,28,29] are also known as knowledge-augmented genetic algorithms or hybrid genetic algorithms. The main difference between memetic algorithms and genetic algorithms is genetic algorithms are concerned with biological evolution; however memetic algorithms are concerned with mimicking cultural evolution. In memetic algorithms, genetic algorithms are used with population-based global search. As a characteristic, memetic algorithms combine global and local search.

In memetic algorithms, local search techniques can be applied to various phases of evolutionary cycle, including initial population generation, selection, crossover and mutation phases. (See Figure 3.2)

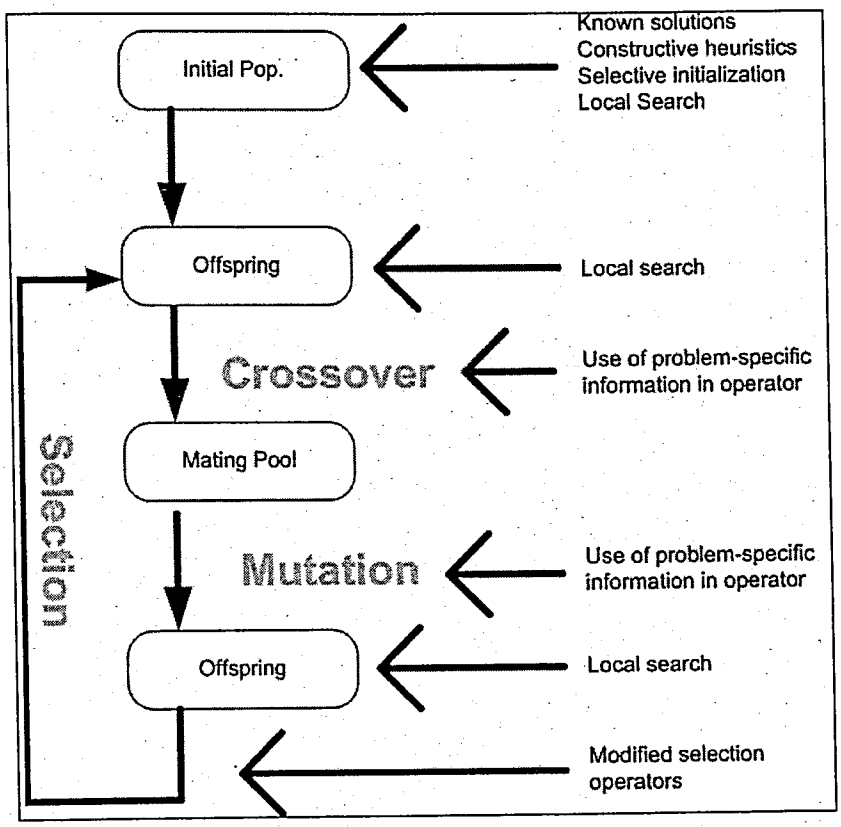

Figure 3.2. Evolutionary cycle of memetic algorithm [30]

The first phase to apply the local search techniques is the initial population generation phase. When individuals have reached an acceptable development as part of initialization, they interact with each other by genetic algorithms.

The evaluation of the initial population are done in the same way with genetic algorithms. After initial evaluation, each individual in the population are processed by a local search engine which tries to increase fitness values of the individuals by making them tessellate in the vicinity of their current position in the solution space. Then, the iterative process begins and the individuals in the population evolve through the generations until a termination condition (in general, amount of change in the convergence between two generations is below a certain threshold, the fittest individual has a fitness value over a certain threshold or a certain number of generations has been exceeded) is satisfied. In this iterative process, a number of recombinations and a number of mutations are determined

then determined number of recombinations and mutations are applied on population consecutively.

```
Begin
        InitializePopulation
        For each individual i in population
                Evaluate(i)
        For each individual i
                Local search engine(i)
        Repeat
                For j = 1 to recombinations
                        Select to merge a set S_mer ⊂ Population
                        Offspring = Recombine (S_mer)
                        Evaluate(offspring)
                        Add evaluated offspring to Pop
                End For
                For j=1 to #mutations do
                        Select to mutate an individual i from Pop
                        i =Mutate(i)
                        i = Local Search Engine (i)
                        Evaluate(i)
                        Put the individual i back to the Pop
                End For
                Pop = SelectPop(Pop)
                If Pop has converged then Pop= RestartPop(Pop)
        Until termination condition true
End
```

Figure 3.3. The local search based memetic algorithm

In recombination, first a subset (has a size of 2) of the population is selected for mating. Then, the crossover operator is applied on individuals in the selected subset. Finally the offspring are evaluated and added to new population. This process is repeated until a predefined number of recombinations are completed.In mutation process, a predefined number of individuals are mutated by selecting a random individual, mutating it according to a defined mutation rule and performing a local search on resultant individual in order to increase the fitness value.

When the recombination and mutation processes are over, the population to be used in the next generation is selected by performing a selection on the resultant population of recombination and mutation processes. The aim of this selection process is to reduce the

population size to some degree while maintaining some certain level of diversity within the population. If the diversity within the population is below a certain level, it probably means that the system is converged on a local optimum before it reaches none of its termination conditions. In such case, the population is re-shuffled and the process is started over.

4. EVOLUTIONARY ALGORITHMS FOR LOCATION AREA MANAGEMENT

In this chapter, our approach to location area management problem for minimizing the registration and paging costs is examined in detail.

4.1. Structure of the Network

The network contains base stations and each base station belongs to one location area, each location area belongs to one base station controller and each base station controller belongs to one mobile services switching center. For constructing this kind of a structure pointer arrays are used. There will be an array for holding all the solution space for our problem. Each solution points another array, which holds the information of mobile services switching center of this solution. Mobile services switching centers points to base station controllers array. Base station controllers array points another array. This array is the location area array. Location area array holds the information about location areas and also points to base station array of each location area contains. (Figure 4.1)

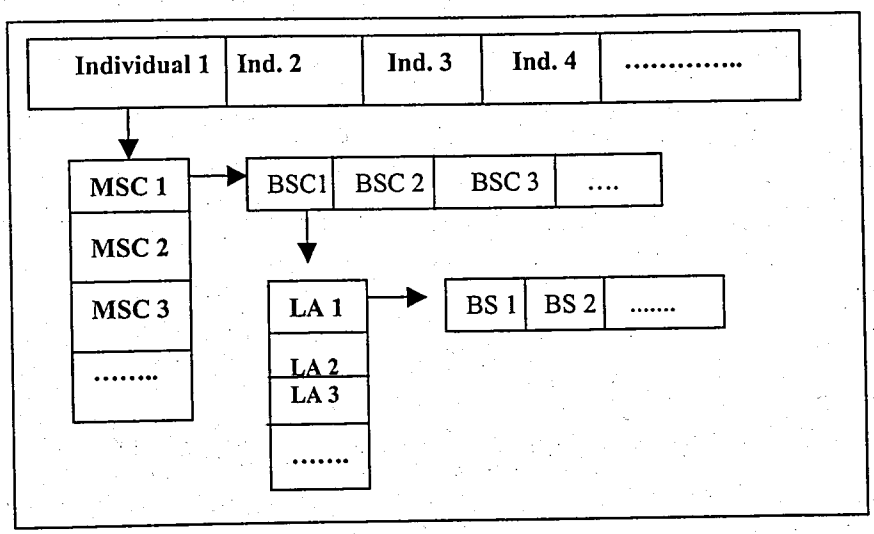

Figure 4.1. Structure of a network

The fields of the data structure of a given network are given in Figure 4.2.

```
struct solution
{
        int **LAarray;
        int **BSCarray;
        int **MSCarray;
        int *BSarray;
        int LAsize;
        int BSCSize;
        int MSCSize;
        int *LAsizeForBSC;
        int *BSCsizeForMSC;
        float fitnessValue[EVALUATIONNUMBER+1];
        int *Capacity[CAPACITY_NUMBER_BSC];
        int Capacity_BS[NETWORKSIZE];
        int *CapacityMSC[CAPACITY_NUMBER_MSC];
}population[POPULATIONSIZE];
```

Figure 4.2. Description of the structure

Detailed explanations of the variables are given below:

- **LAarray:** LAarray holds the information of the location areas in the network. It is a dynamic array because the number of location areas for one solution is undetermined at the initialization. All elements of this variable point an array of base stations, which belongs to the element.
- **BSCarray:** BSCarray is planned to be a dynamic array, too. Because its size can be change in a range of numbers. It holds each base station controller. This variable's elements point the location areas of their own.
- **MSCarray:** MSCarray is a dynamic array, too. Because its size can be change in a range of numbers. It holds each mobile services switching center. This variable's elements point the base station controllers of their own.
- **BSarray:** This variable is created for holding every base station of one location area. Every element of the LAarray variable points one BSarray. A pointer structure is suitable for BSarray because the number in any location area is not specific at the beginning.
- **LAsize:** The LAarray is planned to be dynamic. So another variable is used for holding the information of size of the LAarray. The number of location areas in one solution can be known with this variable. This simplifies the coding part of the thesis.
- **BSCsize:** BSCsize holds the size of the BSCarray. The number of base station controllers is specific with holding this information and can be used in code.

- **MSCsize:** MSCsize holds the size of the MSCarray. The number of mobile services switching centers is specific with holding this information and can be used in code.
- **LAsizeForBSC:** This dynamic array holds the number of location areas in each base station controller. So distribution of the location areas to base station controllers can be controlled.
- **BSCsizeForMSC:** This dynamic array holds the number of base station controllers in each mobile services switching center.
- **FitnessValue:** FitnessValue is an array that holds the fitness value information of each solution. It is an array because a solution can be evaluated more than once.
- **Capacity:** As it is told before, there are some constraints in our approach. Capacity is an array for these constraints that is related with the base station controllers. All base station controller constraint values are hold in this array.
- **Capacity_BS:** All constraints, which are concerned with base stations, are hold in this variable.
- **Capacity_MSC:** All constraints, which are concerned with MSCs, are hold in this array.

