

Location Area Planning in Cellular Networks Using Simulated Annealing^{*}

Ilker Demirkol¹, Cem Ersoy¹, M. Ufuk Caglayan¹, Hakan Delic²

¹NETLAB, Department of Computer Engineering

²BUSIM Lab., Department of Electrical and Electronics Engineering
Bogazici University, Bebek 80815 Istanbul, Turkey

Abstract— Location area (LA) planning plays an important role in cellular networks because of the trade-off caused by paging and registration signaling. The upper bound on the size of an LA is the service area of a mobile switching center (MSC). In that extreme case, the cost of paging is at its maximum, but no registration is needed. On the other hand, if each cell is an LA, the paging cost is minimal, but the registration cost is the largest. In general, the most important component of these costs is the load on the signaling resources. Between the extremes lie one or more partitions of the MSC service area that minimize the total cost of paging and registration. In this paper, we try to find an optimal method for determining the location areas. For that purpose, we use the available network information to formulate a realistic optimization problem. We propose an algorithm based on simulated annealing (SA) for the solution of the resulting problem. Then, we investigate the quality of the SA technique by comparing its results to greedy search and random generation methods.

Index terms—Location Area, Cellular Networks, Simulated Annealing

I. INTRODUCTION

In cellular communication systems, upon the arrival of a mobile-terminated call, the system tries to find the mobile terminal by searching for it among a set of base stations (BSs) using the current region knowledge of the mobile. This search is called paging, and the set of base stations in which a mobile is paged is called a location area (LA). At each LA boundary crossing, mobile terminals register their new location through signaling in order to update the location management database. Therefore, the size of the LA is important in reducing the cost of paging and registration signaling [11]. Although there may be events other than location updates that cause registration, we shall use the term “registration” instead of “location update” throughout the paper.

The paging or registration procedures induce costs with three different bases. One of them is the *select* or *update* queries on the database elements of the cellular network. The other one is the load created on the physical connection part of the cellular network. The last one is the load on the air interface of the cellular network. Among these, the least scalable one (that is, the most affected resource as the population and traffic in a cell grow) is the radio bandwidth resource. In addition, because the radio bandwidth is shared by those control functions such as paging, registration and call traffic, it is considered to be the scarcest resource in a wireless network. Therefore, for paging and registration signaling costs, we may just take the cost of the

load on the radio bandwidth into account. Thus, it is desirable to design wireless networks that efficiently utilize the limited radio bandwidth. Use of effective mobility management schemes emerges as an important means to achieve such efficiency [12].

LA management in Personal Communication Services (PCS) literature can be classified by the location update and paging scheme employed. Static schemes (in which LA boundaries are fixed, and the cells that may result in registration are defined) can be zone-based [11, 12] or profile-based [13], and dynamic schemes (in which LA boundaries are not defined, and the registration decision is made according to some criteria) can be time-based, movement-based [7], distance-based [1], or state-based [10]. Because the LAs have fixed size by the definition in this paper, we propose a solution for zone-based location update and paging schemes.

The research focusing on LA management in PCS networks has generally considered dynamic plans. Nevertheless, both dynamic and static schemes have a common goal: minimizing the total paging and registration cost. To that end, an appropriate objective function requires the addition of paging and registration costs, which do not have comparable units. In order to alleviate this quandary, researchers have made certain assumptions for the relative values of these costs (e.g., assuming one unit-cost for each paging event and ten unit-cost for each registration event). Such assumptions have the deficiency of being the same throughout the entire network. In reality, the load (i.e., cost) of paging and registration to the network varies from cell to cell. In order to avoid the issues relating to the summing these two types of costs, instead of trying to minimize both, we bound the paging cost and minimize the registration cost, which still results in a difficult combinatorial optimization problem. However, compared to minimizing paging within acceptable registration capacities, this goal is advantageous since paging capacity is easier to quantify as a constraint as explained in Section II. We propose a simulated annealing (SA) algorithm for this problem in Section III, and finally, we present the results of the computational experiments in Section IV.

II. FORMULATION OF THE LOCATION AREA PLANNING PROBLEM

The model adopted in this paper complies with the Base Station (BS), Base Station Controller (BSC) and Mobile Switching Center (MSC) hierarchy of the Global System for Mobile Communication (GSM) standard although the basic idea extends to any cellular network. While formulating the LA planning problem, we make use of only the available network

^{*} This work is supported by Turkcell A.S.

information, and try to include all realistic constraints and goals.

The problem in LA planning and dimensioning arises due to the trade-off between the paging cost and the registration cost. The paging cost is a result of the arriving calls to mobiles, i.e., mobile-terminated calls. The called party has to be searched and found in order to establish the connection. The paging capacities of BS i and BSC j , denoted respectively by P_i^{BS} and P_j^{BSC} , should not be exceeded in a properly functional cellular network for any i, j . For evaluating the paging loads, the paging rate per unit time for each cell i , I_i , is used, where I_i just includes the paging generated because of the mobiles residing in that cell.

The maximum call traffic load capacities C_j^{BSC} and C_k^{MSC} of respectively BSC j and MSC k constitute another constraint that must be obeyed for all j and k . To check the constraint, the busy-hour call traffic of each cell, denoted by c_i for cell i , is used.