Some constants are defined at the beginning of the program. The explanations of the constants are given below:

- **POPULATIONSIZE:** The constant "population size" holds the number of the solutions in a population.
- **NETWORKSIZE:** The number base stations in solutions are hold in "network size" constant.
- **EVALUATIONNUMBER:** The constant "evaluation number" holds the information of how many times a solution is evaluated.
- **CROSSNUMBER:** The constant "cross number" holds the information of how many crossovers will be applied to a solution.
- **MUTATIONRATIO:** The probability of mutation for mutation process is held in "mutation ratio" constant.
- **CROSSRATE:** The probability of crossover for crossover process is held in "cross rate" constant.

- **CAPACITY_NUMBER_BSC:** This constant holds the number of capacities that is related with base station controllers. It is used in the struct's "Capacity" array for determining and assigning the values of the capacities.
- **CAPACITY_NUMBER_BS:** The number capacities, which are related with the base stations is hold in this constant.
- **CAPACITY_NUMBER_MSC:** The number capacities, which are related with the mobile services switching centers is hold in this constant.
- **PAGINGCOST:** The value of the paging cost is kept in this constant. In our approach it is assumed that paging cost is 10 times bigger than registration cost.
- **REGISTRATIONCOST**: Registration cost's value is kept in this constant.
- **ELITNUMBER:** An elitist strategy is used in selection process of the algorithms. This constant holds the information of the number of elitist solutions in one process. So these solutions can be selected for the next generation.
- **GENERATIONNUMBER:** This constant holds the value of the number of solution generations in one run.

Table 4.1 gives two example solutions of a 3 X 3 network, which are also graphically given in Figure 4.3. First solution has seven location areas, these location areas are located into four base station controllers and these base station controllers are placed into two mobile services switching centers. In the second example there are four location areas for nine base stations and four base station controllers for locating these four location areas. Three mobile services switching centers are used for placing location areas. The number of base stations is fixed at the beginning but the number of the location areas and base station controllers are determined at the run time. The vicinity of base stations is taking into account when locating the base stations into location areas. Same vicinity constraint is applicable for locating location areas into base station controllers.

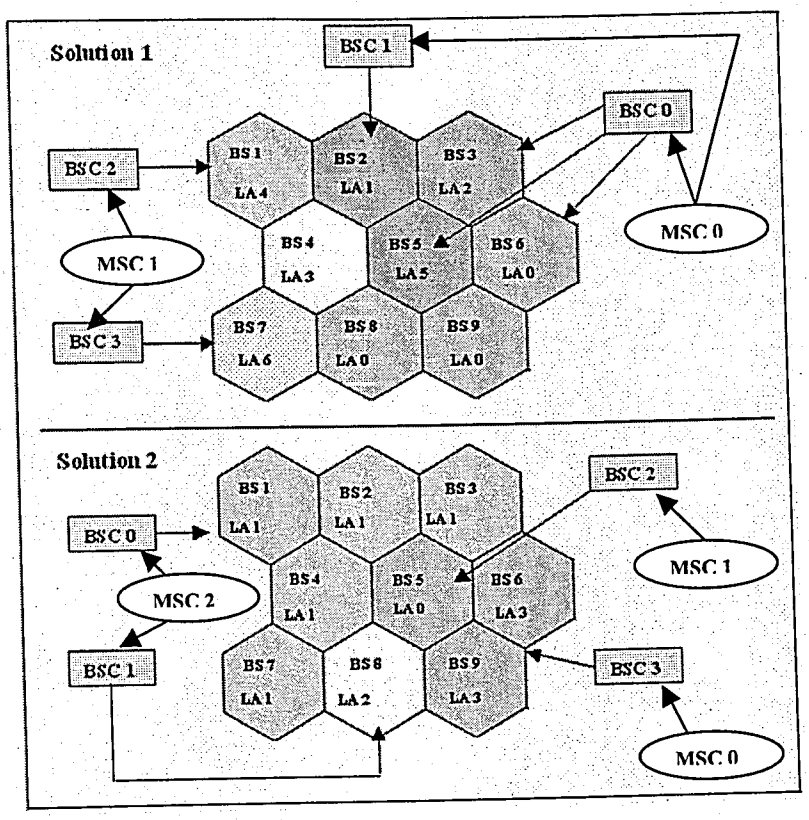

Figure 4.3. Two sample solutions of a 3X3 network

Table 4.1. Showing the sample solutions in a tabular form

Solution 1:

LA 0: BS_6 BS_9 BS_8 **LA 1:** BS_2 **LA 2:** BS_3 **LA 3:** BS_4 **LA 4:** BS_1 **LA 5:** BS_5 **LA 6:** BS_7

BSC 0: LA_5 LA_0 LA_2 LA_3 **BSC 1:** LA_1 **BSC 2:** LA_4 **BSC 3:** LA_6

MSC 0: BSC_0 BSC_1 **MSC 1:** BSC_2 BSC_3

Solution 2:

LA 0: BS_5 **LA 1:** BS_1 BS_2 BS_3 BS_4 BS_7 **LA 2:** BS_8 **LA 3:** BS_9 BS_6

BSC 0: LA_1 **BSC 1:** LA_2 **BSC 2:** LA_0 **BSC 3:** LA_3

MSC 0: BSC_3 **MSC 1:** BSC_2 **MSC 3:** BSC_0 BSC_1

The following three sections are dedicated for the implementation details of genetic algorithms, multi objective genetic algorithms and memetic algorithms, respectively.

4.2. Implementation Details of Genetic Algorithms

Although execution flow of genetic algorithms is straightforward, there are many problem specific implementation details can take place such as string representation, selection method, probabilities of crossover and mutation etc. The pseudo code for the genetic algorithm for our approach is given in Figure 4.4.

```
Procedure GA_Location_Area_Management( )
{
        initialize population
        evaluate population
        Add elitist individuals to next population
       While (generationNumber<maxGenerationNumber)  do {
                select two individuals
                if crossover probability occurs  {
                                crossover two individuals
                               if mutation probability occurs {
                                        mutate individuals
                }
                Add offsprings to next population
        } }
```

Figure 4.4. The pseudo code of GA for location area management

In our genetic algorithm approach, the solution space is generated. Then, solutions are evaluated and fitness values of each evaluation are calculated. According to elitist strategy, a set of solutions, with the best results, are passed to the next generation and then two parents are selected for crossover step. If crossover probability is formed, they enter into the crossover process. If mutation probability is constituted for each solution, these solutions are mutated. These steps are executed until the stopping conditions met. Another important point is validation. After initialization, crossover and mutation validation is processed for catching the initialization criteria. The details of the genetic algorithm are given in following subsections.

4.2.1. Initialization Phase

As part of the initialization phase, first the number of location areas is determined randomly. The base stations in the network are placed in these location areas. Then, the location areas are placed into the base station controllers randomly. Finally, the base station controllers are placed into mobile services switching centers.

Validation is the difficult part of the initialization. There are many constraints for assigning a base station to location area, a location area to base station controller and a base station controller to mobile services switching center. For example base stations in a location area must be neighbors. The details of assigning base stations to location areas are given below:

- In order to avoid having empty location areas, the number of location areas must not be bigger than the number of base stations in the network and there should be at least one location area.
- There is an exception for the last location area. Because all the base stations must be located in one location area, the rest of the base stations must be located in this location area. So the number of base stations in the last location area is equal to the number of base stations left.
- In order to determine the number of base stations, there should be at least one base station left for each of the remaining location areas.
- Another array is filled when assigning a base station to location area, when the neighbors of the assigned base station are put into this array. A new selection is made from this array.
- Number of transmitters for each base station is assigned.

The pseudo code for base station-to- location area part of the initialization is given in Figure 4.5.

```
Do for all the populations (i);
        Find the number of location areas randomly;
        Assign this number to population[i].Lanumber;
        Do for all Location areas (j);
            If the last location area is located
                Find the BS number for the LA randomly
                Do for all BSs in this location area
                    If the first BS will be located
                        Find a BS that isn't located any other location area
                        Locate it to this location area with its random # of transmitters
                    Find its neighbors and keep them
                    Else (if it's not the first BS)
                        If there is only one LA
                          Locate all the BSs to this LA with their random # of transmitters
                    Else (if it is not only one LA)
                                Find BS who isn't located other LA & neighbor to other BSs in LA.
                                Locate it to this location area with its random # of transmitters
                                Find its neighbors and keep them.
            Else (if not the last location area is located)
                Find the BS number for the LA randomly
                Do for all BSs in this location area
                If the first BS will be located
                        Find a BS that isn't located any other location area
                        Locate it to this location area with its random # of transmitters
        Find its neighbors and keep them
                Else (if it's not the first BS)
                        If there is only one LA
                                Locate all the BSs to this LA with their random # of transmitters
                Else (if it is not only one LA)
                                Find BS who isn't located other LA & neighbor to other BSs in LA.
                                Locate it to this location area
                                Find its neighbors and keep them.
                Keep the totalnumber of BSs.
End BS-to-LA.
```

Figure 4.5. Pseudo code for BS-to-LA assigments

The next phase is to assign base-station controllers (BSCs). The number of BSCs is not definite at the beginning of the initialization, and this number is related with the number of the location areas. Pseudo code for base station-to-base station controller is given in Figure 4.6.

```
Create Neighbor Array of all the location areas in the network
Find the number of base station controller randomly.
Assign this number to population[i].BSCnumber;
Calloc the population[i].BSCArray and LAsizeForBSC with this number;
Do for all base station controllers (j);
        If the last base station controller is located
          Find the LA number for the BSC randomly
          Do for all LAs in this location area
                If the first LA will be located
                        Find a LA that isn't located any other BSC
                        Locate it to this Base Station Controller
                Else (if it's not the first LA)
                        If there is only one BSC
                                Locate all the LAs to this BSC
                        Else (if it is not only one BSC)
                                Find LA who isn't located other BSC&neighbor to other LAs in BSC.
                                Locate it to this base station controller
        Else (if not the last base station controller is located)
          Find the LA number for the BSC randomly
          Do for all LAs in this BSC
                If the first LA will be located
                        Find a LA that isn't located any other BSC
                        Locate it to this BSC
                Else (if it's not the first LA)
                        If there is only one BSC
                                Locate all the LAs to this BSC
                        Else (if it is not only one BSC)
                                Find LA who isn't located other BSC &neighbor to other LAs in BSC.
                                Locate it to this BSC
        Keep the totalnumber of LAs.
End LA-to-BSC.
```

Figure 4.6. Pseudo code for BS-to-BSC assignments

The following cases are considered for BS-to-BSC assignments:

- The location areas, which are locating in the same base station controllers, must be neighbors.
- The number of base station controllers is assigned randomly. But this number must not be bigger than the location area number. This control is put to the program.
- If the last base station controller is filled then all the location areas left alone must be put to this base station controller. So the number of location areas of last base station controller is definite.
- There must still remain some free location areas when assigning number of location areas to a base station controller if there exists empty base station controllers. So the program has control for this condition.
- BHCA, Paging, Call Traffic Capacity are assigned for each base station controller.