In order to have a feasible cellular network, the limited call processing capability of MSCs and BSCs may create a bound on the peak call arrival rate, which includes not only the established connections but also the failed attempts. This limit is called the Busy Hour Call Attempt (BHCA) capacity, and for BSC j and MSC k , they are represented as D_k^{MSC} and D_j^{BSC} , respectively. The BHCA load on the BSCs and MSCs are calculated through the call attempt rates of the cells. To that end, let d_i denote the peak call attempt rate of cell i per unit time.

Each BS has a finite number of transmitters (TRXs), which defines the number of channels available in that cell. The TRX capacity is a vendor-specific constraint of BSCs and MSCs. For BSC j and MSC k , the sum of the number of TRXs for each cell connected to that BSC or MSC should comply with limits R_j^{BSC} and R_k^{MSC} , respectively. In order to check for this constraint, the number of TRXs for each cell i (r_i) must be known.

For the registration load calculation, the “idle” (i.e., with handset on, but not in a conversation) mobile flow rate between each cell pair is required. Since these data generally cannot be collected, we approximate this aggregate mobile flow behavior by assuming that the mobile flow rate between any given two cells is proportional to the hand-off rate between these cells. In other words, the hand-off rate is used instead of the mobile flow data. Let h_{ij} be the hand-off rate from i th to j th cell.

The BS-BSC-MSC connections restrict the possible assignments of LAs to BSs. Hence, to have an optimal LA configuration, the BS-BSC-MSC topology decision should also be considered as part of the problem.

To propose a BS-BSC topology, we must know which BS can be connected to which BSC, and discard those BS-BSC pairs that are not feasible (due to physical limitations, geographical constraints, etc.). This information is available in the form of a proximity matrix, A , among BSs and BSCs in which $a_{ij} = 1$ if BS $_i$ may be connected to BSC $_j$, and $a_{ij} = 0$ otherwise.

Similarly, for the BSC-MSC topology, we shall use a proximity matrix, B , in which $b_{jk} = 1$ if BSC $_j$ may be connected to MSC $_k$, and $b_{jk} = 0$ otherwise.

In addition to these, the following design variables will be in the formulation:

The LA-BS topology is represented by the matrix $L = \{l_{in}\}$ in which, if BS i resides in the n th LA, then $l_{in} = 1$, and $l_{in} = 0$ otherwise. The matrix L is used to establish another matrix $D = \{d_{ij}\}$ in which, if BSs i and j reside in different LAs, then $d_{ij} = 1$, and $d_{ij} = 0$ otherwise.

The BSC-MSC topology is represented by the matrix $Y = \{y_{jk}\}$ in which if BSC j is connected to the k th MSC, then $y_{jk} = 1$, and $y_{jk} = 0$ otherwise.

The BS-BSC topology is represented by the matrix $X = \{x_{ij}\}$ in which, if BS i connected to the j th BSC, then $x_{ij} = 1$, and $x_{ij} = 0$ otherwise.

A call arrival to a mobile results in paging at each BS residing in the LA where the mobile is currently registered. Hence, paging load on a cell is determined by the total number of call arrivals to cells that belong to the same LA. For that information, we use a vector, I^* , where the i th entry holds that total value for i th cell:

$$I_i^* = I_i + \sum_j I_j (1 - d_{ij}) \quad (1)$$

As a result, our “minimize the total registration signaling” goal is expressed as follows:

$$\text{Minimize } \sum_i \sum_j d_{ij} h_{ij} \quad (2)$$

subject to the following constraints:

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

$$\sum_j x_{ij} = 1, \forall i. \quad (3)$$

2. Each BSC should be assigned to exactly one MSC. For the j th BSC, the BSC-MSC topology matrix should have only one entry in the j th row that has a value of 1, and all others must be 0.

$$\sum_k y_{jk} = 1, \forall j. \quad (4)$$

3. Each BS should be assigned to exactly one LA. For the i th cell, the BS-LA topology matrix should have only one entry in the i th row that has a value of 1, and all others must be 0.

$$\sum_n l_{in} = 1, \forall i. \quad (5)$$

4. Each LA must reside within exactly one MSC. If a cell pair belong to the same LA, then they also must belong to the same MSC. Thus, if

$$\sum_n l_{in} l_{sn} = 1,$$

then

$$\sum_k (\sum_j x_{ij} y_{jk} \cdot \sum_r x_{sr} y_{rk}) = 1, \forall(i, s). \quad (6)$$

5. The paging capacity of each BS must not be exceeded.

$$I_i^* < P_i^{BS}, \forall i. \quad (7)$$

6. The paging capacity of each BSC must not be exceeded. Assuming that a BSC pages the mobile station (MS) separately in each cell of the same LA, this constraint becomes

$$\sum_i x_{ij} I_i^* < P_j^{BSC}, \forall j. \quad (8)$$

7. The call traffic capacity of each BSC must not be exceeded.

$$\sum_i x_{ij} c_i < C_j^{BSC}, \forall j. \quad (9)$$

8. The call traffic capacity of each MSC must not be exceeded.

$$\sum_j \sum_i x_{ij} y_{jk} c_i < C_k^{MSC}, \forall k. \quad (10)$$

9. The BHCA capacity of each BSC must not be exceeded.

$$\sum_i x_{ij} d_i < D_j^{BSC}, \forall j. \quad (11)$$

10. The BHCA capacity of each MSC must not be exceeded.

$$\sum_j \sum_i x_{ij} y_{jk} d_i < D_k^{MSC}, \forall k. \quad (12)$$

11. The TRX capacity of each BSC must not be exceeded.

$$\sum_i x_{ij} r_i < R_j^{BSC}, \forall j. \quad (13)$$

12. The TRX capacity of each MSC must not be exceeded.

$$\sum_j \sum_i x_{ij} y_{jk} r_i < R_k^{MSC}, \forall k. \quad (14)$$

13. The proximity constraints must be satisfied.

$$x_{ij} \leq a_{ij}, \forall(i, j). \quad (15)$$

$$y_{jk} \leq b_{jk}, \forall(j, k). \quad (16)$$

The LA planning problem, formulated in equations (1)-(16) in the most general setting, poses a difficult optimization issue in which the LA-BS topology (the matrix L), the BSC-MSC topology (the matrix Y) and the BS-BSC topology (the matrix X) have to be decided. There may be many special cases of the general LA planning problem, and some of the constraints may not be necessary. For example, there may not be any call capacity requirements on the MSCs, in which case Constraint (10) is omitted. Another special case arises when there is exactly one BSC for each MSC. Then, Y is already known.

III. SIMULATED ANNEALING-BASED SOLUTION TECHNIQUE

Our aim is to propose an algorithm that takes the available constraints, capacity and load information described in Section

II as inputs, and find an optimal or near-optimal solution as a network topology which includes the assignment of cells (BSs) to switches (BSCs and MSCs) and cells to location areas (LAs).

The determining factors of the solution space size are BS-to-BSC assignments, BSC-to-MSC assignments, and LA-to-BS assignments. With n BSs, m BSCs and p MSCs, the number of possible BS-to-BSC assignments is n^m as for each BS, there are m possible BSCs to be connected. Likewise, the number of possible BSC-MSC assignments is m^p , and because the upper bound on the number of LAs is the number of BSs, the number of possible BS-to-LA assignments is n^n . Consequently the size of the solution space is found to be $n^m m^p n^n$. Since the solution space size depends on a number of variables, exhaustive methods would result in exponential time complexity. For instance, for a system with 1000 BSs, 10 BSCs and 10 MSCs, there are 10^{3040} distinct (feasible or not) network formations.

The ‘‘bin packing’’ problem is NP-hard; i.e., there is no optimization technique that solves the ‘‘bin packing’’ problem in linear time complexity [3]. The sub-problem of assigning BSs to LAs can be mapped to the bin packing problem in the following manner. LAs correspond to bins due to their common paging capacity limit. We try to pack the BSs to LAs in such a way that the number of LAs is minimized so that the network operates with the least cost. Hence, even the BS-to-LA assignment part of the problem is NP-hard, and as a consequence, the overall optimization problem can be classified as NP-hard.

Considering the difficult optimization problem at hand, it is not possible to guarantee the optimal solution in reasonable run times. Therefore, techniques that offer near-optimal solutions within acceptable run times are required. One such method that finds a sub-optimal solution in reasonable time without searching the whole solution space is simulated annealing (SA). SA was introduced by Metropolis *et al.* [8] and is used to approximate the solution of very large combinatorial optimization problems [6]. Besides the traditional greedy local search techniques, the stochastic properties of the SA algorithm prevent it to get stuck to local minima. On the other hand, in traditional greedy local search, the quality of the final result heavily depends on the initial solution. In contrast, the idea behind SA is to adequately explore the whole solution space early on so that the final solution is insensitive to the starting state [5].

The information supplied to the SA based algorithm includes the BHCA, call and TRX capacity constraints for BSCs and MSCs; paging capacity constraints and proximity data (allowed connections to MSCs and BSCs, respectively) for BSCs and BSs. Moreover, information related to the call traffic for each BS is also provided. This cell traffic and capacity information include the peak call attempt rate, peak call traffic rate, hand-off rate to neighboring BSs, and the number of TRXs attached.

The SA-based algorithm described here finds a network topology that consists of the LA-to-BS assignment matrix L , the BSC-to-MSC topology matrix Y and the BS-to-BSC topology matrix X . In addition, the total registration cost of the solution is presented, as well.

The algorithm starts with an initial feasible solution, which is set as the current solution. Randomly, a neighboring solution from the solution space is chosen, and its cost is compared to that of the current solution. If the cost is improved, this neighbor solution is kept and set as the current solution. Otherwise, this solution is accepted with a probability that is calculated according to the stage the algorithm is in (we designate this stage via a variable called “temperature”). The probability of accepting a worse solution is [6]:

$$P = e^{-DE/T} \quad (17)$$

where DE is the difference between cost of the neighbor solution and the current solution. The temperature is managed by the cooling schedule that controls the execution of the algorithm.