The implementation details of assigning base station controllers to mobile services swithching centers are same with the the LA-to-BSC assignments.

4.2.2. Evaluation Phase

Evaluation phase runs after the values of parameters and constraints are set for a given experimental network. The incoming calls for each base station and movements from one location area to another are taken into account as a cost. Also the constraints such as BHCA, paging capacity are controlled.

Evaluation step runs "evaluation number" times ("evaluation number" is a constant that is we determine) for all population. These values are kept in each solution's structure. At the end of the step, average of these fitness values is calculated. Instead of one fitness value, the average of all fitness values is used in selection step.

4.2.3. Selection Phase

After calculating fitness values of each solution, selection step runs for determining which solutions passes to next generation and which solutions will enter the crossover and mutation step. In our approach an elitist strategy is followed. "Elitist number" value is assigned at the beginning of the program and best "elitist number" solutions passes to next generation. In our approach, "roulette wheel" method is used for selection process. In roulette wheel bigger fitness values have more chance for selection but our criteria is cost, and we want to select minimal cost given solutions for reproducing new generations. So 1/fitness value is used in selection process. The 1/fitness values are normalized to [0..1] range. So the values are located into this range. In each selection step two numbers between 0 and 1 is taken and the solutions, which are located in these numbers, are selected for reproduction.

4.2.4. Crossover Phase

Two selected solutions enter crossover process if crossover probability holds. There are many different ways of applying crossover such as base station based crossover or base station controller based crossover. We select a location area based crossover methodology. In location area based crossover method, a location area is selected and solutions change their location areas according to index of this location area. When determining the index of location area, the number of location areas in smaller solution is used. Index can be minimum 1 and maximum (number of location areas in smaller solution - 1). The number of location areas in each solution does not change after crossover, instead the number base stations and the base stations in location areas change.

Validation is the most difficult part in crossover. Three main validation problems may occur after the crossover operator:

- A base station may disappear after crossover. Because two solutions change their location areas one solution can give the part of itself, which does not have one base station.
- A base station may occur twice in structure.
- The location areas in base station controllers may lose their vicinity.

```
For every Base Station in the network do
        If LA number = 1
                Put all the Base Station to the same location area
        Else (If LA numbrt is not 1)
                Find how many times current BS occurs.
                If occurrence of current BS >1
                        Find both LAs that the BS occurs.
                        If both LA size !=1
                                Remove the BS from biggest LA.
                        Else
                                Add the BS to tempArray.
For every Base Station in the network do
        Find how many times current BS occurs
        If it doesn't occur any of the LAs
                Find the neighbors of the current BS in the LAs
                Locate the BS one of neighbor's location area, which has minimum BS
Make validation of the BSs in the temp array.
Make validation of the location areas that has un-neighbor BSs.
```

Figure 4.7. Validation phase after crossover operation is applied

A validation phase (given in Figure 4.7) is introduced in order to handle these three cases.

4.2.5. Mutation Phase

In mutation, first a location area is selected randomly in crossovered solutions. Then a base station in this location area is selected and a new base station is assigned to this location. After mutation a validation is needed, because there can be two same base stations in a solution and a base station will be disappeared after mutation.

4.3. Implementation Details of Multi Objective Genetic Algorithms

In multi objective genetic algorithm approach, more than one population are created, one for each objective. They are evaluated only for their objectives and best resultants are combined for evaluating in superior genetic algorithms. Blickle [8] first suggested the principle ideas of multi objective optimization for problems. Each fundamental idea was examined for strengths and weaknesses. The population is divided into disjoint sub-populations and, where each population is sub-population optimized its own objective. There are two main objectives in our problem. These are:

- Reducing the paging cost
- Reducing the registration cost

Figure 4.8 is the flowchart of the multi objective genetic algorithm applied on a location area management problem. As can be seen from the flowchart, processing multi objective genetic algorithm approach, two populations are created. One population is used to find best given fitness values for reducing paging cost. In that part, the fitness function will be:

$$C_p(x) = P. \sum_i \lambda(1 - dis) \qquad (4.1)$$

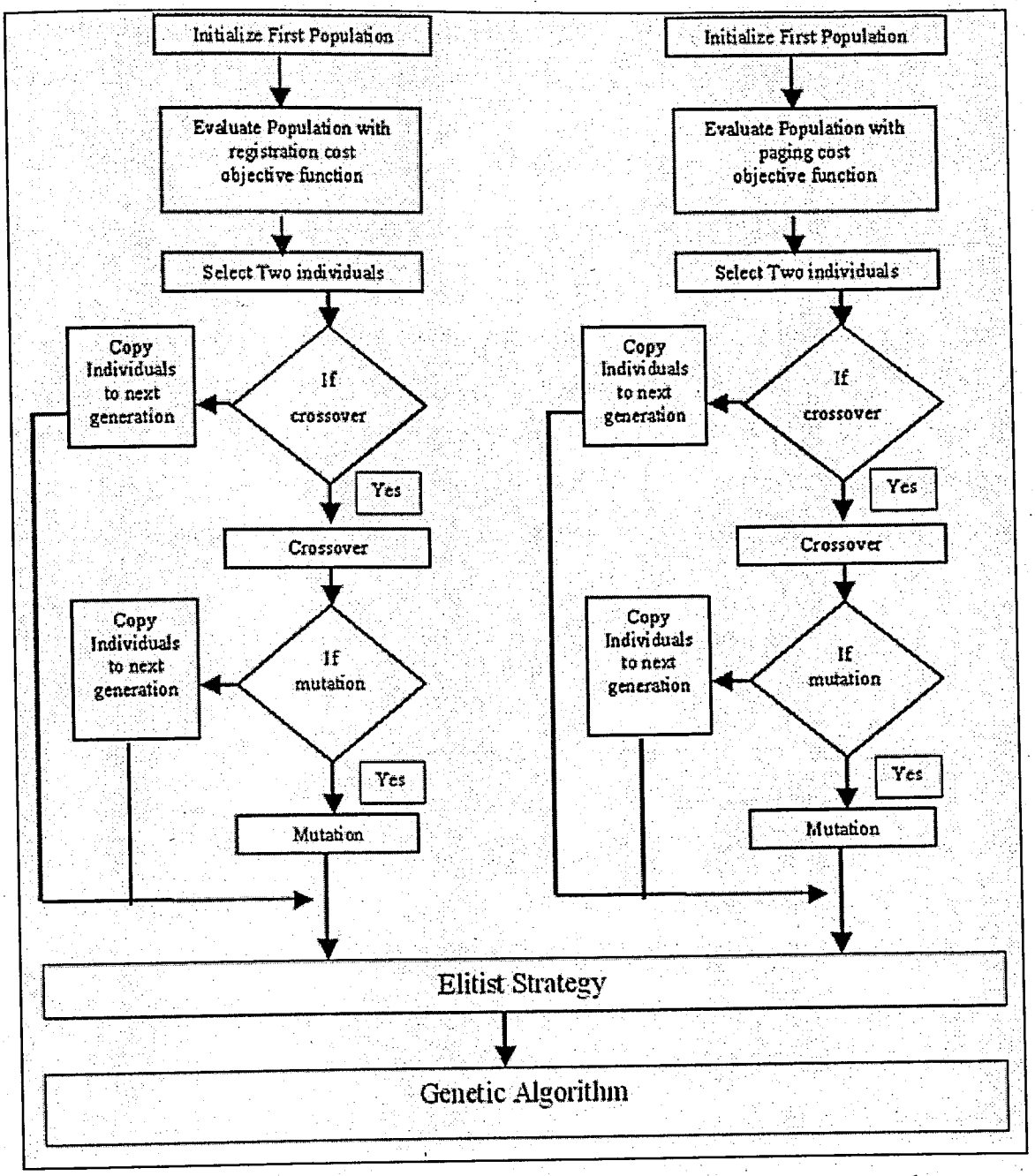

Figure 4.8. Flowchart of the multi objective genetic algorithm approach

Second population is used for finding best fitness values for reducing registration cost. In this part the fitness (objective) function will be:

$$C_r(x) = R. \sum_i q(i) \tag{4.2}$$

There are two objectives to be optimized. The population is divided into 2 sub-populations. These sub-populations are evolved, separately. Selection, crossover, mutation

are applied to each sub-population. After the sub-populations genetic algorithm process, an elitist strategy searches the solutions, which are generated by sub-population optimization. Best results are taken and inserted into genetic algorithm step.

4.4. Implementation Details of Memetic Algorithm

Memetic Algorithms, also known as knowledge-augmented genetic algorithms or hybrid genetic algorithms, are population-based heuristic search approaches for combinatorial optimization problems based on cultural evolution [9,28,31]. In memetic algorithms local search plays a significant role around the evolutionary algorithms. Memetic Algorithm is a marriage between a population based global search and the heuristic local search made by each of the individuals[28,29]. In the memetic algorithms approach genetic algorithms are combined with local search.

In our approach, there are two places that we use local search. Firstly, the local search is placed into initialization step. When individuals have reached an acceptable development, they interact with each other by genetic algorithms. So that, a solution, which gives better results are selected for genetic algorithms approach. Another local search approach is applied into mutation step. In mutation, best evaluation values giving resultant is selected as mutated individual.

At initialization step, solutions of populations are created, randomly. After that every solution is evaluated according to the objective function. The sort by ascending order of the fitness values gives permission to have best potential solutions as the first elements of population. For selecting the elements of new generation, elitist strategy is used for some of the solutions. So that, best result-given solutions pass to the new generation. For the rest, we use roulette wheel method.