For a successful SA implementation, the two key items that must be defined carefully are the moves that create the neighbor solutions and the cooling schedule.

A. Neighborhood Structure

A feasible solution is a topology where all network nodes (BSs, BSCs, MSCs) are connected, and the LA borders are specified such that the constraints are satisfied. Therefore, a (feasible) neighbor solution may be generated by any of the following three types of moves:

1) Changing a BS-to-BSC assignment: A BS to be moved is randomly chosen, and then a BSC is randomly selected among all BSCs in the proximity of that BS. Before executing this move, the capacity constraints affected by this BS-BSC connection are checked. If these constraints are not violated, the new BSC is assigned to the BS. Then, a random feasible LA is searched to assign that BS among the existing LAs residing in the new BSC. Because of the limitation that an LA cannot spread over multiple MSCs, instead of checking whether the new and the old BSC are connected to the same MSC, we directly try to find an LA residing in the new BS. If no feasible LA is found, then a new LA is created for the new BSC. Next, all load updates resulting from that move are calculated on the network.

2) Changing a BSC-to-MS C assignment: This move alters the network topology significantly, since all BSs connected to a BSC also move with it. Firstly a BSC, denoted by BSC^* , is chosen randomly to change its MSC connection. A candidate MSC is also randomly chosen among all MSCs. Again, the capacity and proximity constraints of the BSC-MS C connection are checked. If these constraints are not violated, the new MSC is assigned to BSC^* . All LAs residing merely in BSC^* (i.e., none of their BSs are connected to another BSC) stay with their BS connections (i.e., nothing is altered for these LAs), but LAs having only some of their BSs connected to BSC^* undergo some re-arrangement. The assignments of BSs to those LAs are released, and therefore those LAs have no more BSs assigned to them from BSC^* . Moreover, for those “released” BSs, an adequate number of new LAs are created (if the capacity of a new LA is full, then another LA is created).

3) Changing a BS-to-LA assignment: Without affecting the BS-BSC connection, a new LA assignment is carried out by searching the available LAs residing within the same BSC. Firstly a BS is chosen randomly to change its LA assignment. Then, a candidate LA is randomly chosen among all the LAs residing within the connected BSC. After the capacity constraint of the LA-BS connection is checked, the new LA is assigned to the BS.

The three moves listed above alter the network architecture in dissimilar ways. For example, a change in the BSC-MS C connection may have a large impact on the cost of the network. Therefore, the particular choice of a move type, when a new neighbor solution is to be created, is not uniformly distributed among these three. The move types are selected according to some probability distribution, which has a great impact on the performance of the SA algorithm. The effect of different probability assignments on the quality of results, and the computation time will be investigated.

B. Cooling Schedule

A cooling schedule consists of three important components: setting the initial temperature, the decision of when and how to decrease the temperature, and the decision of when to stop the algorithm. The most efficient cooling schedule may be found through trial and error, and by observing the effect on both the quality of the resulting solution and the rate at which the process converges [9].

Initial temperature calculation is crucial, because if it is too high, then it may take too long to reach the result. On the other hand, if the initial temperature is taken too small, then the algorithm does not have the freedom of making sufficient number of random moves to visit different local minima and stay in a relatively deep one. The final result may depend on the initial topology [4]. The idea is that, at the initial stage of the algorithm, nearly all neighbors should be accepted. In other words, the initial probability of accepting a worse solution (P_0) should be very close to 1, e.g., $P_0 = 0.99$. We set our initial temperature (T_0) to achieve that probability through equation (17). Extracting T yields

$$T_0 = \frac{-\max \Delta E}{\ln P_0}. \quad (18)$$

To obtain the maximum change between the cost of neighbor solutions ($\max DE$ in equation (18)), a predefined number of neighbors starting with the initial solution is created and the maximum difference between the costs of each of the neighbors is calculated. Whenever a specified number of new solutions are accepted (which is an SA parameter), the temperature decreases according to the formula

$$T_{new} = T_{old} \alpha \quad (19)$$

The cooling factor, α , effects the quality of the solutions and the run times significantly. Typical values of α lie between 0.85 and 1.

Towards the end of the algorithm, the temperature goes down to zero, and it becomes difficult to accept new topologies.

For that reason, we have another criterion for deciding as to when the temperature should be decreased. At a specific temperature, if the total number of neighbors tested (not just the accepted ones) equals approximately to the neighborhood size, then the temperature is decreased using equation (19) again. This neighborhood size in our implementation is considered to be roughly equal to the multiplication of the number of BSs, BSCs, and MSCs in the network, since, at a specific topology, one has almost that many possible changes in the network.

The algorithm must stop if it cannot find better solutions anymore. In our implementation, the SA algorithm stops if we cannot accept new solutions in a predefined number of temperature decreases. This stopping criterion too affects the quality of results and the execution time of the algorithm significantly. In order to guarantee a finite run-time, the SA search is also halted if a very large number of solutions are tried. During the SA search, “the best solution recorded and presented when the algorithm stops.

IV. COMPUTATIONAL EXPERIMENTS

Platforms used for experiments are Linux and Windows systems with PII-350 processors. The cluster system named “ASMA” (Advanced System for Multicomputer Applications) is also employed for experiments. ASMA resides in the Computer Engineering Department of Bođaziçi University, and has 24 personal computers (PCs) with Linux operating system [2]. The execution time varies with the size of the problem and SA parameters. In order to find the SA parameter values that give the best result, the first group of experiments are performed using different parameter values on different data sets. Experiments are repeated 30 times for each data and parameter set. The data sets are obtained from an operational GSM network in Istanbul at different times of the week. By choosing a pilot area from the GSM network, the information about the network elements (BS, BSC, and MSCs) is extracted. The number of these network elements defines the problem size.

In the second group of experiments, the performance of simulated annealing is compared to greedy search (with random initial topology) and random generation techniques. In the sequel, the total registration cost of the network will have the unit “total number of hand-offs incurred between the LAs per second”.

A. Typical Run of the SA Algorithm

Fig. 1 shows the cost of the solutions found at different stages of a typical SA run for a moderate sized example network which has about 600 BSs, 6 BSCs and 6 MSCs. The SA parameter optimization is performed with this network and verified by others. At the initial stages, since the temperature is very high, the SA algorithm accepts nearly all solutions. It acts as random search first, and the cost of the accepted solutions varies in a wide range as seen in Fig. 1. As the temperature decreases, the probability of accepting worse solutions also

decreases. Hence, at later stages of the run, the search becomes greedy and only better solutions are accepted.

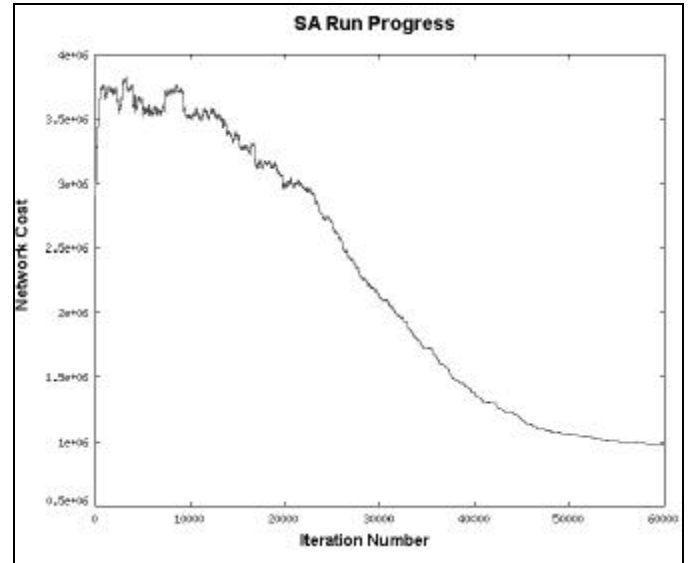


Fig. 1. A typical run of the SA algorithm on an example network. The registration cost of the network is computed in total number of hand-offs incurred between the LAs per second.

B. Effect of the SA Parameters

To find the optimal SA parameter values, different parameter sets are tested on several sample problems. The figures present result of applying different values for each SA parameter on a sample problem in which the network consists of 600 BSs, 6 BSCs, and 6 MSCs. After 30 executions of the SA algorithm on the sample problem, where the longest run-time was about 1.5 hours, the minimum, average and the maximum values are determined.

The first SA parameter is the number of neighbors generated starting with the initial solution to calculate the initial temperature according to equation (18). This parameter affects only the initial temperature, since the larger the number of neighbors traced, the greater the possibility of finding higher cost differences between two neighbors, and hence, the higher the initial temperature as seen in Fig. 2. Therefore, to be able to start in a hot system (a system accepting nearly every neighbor solution), this parameter should be set to a high value such as 100,000.

The second parameter is α which is instrumental in decreasing the temperature of the system as per equation (19). The higher its value, the slower the cooling down of the system. By setting the α value high, larger portion of the solution space can be searched, but run-time takes longer. Fig. 3 displays that the value 0.9999 performs better than both 0.99 and 0.999.

The number of accepted solutions that decrease the temperature (AD) is the third parameter considered. If the AD value is small, then the SA algorithm converges faster. The values assigned to this parameter range between 10 and 30 in

our experiments. As Fig. 4 shows, although the particular choice of AD does not have much effect on the quality of the results, AD = 20 produces slightly better results.

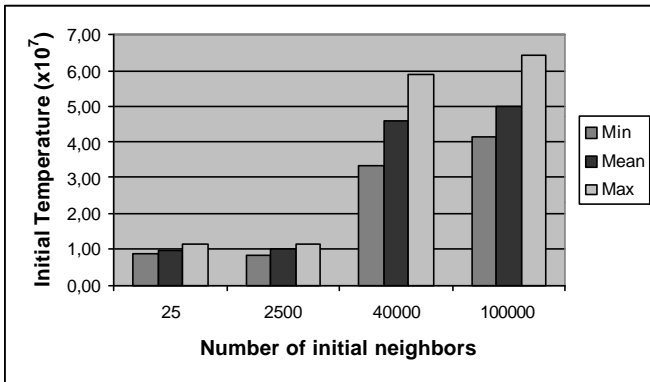


Fig. 2. The effect of the number of initial neighbors on the initial temperature.