The crossover operator crosses two selected solution, under a certain probability. If generated probability is lower than a determined crossover probability, the parents pass to next generation, otherwise the parents reproduce new children for next generation by using

crossover method. After crossover, the mutation is applied according to the mutation probability. In mutation, another local search is carried out to the solution. Some new solutions are created with selecting different mutation points and different base stations. Then these solutions are evaluated for finding the best fitness value. The validation process is an important point for implementing memetic algorithms.

There are some parameters that we have to decide among the memetic algorithms. The number and size of populations, the number solutions that we have to create among the mutation step are some of them. They are also experimental parameters. In Figure 4.9 a pseudo code for our memetic algorithm approach is shown:

```
Begin
        InitializePopulation
        For each individual i in population
                Evaluate(i)
        For each individual i
                Local search engine(i)
        Repeat
                For j = 1 to recombinations
                        Select to merge a set Smer ⊂ Population
                        Offspring ← Recombine (Smer)
                        Evaluate(offspring)
                        Add evaluated offspring to Pop
                End For
                For j←1 to #mutations do
                        Select to mutate an individual i from Pop
                        i ←Mutate(i)
                        i ← Local Search Engine (i)
                        Evaluate(i)
                        Put the individual i back to the Pop
                End For
                Pop ← SelectPop(Pop)
                If Pop has converged then Pop← RestartPop(Pop)
        Until termination condition true
End
```

Figure 4.9. Pseudo code for memetic algorithm approach

5. EXPERIMENTS

In experimental phase, we compared three evolutionary algorithms that we improved for solving location area management problem.

A Pentium 4 2 GHz server with 2.6GB RAM is used for experiments. The server has Windows 2003 Server operating system. The execution time varies with the size of problem and parameters of the methods, but so far the longest run has taken about 12 hours for our reference case. In order to find the GA parameter values that give the best result, the first group of experiments is performed using different parameter values on different data sets. Experiments are repeated approximately 5 times for each data and parameter set. Because MOGA is an extended version of genetic algorithms, same parameter set that gives the best result for the GA is used in MOGA. But for memetic algorithms, a group of experiments is performed for finding the best parameter values. A scenario generator is run for creating data sets. In this scenario generator, paging rate, handover rate and busy hour call attempt rates are assigned to each base stations in the network for each time step. Number of base stations and base station controllers in the network defines the size of the problem.

After deciding best parameters for each method, our solution techniques are applied to the same data sets with their best parameters. The performance of genetic algorithm, multi objective genetic algorithm and memetic algorithm are compared. Comparison of GA with other evolutionary algorithms for different problems can be found in [32,33,34]. Some other problems, which MOGA is applied, can be found in [35,36,37,38] and comparison of MA with other algorithms can be found in [39,40,41,42,43].

5.1. Implementation of Algorithms

Algorithms are implemented by using C programming language. The structure, given, in section 3.2 is used as network definition for all algorithms given in this study. The implementation includes several functions which are summarized below:

- **Test File Generator:** This function creates the paging, handover and busy hour call attempt rates for each base station. Test file generator run once at the beginning of the program and used for all generations in the solution.
- **Initialization:** Random populations are created according to given parameters.
- **Evaluation:** For each generation, the fitness values of each individual are calculated in the evaluation function. The scenario that is created by scenario generator function is used for finding the costs of paging and registration.
- **Selection:** In the selection function, first, elitism is applied to the population. Elite individuals pass to the next generation, directly. The number of elite individuals is given as a parameter. Then, two individuals are selected for reproduction and resultant offspring are copied to next generation.
- **Crossover:** Two individuals are crossed-over in this function.
- **Mutation:** One of the base stations in the individual mutates in this function.
- **Validation:** The individuals, which are reproduced may be invalid. Some base stations can be lost, some of them can occur more than once or location areas in a base station can lose their vicinity. So a validation function is run after reproduction.

Same kind of functions is used for multi objective genetic algorithms but the implementation of them is different. Two separate evaluation functions are implemented for paging and registration. For memetic algorithms local search functions are written for initialization and mutation steps.

5.2. Experimental Study of Genetic Algorithms

This section has two main parts: Pre-experiments for Setting the Values of GA Parameters and experiments with respect to various network characters.

5.2.1. Experiments for Setting the Values of GA Parameters

A set of pilot experiments was run in order to determine appropriate parameters of genetic algorithms, which are crossover probability and mutation probability ratios.

Different crossover and mutation probability ratios are tried for different data sets, network sizes, generation numbers and population sizes.

In order to obtain optimal crossover probability, its value is set from a range between 0.1 and 0.9 with an increment of 0.1. The other parameters are remained fixed. Different sets are created for the other parameters and these sets are used for all combinations of crossover and mutation ratios. Network sizes of 16, 25 and 256, population sizes of 50, 100, 500 and generation numbers of 50, 100 and 500 are tried. In all experiments, in selection process, 10 per cent of each population's best results are reserved as elite solutions. High crossover rates give better results as it can be seen in Figure 5.1. So the crossover rate 0.7 is used for the remaining part of experiments.

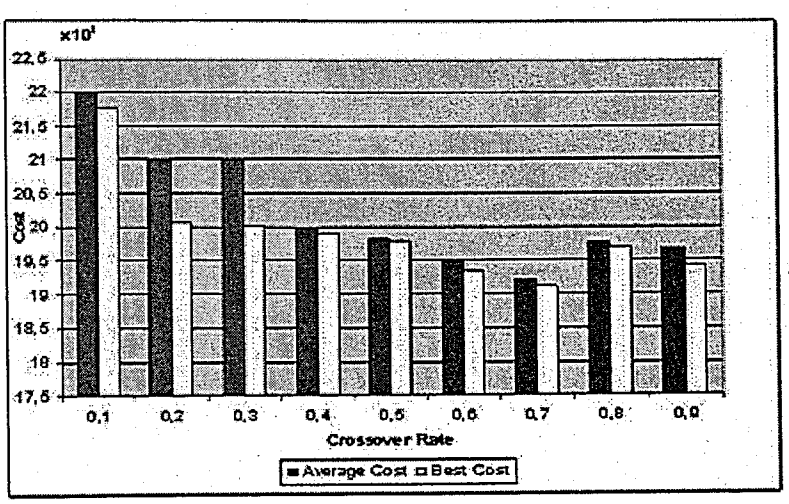

Figure 5.1. Effect of crossover rate to genetic algorithm

In order to obtain optimal mutation probability, its value is set from a range between 0.05 and 0.5 with an increment of 0.05; and the other parameters are remained fixed. So that, these mutation probabilities are run for the all the combinations of the parameters; network size 16, 25 and 256, population size 50, 100 and 500, generation size 50, 100 and 500 and crossover ratios 0.1 to 0.9 with step 0.1 for five scenarios. Figure 5.2 states the value 0.15 performs better compared to the other values.

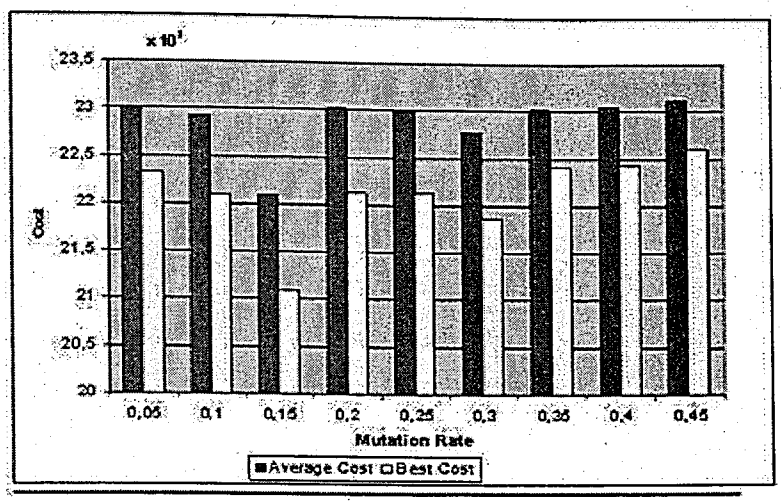

Figure 5.2. Effect of mutation ratio to genetic algorithm

So best crossover rate is determined as 0.7 and best mutation ratio is determined as 0.15. These parameters are used in main experimentation stage.

5.2.2. Main Experiments

After obtaining the mutation and crossover parameters, the scenarios ran for these parameters. There were three different kinds of parameters. These are network size, population size and generation size. Combinations of different variations of these parameters are applied to the genetic algorithm. The implementation is ran for network size of 16, 25, 256, population size of 50, 100, 500 and generation size of 50, 100 and 500 with crossover rate 0.7 and mutation rate 0.15 for different scenarios. The first thing to mention here is that the costs (average and best ones) increase exponentially when we increase network size. Table 5.1 shows the average and best costs for different network sizes. While cost of the network increases, the running time increases, too. This point is discussed in Section 5.4.

Table 5.1. Cost vs network sizes for GA-based study

Network Size	Average Cost	Best Cost
16	116335	95865
25	251688	197382
256	11932137	7017643

When we run genetic algorithm implementation for different scenarios, we find that both average and best costs decreases while GA reproduces new generations. The variation in averages costs through the generations, while network size is 25, can be seen in Figure 5.3.

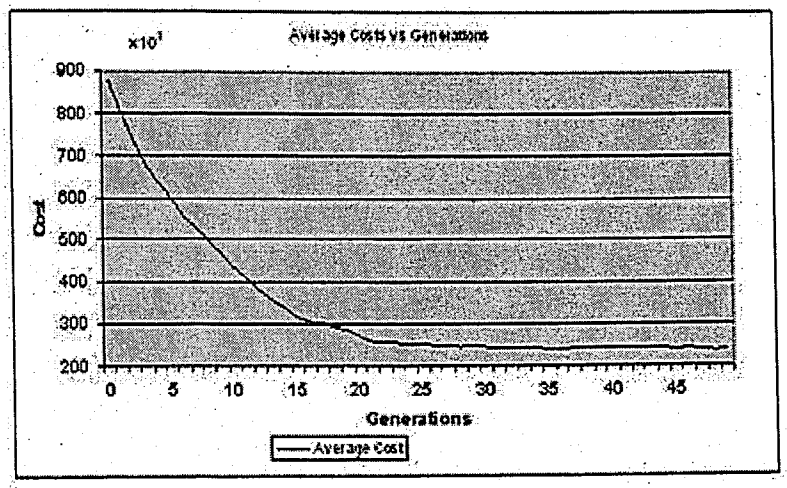

Figure 5.3. Average cost vs generation number for GA

The variation in best costs through the generations, while network size is 25, can be seen in Figure 5.4. As it can be seen easily, although the averages costs ripple during the generations, best costs always decreases. The reason for this dilemma is the elitist strategy that is applied in the algorithm. The best results of one generation are kept as elite for the next generation, best cost changes, if a lower cost is generated in new generation. But for the average cost, elitist strategy is not a key point.

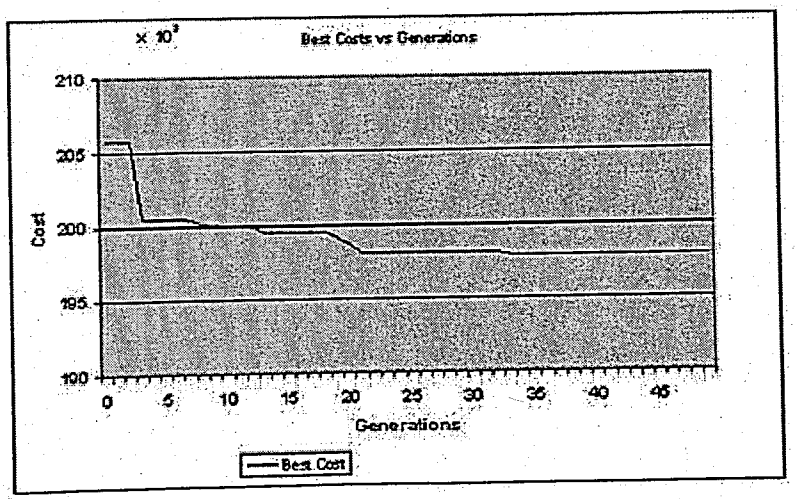

Figure 5.4. Best costs vs generation number for GA

Average and best cost variations when network size increases to 256 can be seen in Figure 5.5 and Figure 5.6. As it can be seen easily, the cost decreases, when generation number increases.

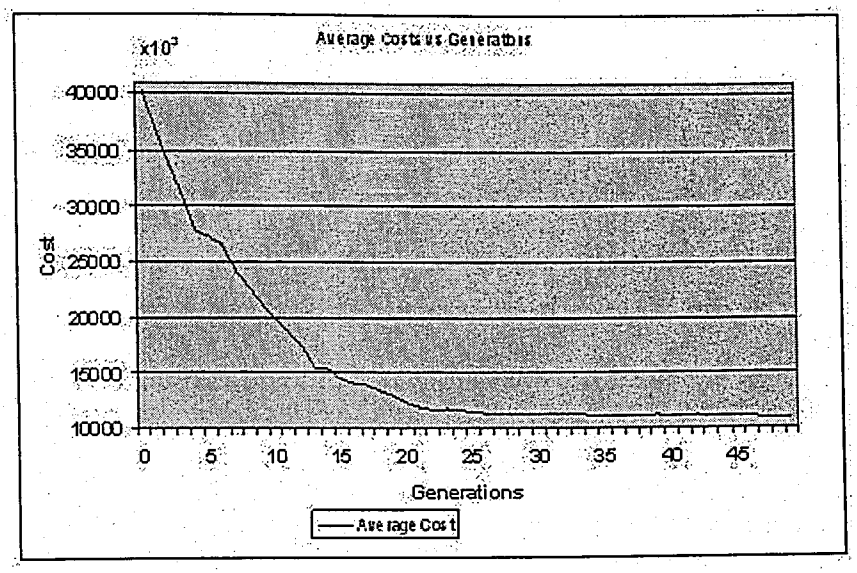

Figure 5.5. Average cost vs generation number for GA (2)

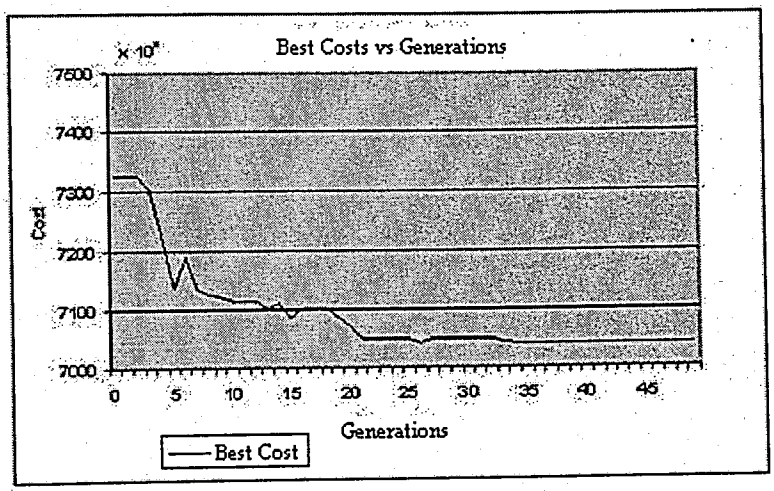

Figure 5.6. Best cost vs generation number for GA (2)

The following table contains information about the individual CPU time consumptions of operations in the algorithm. It can be easily seen in the table that the validation operation takes most of the CPU time. The reason for that is due to highly constrained definition of the problem, it is nearly impossible that an evolutionary operator resulted a valid offspring therefore a validation operation for almost every crossover and

mutation operation takes place. Validation operation itself is computationally intensive again due to constraints in the problem definition.

Table 5.2. CPU Time percentages of operations for GA

Element	CPU Time Percentage (%)
Generate Initial Population	3.05
Evaluation	0.58
Crossover	41.74
Mutation	0.43
Validation	54.2

5.3. Experimental Study of Multi Objective Genetic Algorithm

For the multi objective genetic algorithms experiments, optimal crossover and mutation probabilities determined in genetic algorithms are used. Different network size, generation size, population size combinations are tried for the crossover ratio 0.7 and mutation ratio 0.15. These combinations are tried for 5 different scenarios. MOGA has three main parts. In the first part a new population is generated and evaluated for registration cost. New generations are created according to this cost. Also another population which is created for paging cost is evaluated and best results from these two populations are combined to process in genetic algorithm.

CPU time percentages of operations for paging, registration and genetic algorithms can be seen in Table 5.3. There is no initialization cost in genetic algorithms because the populations of registration and paging are combined to be used in genetic algorithm.

Table 5.3. CPU Time percentages of operations for MOGA

Element	Paging CPU Time (%)	Registration CPU Time (%)	GA CPU Time (%)
Initialization	0.41	0.03	0
Evaluation	1.23	0.48	0.55
Crossover	49.57	52.53	39.41
Mutation	7.03	7.50	5.64
Validation	41.76	39.46	54.4

CPU time percentages of paging, registration and genetic algorithms parts of the whole application are shown in Table 5.3. GA part takes longer time because its population is more than the other two and also two costs are taking into account in evaluation operation.

Table 5.4. CPU Time percentages of three main parts of MOGA

Steps	CPU Time Percentage
Paging	18.20
Registration	26.66
GA	55.14

The variation in average and best costs through the generations for paging can be seen in Figure 5.7 and in Figure 5.8. In paging step, registration cost doesn't take into account. So that, the population tends to give results which have more location areas.

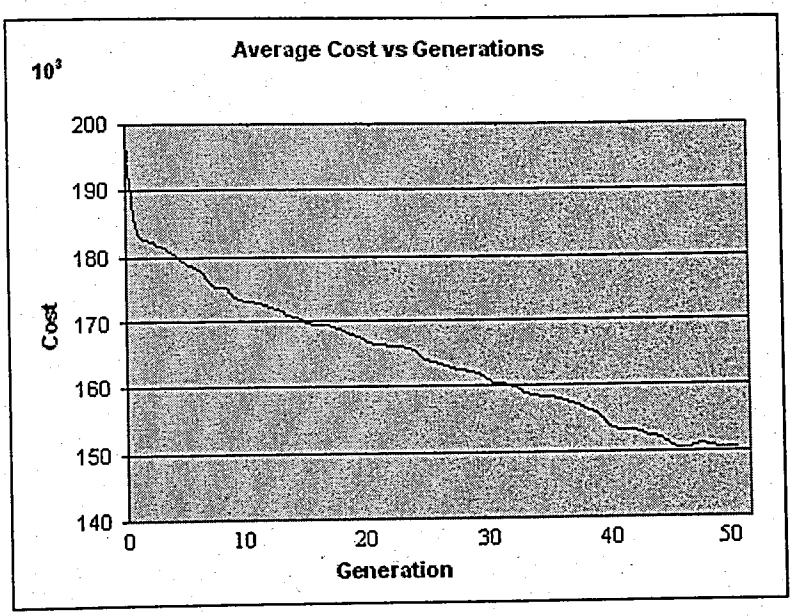

Figure 5.7. Average cost vs. generations for paging step

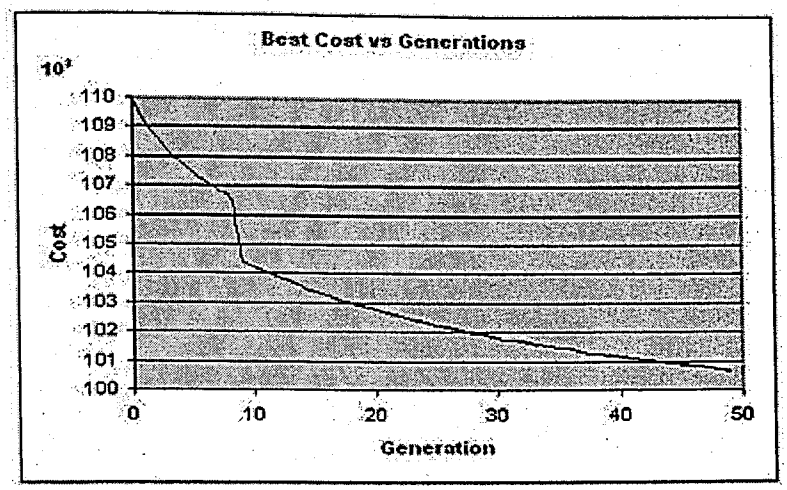

Figure 5.8. Best costs vs generations for paging step

Average and best cost variation of registration step can be seen in Figure 5.9 and Figure 5.10. The only cost taking into account is registration, so that the population tends to have less location areas while generation number increases.