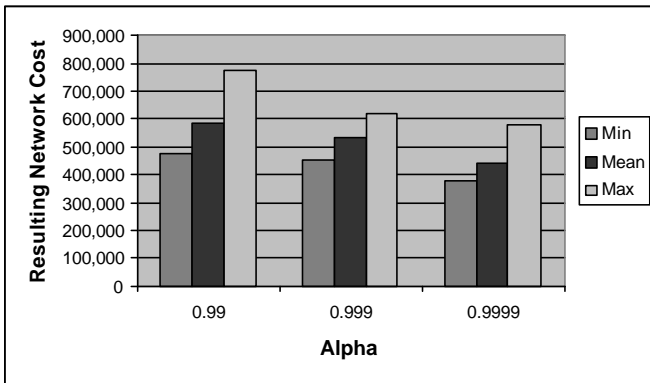


Fig. 3. The effect of the cooling factor α on the registration cost of the network computed in total number of hand-offs incurred between the LAs per second.

A run is ended if a specified number of temperature decrements are made without any improvement in the cost (denoted by WI), or if the number of neighbors tested exceeds an iteration limit IL. These parameters do not affect the quality of the resulting solution directly. However, if a run is stopped before the system is cooled down enough, better solutions may be missed. On the other hand, unnecessarily high WI or IL value assignments result in longer run times. In the experiments, these parameters values are selected such that the algorithm converges in a reasonable time. According to Figures 5 and 6, WI and IL should respectively take on values around 5,000 and 5,000,000.

V. COMPARISON OF THE SA RESULTS WITH OTHER TECHNIQUES

Both SA and greedy search (GS) are neighborhood search algorithms, and are natural competitors. Among feasible neighbors, GS starts with a feasible random solution. If this neighbor improves the cost of the current solution, then it is

accepted as the current solution. The search continues until an “iteration limit” is reached, or a “specified number of iterations” result in no improvement.

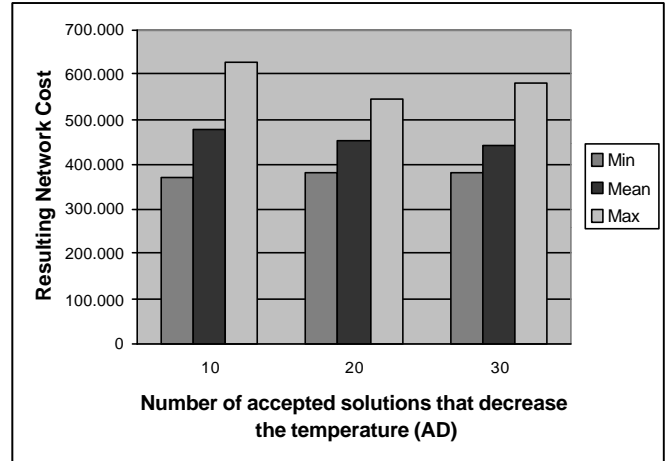


Fig. 4. The effect of the number of accepted solutions that decrease the temperature (AD) on the registration cost of the network computed in total number of hand-offs incurred between the LAs per second.

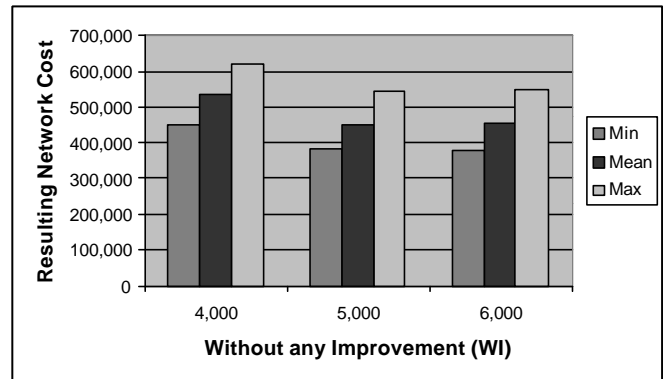


Fig. 5. The effect of the specified number of temperature decrements allowed without any improvement in the registration cost of the network.

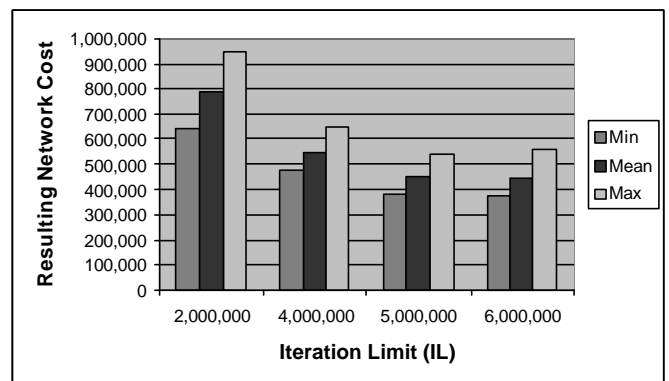


Fig. 6. The iteration limit IL on the number of neighbors tested plotted against the registration cost of the network computed in total number of hand-offs incurred between the LAs per second.

The performance of GS should form an upper bound on SA, since the greedy search can only find the local minima of the solution space. However, the quality of the GS results depends heavily on the initially formed solution. Therefore, a number of runs are made to test the GS with different initial solutions. The solution with the least network cost, computed as the average over multiple GS runs, is extracted from the experiments.

A statistical quality evaluation method is also applied to the SA solutions. In order to have a feeling about the range of the solution space, more than 15,000 random solutions are generated, a histogram of these solutions are prepared and then compared to the SA solutions. The so-called random generation (RG) methods are based on the procedure that first BSs are connected randomly to feasible BSCs (subject to the constraints), and then BSCs are connected to feasible MSCs. In the first approach (called RG1), for each MSC starting with one LA, BSs are assigned to an LA. If the LA capacity (paging capacity of BSs) reaches its limit, then a new LA is created and the remaining BSs are assigned to that LA. Here, the aim is to create the minimum number of LAs for each MSC (after randomly establishing the BS-BSC-MSC topology). In a way, RG1 is not truly random, and it possesses some minimization effort.

In the other method (called RG2), a random number of LAs are created for each MSC. The randomization is based on the uniform distribution with upper limit being the number of BSs connected to the BSCs of the MSC. Then, all BSs are assigned to a random feasible LA. If for a BS, no feasible LA exists, then a new LA is created, and the BS is assigned to it. The goal is to make all of the network topology decisions randomly and therefore to simulate the solution space more accurately.

A histogram obtained from approximately 27,000 solutions via the RG1 method on the example problem described in Section IV can be seen in Fig. 7. The figure depicts that the distribution is bell-shaped, resembling the normal distribution.

The comparison between SA and the other techniques are based on the resulting network costs. Three pilot areas in Istanbul are considered in the experiments corresponding to data sets numbered one, two to five, and six, respectively. The data for the second pilot area are collected at different times of a week.

The number of LAs cannot be less than the number of MSCs in a cellular system. In the RG2 approach, because a random number of LAs are created for each MSC, the total number of LAs in the system deviates considerably from the minimum possible number of LAs. For instance, if there are 6 MSCs in the system and the number of BSs is 100 per BSC on average, then the minimum and maximum number of LAs that can be created is 6 and 100, respectively. Because LAs are created randomly, the probability of having close to 6 LAs is very small. As a result, many LAs trigger a rise in the registration cost, and the RG2 solutions accumulate to the right side of Fig. 8. The histogram is established with approximately 16,000 RG2 runs.

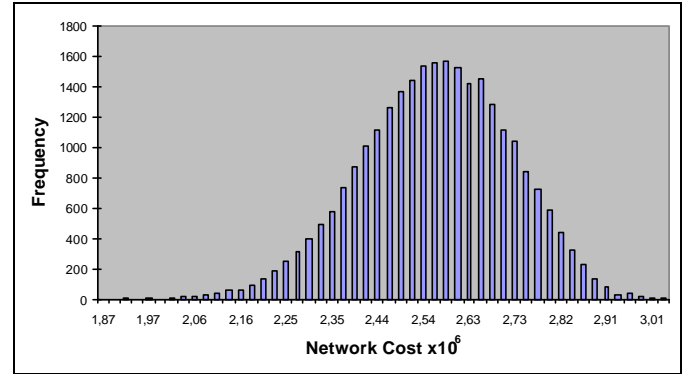


Fig. 7. Histogram of the RG1 approach. The registration cost of the network computed in total number of hand-offs incurred between the LAs per second.

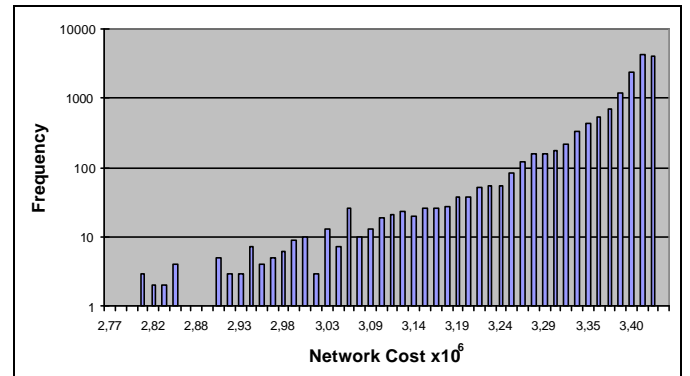


Fig. 8. Logarithmic histogram of the RG2 approach. The registration cost of the network computed in total number of hand-offs incurred between the LAs per second.

The number of SA runs performed on each data set is 10, and the value shown in the Table 1 is the average of the resulting network costs. The GS algorithm is run 30 times for each data set, and the minimum network cost is displayed in Table 1. RG1(Min) and RG2(Min) columns contain the minimum network costs found in the runs performed to establish the RG histograms. Table 1 indicates that the SA-based solution technique outperforms the other methods.

Table 1. Network cost (computed in total number of hand-offs incurred between the LAs per second) induced by SA, GS, RG1 and RG2 for six data sets.