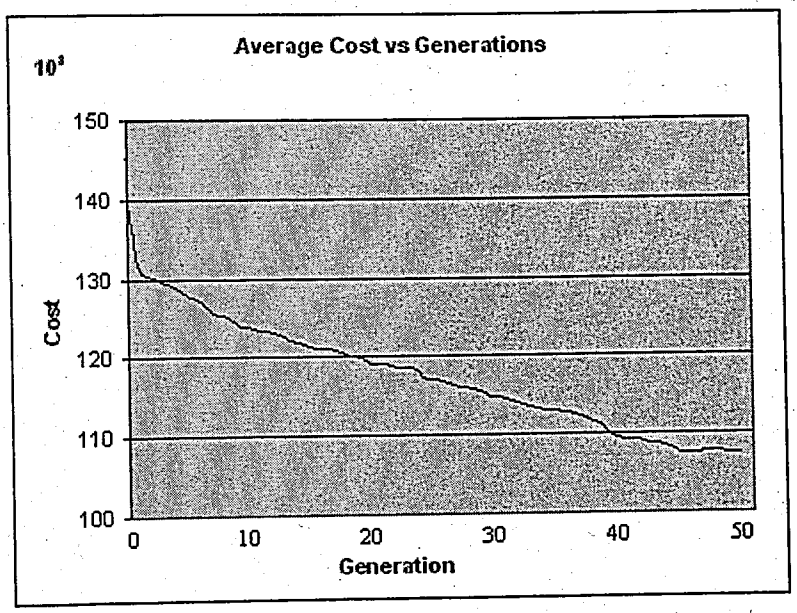

Figure 5.9. Average costs vs generations for registration step

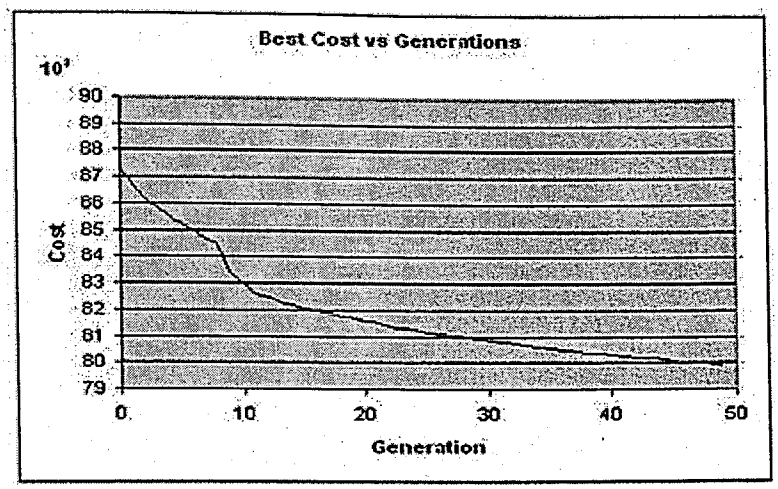

Figure 5.10. Best costs vs generations for registration step

After paging and registration step, the populations, which were created in these steps enter to the genetic algorithm step. The Figure 5.11 and Figure 5.12 shows the results of GA. As it can be seen easily, both the average and best costs are worse than the paging and registration costs. The reason for that is, in GA both paging and registration cost is taking into account.

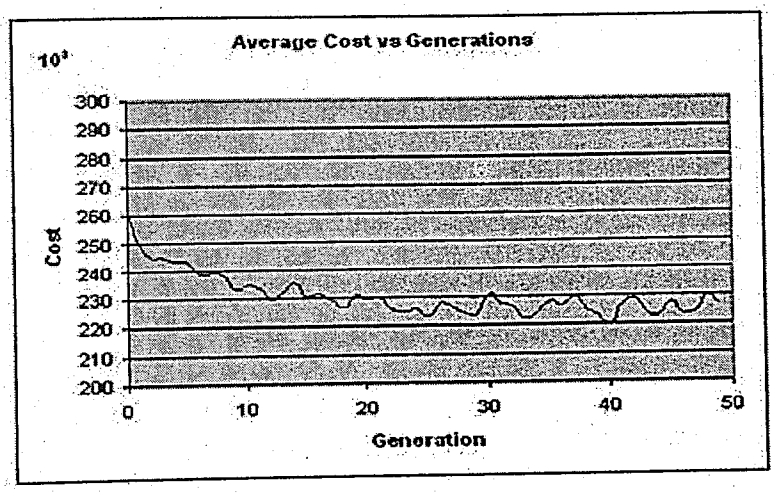

Figure 5.11. Average costs vs generations for GA step in MOGA

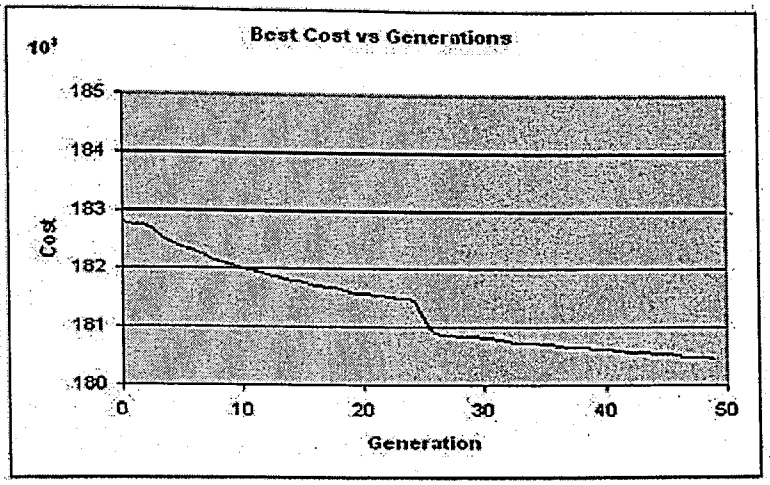

Figure 5.12. Best costs vs generations for GA step in MOGA

5.4. Experimental Study of Memetic Algorithm

As in the genetic algorithms part, this section has two main parts: Pre-experiments for Setting the Values of MA Parameters and main experiments.

5.4.1. Experiments for Setting the Values of MA Parameters

Two local searches run in memetic algorithm. First search is in initialization step. When an individual is created in initialization, a local search is made for this individual and the individual which gives the best result enters the initialized population. Different base stations mutate in this step, and the individual which gives the best cost is selected. The memetic algorithm has different parameters from genetic algorithms so a set of pilot experiments was run in order to determine appropriate parameters for the algorithm. Local search number is one of them. Deciding optimized local search number is important, because higher local search numbers results longer run times and small ones can cause huge costs. Mutation ratio is the other parameter. One of the local searches runs in mutation so we need high mutation ratios. We used crossover ratio 0.7 for memetic algorithm, too. Different local search number and mutation ratios are tried for various network sizes, generation numbers and population sizes.

The first MA parameter, which must be obtained, is the value of mutation ratio used to determine the mutation probability in each reproduction process. In order to obtain

optimal mutation rate, mutation probabilities from 0.25 to 1 with a step of 0.25 and local search numbers from 1 to (network size –1) with a step (network size/4) are tried while the other parameters are remained fixed. Different sets are created for the other parameters and these sets are used for all combinations of crossover and mutation ratios. Network sizes of 16, 25 and 256, population sizes of 50, 100, 500 and generation numbers of 50, 100 and 500 are tried. Best results for our testing conditions and is used for the remaining part of experiments. Mutation rate 0.5 gave best result for our test conditions so 0.5 is used for the rest of experiments (Figure 5.13).

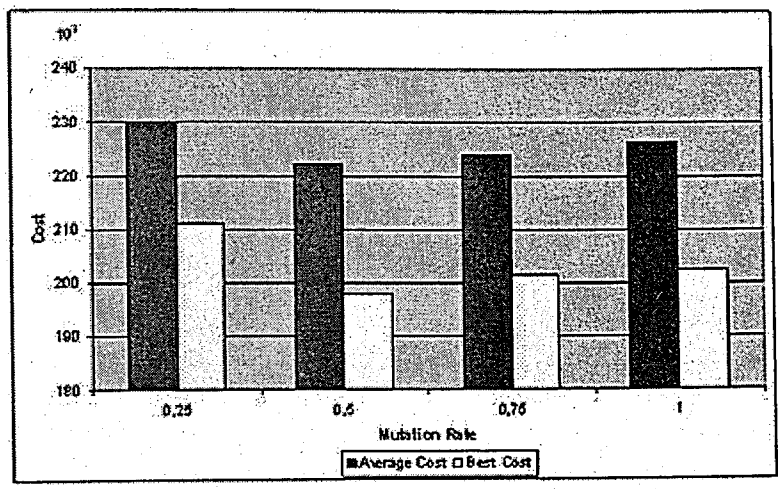

Figure 5.13. Costs vs mutation rate for MA

The second parameter used in MA algorithm is the local search number. This parameter denotes the number of local searches in initialization and mutation step. Local search numbers from 1 to (network size –1) with a step (network size/4) are tried while keeping all other parameters are fixed. So that, these local search numbers are run for the all the combinations of the parameters; network size 16, 25 and 256, population size 50, 100 and 500, generation size 50, 100 and 500, crossover rate 0.9, and mutation ratios 0.25 to 1 with step 0.25 for five scenarios. Figure 5.14 shows the results for network size 25.

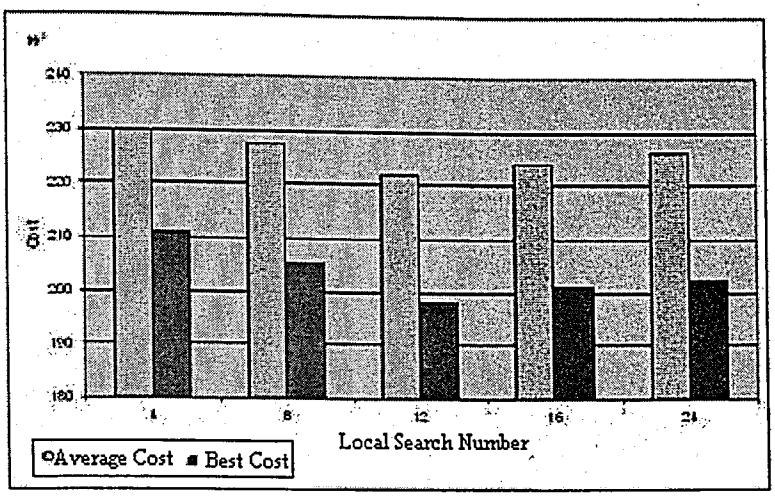

Figure 5.14. Local search number vs cost for MA

Increasing local search number, increases CPU time of the program, so 12 is selected as local search number, because it gives as good results as bigger local search numbers. Run times of different local search numbers for network size 25 is given in Figure 5.15.