Data Set Number	SA (Mean)	GS (Min)	RG1 (Min)	RG2 (Min)
1	450,882	792,769	1,870,809	2,772,498
2	12,161	24,693	134,410	161,974
3	15,352	27,930	138,889	161,365
4	11,514	20,839	109,852	131,300
5	7,308	14,910	84,571	101,962
6	901	1,956	22,402	31,508

To have an appreciation as to how far the SA results deviate from the mean of the solution space, RG methods are

employed. In Table 2, the mean value of the SA results are compared to the mean and standard deviation of RG1 runs.

Table 2. Network cost (computed in total number of hand-offs incurred between the LAs per second) induced by SA and RG1 for six data sets.

Data Set Number	SA (Mean)	RG1 (Mean)	RG1 (Std. Dev.)
1	450,882	2,572,020	156,555
2	12,161	150,569	3,220
3	15,352	153,915	3,305
4	11,514	122,772	2,493
5	7,308	94,253	1,845
6	901	29,937	1,692

Table 3. Network cost (computed in total number of hand-offs incurred between the LAs per second) induced by SA and RG2 for six data sets.

Data Set Number	SA (Mean)	RG2 (Mean)	RG2 (Std. Dev.)
1	450,882	3,379,986	61,265
2	12,161	178,509	1,993
3	15,352	182,433	2,034
4	11,514	145,526	1,636
5	7,308	111,731	1,206
6	901	43,621	1,527

RG2 solutions represent the actual feasible solution space. Finally, in Table 3, the mean value of SA results are compared to the mean and standard deviation of the RG2 runs. For both methods and for all the sample problems, the difference between the cost of the SA solution and the mean cost of the RG solutions is much more than several standard deviations. This huge difference proves that SA beats any random solution by a large margin. In other words, in a reasonable time, the probability of randomly creating a solution better than the SA solution is very close to zero. We accept this as a statistical measure showing the high quality of SA results.

VI. CONCLUSIONS AND FUTURE WORK

In order to design a feasible cellular network, constraints related to the call handling capacities of network elements and costs related to the paging and registration activities should be considered. Utilizing the available network information in a realistic manner, we formulate an optimization problem for the location area planning and the assignment of cells to switches. The formulation is very detailed, and it includes the majority of the previously considered problems as special cases. A solution technique based on simulated annealing is proposed. The implementation details and empirical parameter tuning of the algorithm are described. Computational experiments are performed using real data sets. The SA results are promising, since for the cases studied, the quality of SA results are much better than those obtained by multiple runs of greedy search. Moreover, the statistical quality evaluation methods show that

it is almost impossible to create a random solution better than that offered by SA in tolerable run-times.

As future work, we plan to develop a heuristic algorithm that will be aimed to solve the formulated optimization problem. The SA-based program could also be run on a wider test set to check the suitability of the currently proposed SA parameter values.

REFERENCES

- [1] I. F. Akyildiz and J. S. M. Ho, "A mobile user location update and paging mechanism under delay constraints", *Proceedings of ACM SIGCOMM'95*, Cambridge, Massachusetts, September 1995, pp. 244-255.
- [2] Bođazići University ASMA Project, <http://www.asma.cmpe.boun.edu.tr>, 2000.
- [3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco, California: W. H. Freeman, 1979.
- [4] B. Hajek, "Cooling schedules for optimal annealing", *Mathematics of Operations Research*, Vol. 13, No. 2, pp. 311-329, May 1988.
- [5] D. S. Johnson, C. R. Aragon, L. A. McGeoch and C. Schevon, "Optimization by simulated annealing: an experimental evaluation, part 1: graph partitioning", *Operations Research*, Vol. 37, No. 6, pp. 865-892, November-December 1989.
- [6] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by simulated annealing", *Science*, Vol. 220, No. 4598, pp. 671-680, May 1983.
- [7] Z. Mao and C. Douligeris, "Two location tracking strategies for PCS systems", *Proceedings of the 8th International Conference on Computer Communications and Networks*, Boston, Massachusetts, October 1999, pp. 318-323.
- [8] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, "Equation of state calculations by fast computing machines", *Journal of Chemical Physics*, Vol. 21, No. 6, pp. 1087-1092, June 1953.
- [9] E. Rich and R. Knight, *Artificial Intelligence*, 2nd Edition, McGraw-Hill, 1991.
- [10] C. Rose, "State-based paging/registration: a greedy technique", *IEEE Transactions on Vehicular Technology*, Vol. 48, No. 1, pp. 166-173, January 1999.
- [11] C. U. Saraydar, O. Kelly, C. Rose, "One-dimensional location area design", *IEEE Transactions on Vehicular Technology*, Vol. 49, No. 6, November 2000, in press.
- [12] C. U. Saraydar and C. Rose, "Location area design using population and traffic data," *Proceedings of the 32nd Annual Conference on Information Sciences and Systems*, Vol. II, Princeton, New Jersey, March 1998, pp. 739-744.
- [13] S. Tabbane, "An alternative strategy for location tracking", *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 5, pp. 880-892, June 1995.