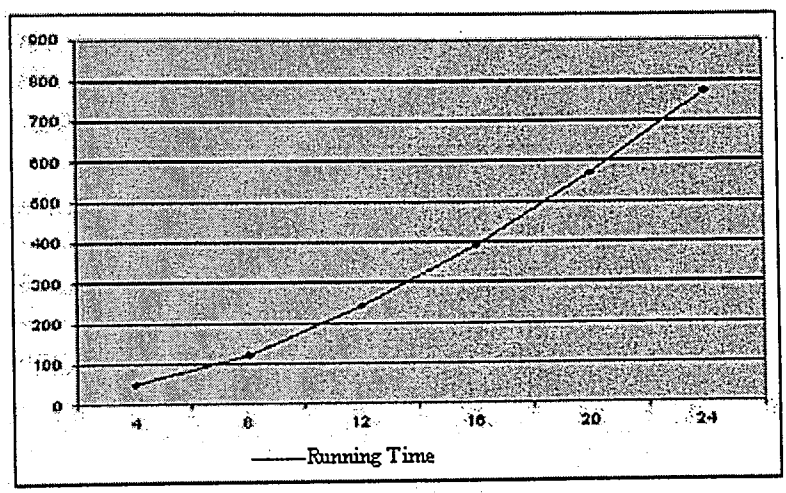

Figure 5.15. CPU time vs. local search numbers for MA

5.4.2. Main Experiments

After obtaining the mutation and local search number parameters, the scenarios ran for these parameters. There were three different kinds of parameters. These are network size, population size and generation size. Combinations of different variations of these parameters are applied to the genetic algorithm. The implementation is ran for network size of 16, 25, 256, population size of 50, 100, 500 and generation size of 50, 100 and 500 with

local search number 12 and mutation rate 0.5 for different scenarios. The following table contains information about the individual CPU time consumptions of operations in the algorithm. It can be easily seen in the table that the initialization and mutation operations take most of the CPU time because of the local searches.

Table 5.5. CPU Time percentages of operations for MA

Element	CPU Time Percentage
Generate Initial Population	51.07
Evaluation	0.14
Crossover	8.48
Mutation	32.79
Validation	7.52

The costs (average and best ones) increase exponentially when we increase network size like genetic algorithms. Table 5.6 shows the average and best costs for different network sizes. While cost of the network increases, the running time increases, too.

Table 5.6. Costs vs network size generation for MA

Network Size	Average Cost	Best Cost
16	110865	92788
25	220628	175589
256	11683342	6113532

5.5. Comparison of Solution Techniques

The comparison between GA, MOGA and MA are done based on the resulting costs found by them. Same scenarios are applied to three of them for their best parameters, then the algorithms are run for the other parameters such as network size, population size and generation number. The implementations were run for network size of 16, 25, 256, population size of 50, 100, 500 and generation size of 50, 100 and 500. For genetic and multi objecttive algorithms crossover value is taken as 0.7 and mutation ratio is 0.15. Local search number is taken as 12 and mutation ratio is taken as 0.5 for memetic algorithms.

Figure 5.16 shows the average costs vs generation number for three algorithms on a network size of 256. Memetic algorithm give better results for the same scenarios, for both average and best costs (Figure 5.17).

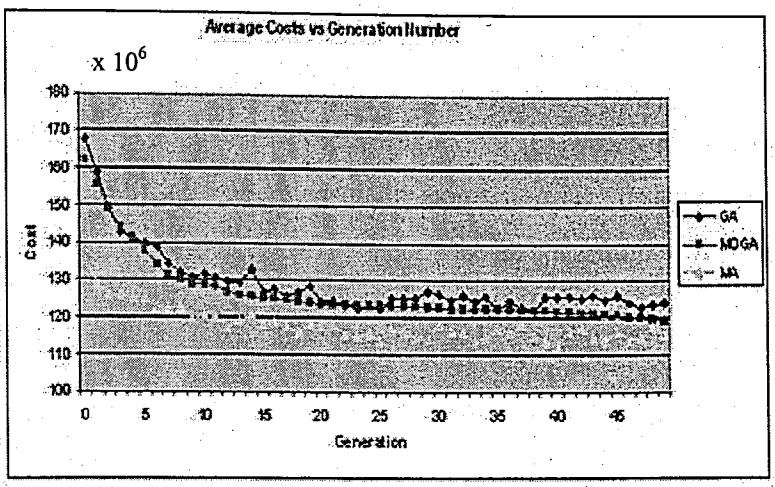

Figure 5.16. Comparison of average costs of three algorithms

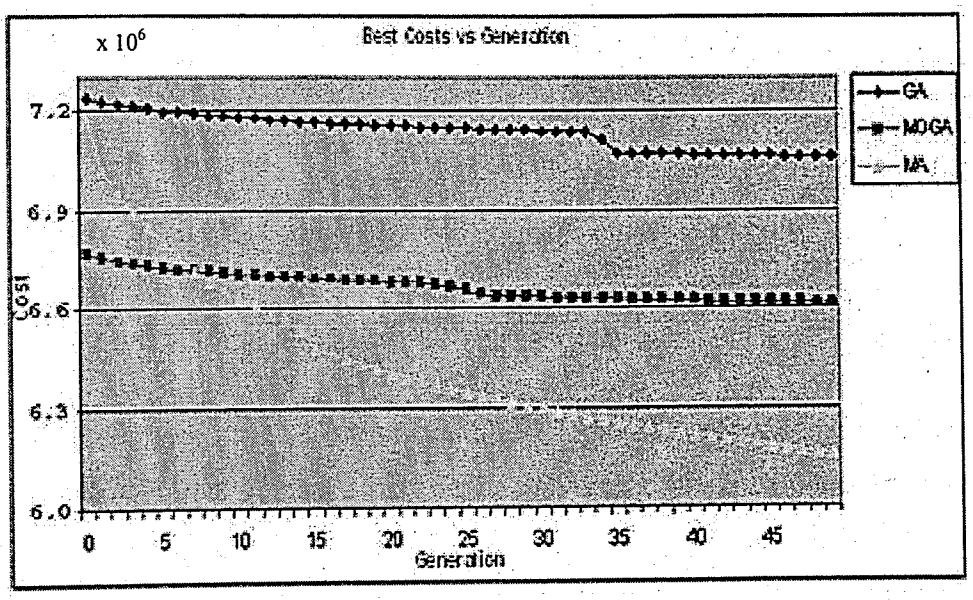

Figure 5.17. Comparison of best costs of three algorithms

Although memetic algorithm gives best results, the local search phase that is added to initialization and mutation, increases its run time. The CPU time comparison of three algorithms is given in Table 5.7.

Table 5.7. CPU Time comparison for three algorithms

	CPU Time
GA	5.72
MOGA	11.211
MA	69.99

We compare three algorithms, according to two network sizes and different random test files. These test files are generated at the beginning and used in the execution. The network sizes of 256 and 576 are used for experiments. The number of location areas, base station controllers and mobile services switching numbers are decided in the program. The results of four random sets for network size 256 and four random sets for network size 576 is given in Table 5.8. As it can be seen from the table memetic algorithm gives best results for both random sets and large random sets.

Table 5.8. Comparison for different problems of networks with sizes 256 and 576

Data	MA	MOGA	GA
Random Set 1	6190081	6737747	7076778
Random Set 2	6164078	6760515	7089641
Random Set 3	6171100	6729870	7059408
Random Set 4	6145502	6695736	7128708
Large Random Set 1	11387818	12551942	13213686
Large Random Set 2	11350490	12513284	13099680
Large Random Set 3	11403596	12417575	13054601
Large Random Set 4	11339959	12495774	13067994

Also we compare three algorithms with certain number of base stations, base station controllers and mobile services switching centers. A total of 9 experiments are done. Six of them are for 256 BSs, 8 BSCs and 8 MSCs and three of them are for 576 BSs, 16 BSCs and 16 MSCs. The results for these experiments are given in Table 5.9. For static number of network components the best results are obtained from memetic algorithms.

Table 5.9. Comparison of networks with certain number of units

Data	MA	MOGA	GA
Random Set 1	6144300	6712429	7091452
Random Set 2	6166256	6776539	7076173
Random Set 3	6114567	6773998	7118277
Random Set 4	6182006	6723659	7109052
Random Set 5	6212475	6736994	7106415
Random Set 6	6119193	6780836	7150934
Large Random Set 1	11367928	12506734	13163344
Large Random Set 2	11351544	12434506	13116654
Large Random Set 3	11429588	12395025	13182304

Some experiments are done for observing the effects of network properties. Handover rate, paging rate and call traffic rates are the properties, which are experimented. These properties are tested for low and high values. The results of these effects can be seen below:

Table 5.10. Comparison of networks with different network properties

	MA	MOGA	GA
Handover Rate			
High	6206598	6773359	7096921
	6170884	6701412	7058868
	6179701	6776569	7100970
	6203073	6766505	7118204
Low	6131275	6715668	7065760
	6114501	6690908	7064245
	6118961	6690074	7080212
	6134472	6711360	7082553
Paging Capacity			
High	6215478	6855930	7116211
	6226300	6728585	7125899
	6218279	6861176	7180706
	6287127	6798673	7149692
Low	6230987	6774830	7136776
	6208297	6753728	7143729
	6216939	6735318	7085944
	6234096	6751432	7104785
Call Traffic Capacity			
High	6254568	6942822	7129049
	6266408	6779541	7137907
	6285861	6912957	7184414
	6345046	6820403	7246032
Low	6244993	6810742	7169288
	6209816	6791638	7151353
	6245685	6830489	7174565
	6321029	6803777	7160504

6. CONCLUSIONS

Location area management is an important topic and it has many advantages. One of the gains of location area management is reducing the unnecessary resource usage in mobile networks so that mobile users can communicate more easily. Also it helps to decrease the network construction costs by reducing the network elements needed. There are some constraints for designing feasible networks such as paging capacity, busy hour call attempt rate capacity etc. We formulated an optimization problem for location area management so that base stations assign to location areas. We proposed three solution techniques based on genetic algorithm, multi objective genetic algorithm and memetic algorithm. We described the implementation details of these algorithms and also compared them with each other.

For comparing three methods, three types of experiments are performed. First, optimized parameters are found for three methods. The algorithms are run multiple times to guarantee the correctness. The minimum or average of the results is taking into account. A scenario generator is run for creating different scenarios. It is found that memetic algorithm gives the best results when compared to other methods. Also multi objective genetic algorithm gives better results than genetic algorithms. However, memetic algorithm has a time cost. It runs about 14 times longer than genetic algorithm and about 7 times longer than multi objective genetic algorithm. As a future work, parallelization techniques may be applied to these three algorithms. So that the CPU time disadvantage of memetic algorithm can be obstructed. There is an MS thesis about this topic.

As a result, we believe that we formulated the location area management and base station assigning problem that includes the majority of the previously proposed problems as its special cases. We also proposed hopeful algorithms for this difficult optimization problem.

REFERENCES

1. Guanling, L. and L. P. C. Arbee., "The Design of Location Regions Using User Movement Behaviors in PCS Systems", *Multimedia Tools and Applications*, Volume 15, Issue 2, Pages 187 – 202, November 2001.

2. Abutaleb, A. and O.K.V. Li , "Location Update Optimization in Personal Communication Systems", *Wireless Networks*, Volume 3, Issue 3, Pages 205–216, 1997.

3. Scourias, J., *Overview of the Global System for Mobile Communications*, http://ccnga.uwaterloo.ca/ jscouria/GSM/ gsmreport.html, 1997.

4. Merchant, A. and B. Sengupta, "Assignments of Cells to Switches in PCS Networks", *IEEE/ACM Transactions on Networking*, Vol.3, No.5, pp.521-526, October 1995.

5. Demirkol, İ., C. Ersoy, M. U. Çağlayan and H. Deliç, "Location Area Planning and Cell to Switch Assignment in Cellular Networks Using Simulated Annealing", *IEEE Tr. on Wireless Communications*, Vol.3, No.3, pp.880-890, May 2004.

6. Garey, M.R. and D. S. Johnson, *Computers and Interactability: A Guide to the Theory of NP-Completeness*, San Francisco, California: W.H. Freeman, 1979.

7. Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

8. Blickle, T., and L. Thiele , "A Comparison of Selection Schemes Used in Genetic Algorithms", *TiK Report Nr.11*, 2nd Edition, Swiss Federal Institute of Technology, 1995.

9. Moscato, P., *Memetic Algorithms: A Short Introduction, New Ideas in Optimization*, McGraw-Hill Ltd., UK, Maidenhead, UK, 1999.

10. Zhang, J., "Location Management in Cellular Networks", *in Handbook of Wireless Networks and Mobile Computing*, Ivan Stojmenovic (Editor), John Wiley & Sons, 27-49, 2002.

11. Das, S., "A Profile Based Location Update Strategy for Cellular Networks", *Dept of Computer Sc. & Tech, Bengal Engineering College*, Howrah, INDIA.

12. Demirkol, İ., "*Location Area Planning and Cell to Switch Assignment in Cellular Networks*", M.Sc. Report, Boğaziçi University, 2002.

13. Sen, S. K., A. Bhattacharya, and S. Das, "A Selective Location Update Strategy for PCS Users", *IEEE Network*, vol. 14, no. 3, pp. 6-14, May 2000.

14. Okasaka, S., S. Onoe, S. Yasuda, and A. Maebara, "A New Location Updating Method for Digital Cellular Systems", *Proc. 41st IEEE Vehicular Technology Conf. Gateway to the Future Technology in Motion*, 1991.

15. Plassmann, D., "Location Management Strategies for Mobile Cellular Networks of Third Generation", *Proc. IEEE 44th Vehicular Technology Conf.*, 1994.

16. Yeung, K. L. and T. S. P. Yum, "A Comparative Study on Location Tracking Strategies in Cellular Mobile Radio Systems", *Global Harmony IEEE Global Telecomm. Conf. Technical Program Conf. Record*, (GLOBECOM '95), 1995.

17. Saraydar, U., O. Kelly and C. Rose, "One-dimensional Location Area Design", *IEEE Trans. Vehicular Technol. 49 (5)*, 1626-1632 , 2000.

18. Demirkol, İ., C. Ersoy, M. U. Çağlayan and H. Deliç, "Location Area Planning in Cellular Networks Using Simulated Annealing", *INFOCOM 2001*:13-2.

19. Subrata, R. and A. Zomaya, "A Comparison of Three Artificial Life Techniques for Reporting Cell Planning in Mobile Computing", *IEEE Trans. Parallel Distrib. Syst. 14(2)*: 142-153, 2003.

20. Subrata, R. and A. Zomaya, "Evolving Cellular Automata for Location Management in Mobile Computing Networks", *IEEE Trans. Parallel Distrib. Syst. 14(1)*: 13-26, 2003.

21. Quintero, A. and S.Pierre, "Sequential and Multi-population Memetic Algorithms for Assigning Cells to Switches in Cellular Mobile Networks", *Computer Networks*, Vol. 43, No. 3, pp. 247-261, 2003.

22. Quintero, A. and S. Pierre, "Evolutionary Approach to Optimize the Assignment of Cells to Switches in Personal Communication Networks", Computer *Communications 26(9)*: 927-938, 2003.

23. Eren, M. and C. Ersoy, "Optimal Virtual Path Routing Using a Parallel Annealed Genetic Algorithm," *IEEE ICT 2001*, Vol 1, pp. 336-341, Bucharest, 4-6, June 2001.

24. Beightler, C.S., D. T. Philips and D. J. Wilde, *Foundations of Optimization*, 2nd ed., Prentice-Hall, Inc., Englewood, N.J.,1979.

25. Wall, M., *An Introduction to Genetic Algorithms*, http://lancet.mit.edu/~mbwall/ presentations/IntroToGAs/, 2003.

26. Cassacia, L., *A Short Survey of Genetic Algorithms*, http://www.illabirinto.com/ science/genalgo.htm, 2003.

27. Sullivan, M., *Basic Genetic Algorithm Operations*, http://www.cs.qub.ac.uk/ ~M.Sullivan/ga/ga4.html, 2003.

28. Moscato, P. and M. G. Norman, "A `Memetic' Approach for the Traveling Salesman Problem. Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems", *Parallel Computing and Transputer Applications*, edited by M. Valero, E. Onate, M. Jane, J.L. Larriba and B. Suarez, Ed. IOS Press, Amsterdam, pp. 187-194, 1992.

29. Moscato, P., "On Genetic Crossover Operators for Relative Order Preservation", *Caltech Concurrent Computation Program*, C3P Report 778, 1989.

30. Eiben, A. E. and J. E. Smith, *Introduction to Evolutionary Computing*, Springer, 2003.

31. Reynolds, R. G., and W. Sverdlik, "Problem Solving Using Cultural Algorithms", *International Conference on Evolutionary Computation*, 645-650, 1994.

32. Tate, D. M. and A. Smith, "A Genetic Approach to the Quadratic Assignment Problem,", *Computers and Operations Research*, vol. 22, no.1, 73-83, 1995.

33. Wright, A. H., J. E. Rowe and J. R. Neil, "Analysis of the Genetic Algorithm on the Single-peak and Double-peak Landscapes", *Proceedings of CEC 2002*, IEEE Press, 2002.

34. Horváth, M., A. Márkus and J. Váncza, "Process Planning with Genetic Algorithms on Results of Knowledge-Based Reasoning." *Int. J. of Computer Integrated Manufacturing*, 9, 145-166, 1996.

35. Deb, K., "Solving Goal Programming Problems Using Multi-Objective Genetic Algorithms." *Proc. Congr. on Evolutionary Computation, Washington DC/USA*, p.77 – 84, 1999.

36. Morita, M. E, R. Sabourin, F. Bortolozzi and C. Y. Suen, "Unsupervised Feature Selection Using Multi-Objective Genetic Algorithms for Handwritten Word Recognition". *ICDAR 2003*: 666-670.

37. Shaw, K. J. and P. J. Fleming, "Initial Study of Multi-Objective Genetic Algorithms for Scheduling the Production of Chilled Ready Meals". *Proceedings of Mendel '96, the 2nd International Mendel Conference on Genetic Algorithms, Brno*, Czech Republic, June 26th - 28th, 1996.

38. Chipperfield, A. J. and P. J. Fleming, "Gas Turbine Engine Controller Design Using

Multiobjective Genetic Algorithms", *Proc. 1st IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, Sheffield, UK, pp. 214-219, 12-14 Sept. 1995.

39. Fang, H. L., P. Ross and D. Corne, "A Promising Hybrid GA/Heuristic Approach for Open Shop Scheduling Problems" *In ECAI 94: 11th European Conference on Artificial Intelligence*, A. Cohn (ed), John Wiley and Sons Ltd, 1994, pages 590-594,1994.

40. Alpert, C. J. and A. B. Kahng, "A Hybrid Multilevel/Genetic Approach for Circuit Partitioning", *Fifth ACM/SIGDA Physical Design Workshop*, pp. 100-105, April 1996.

41. Whitley, D., K. Mathias, C. Stork and T. Kusuma, "Staged Hybrid Genetic Search for Seismic Data Imaging", *IEEE Conf. on Evolutionary Computation, Volume 1*, pp: 356-361, 1994.

42. Merz, P. and B. Freisleben, "Memetic Algorithms and the Fitness Landscape of the Graph Bi-Partitioning Problem", *Parallel Problem Solving from Nature V*, PPSN '98.

43. Burke, E. K., J. P. Newall and R. F. Weare, "A Memetic Algorithm for Univerisity Exam Timetablingin The Practice and Theory of Automated Timetabling", *Springer-Verlag Lecture Notes in Computer Science*, Vol. 1153.