# Designing Cellular Mobile Networks Using Non–Deterministic Iterative Heuristics

Sadiq M. Sait, Marwan H. Abu–Amara, Abdul Subhan

*Computer Engineering Department*
*King Fahd University of Petroleum & Minerals*
*Computer Engineering Department*
*Dhahran–31261, Saudi Arabia*

**Abstract**

Network planning in the highly competitive, demand-adaptive and rapidly growing cellular telecommunications industry is a fairly complex and crucial issue. It comprises collective optimization of the supporting, switching, signaling and interconnection networks to minimize costs while observing imposed infrastructure constraints. This work focuses on the problem of assigning cells to switches, which comprise the Base Station Controller and Mobile Switching Center, in a cellular mobile network. As a classic instance of the NP-hard Quadratic Assignment Problem (QAP), deterministic algorithms are incapable of finding optimal solutions in the vast complex search space in polynomial time. Hence, a randomized, heuristic algorithm, such as Simulated Evolution is used in this work to optimize the transmission costs in cellular networks. The results achieved are compared with existing methods available in literature.

*Key words:* Network planning, Cellular Mobile Network, Assignment, Quadratic Assignment Problem, Heuristics, Evolutionary Heuristics, Soft Computing.

## 1 Introduction

Mobile cellular systems have become a ubiquitous component of telecommunication technologies. The growth in this sector is driven by an ever-increasing subscriber demand and the potential for data technologies in 3G and 4G systems. These networks provide users freedom of mobility and ease of use, while

*Email address:* {sadiq,marwan,subhan}@ccse.kfupm.edu.sa (Sadiq M. Sait, Marwan H. Abu–Amara, Abdul Subhan).

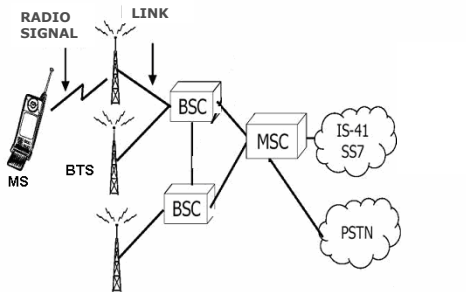



Fig. 1. Basic architecture of cellular mobile network.

maintaining optimum conditions of quality and price. In other words, they must provide a system of communication that the customer perceives to be as efficient, simple, fast, reliable and as cheap as possible. The basic difference between a fixed network and a mobile network is the medium of subscriber access. In the former, the medium is by subscriber loop, while a mobile network uses radio technology [1].

Though cellular networks are inherently scalable [2], managing the exponential growth is a tremendous task. The growing subscriber base, scarce existing network resources and intense competition in the telecommunication market place an ever-increasing emphasis on more efficient and 'demand-adaptive' network design for the cellular network providers. Further, with the upcoming applications for data communications on these networks as advocated by 3G and 4G systems, there is a critical need for optimization of all aspects, including efficient and flexible network structures [3].

The switching architecture of a cellular network follows a hierarchical approach and is composed of three main types of elements:

(1) The Base Transceiver System (BTS)
(2) The Base Station Controller (BSC)
(3) The Mobile Switching Center (MSC)

These components are illustrated in Figure 1, where a Mobile Station (MS) is in communication with the above three mentioned elements. The following paragraphs describe the functions of the MS, the BTS, the BSC, and the MSC.

**Mobile Station (MS):** It is the user mobile terminal that allows users to communicate, and also provides means of interactions and control between a user and the network.

**Base Transceiver System (BTS):** The Base Transceiver System is the entity corresponding to the site communicating with the MS. Usually, the BTS will have an antenna with several radio transceivers each of which communicates on a radio frequency. The link-level signaling on the radio-channels is

interpreted here, whereas most of the higher-level signaling is forwarded to the BSC and the MSC.

**Base Station Controller (BSC):** Each Base Station Controller (BSC) controls several BTSs. It takes care of a number of different procedures regarding call setup, location update and handover for each MS.

**Mobile Switching Center (MSC):** The Mobile Switching Center is a normal ISDN (Integrated Services Digital Network) switch with extended functionality to handle mobile subscribers. Its basic function is to switch voice and data connections between BSCs, other MSCs, other wireless networks, and external non-mobile-networks. The MSC also handles a number of functions associated with mobile subscribers, such as registration, location updating and handover. Normally there exist only a few BSCs per MSC, due to the large number of BTSs associated with the former.

In this paper, following the convention by Pierre and Houetto [4], the combination of BSC and MSC is referred to as a switch. In a typical cellular network, the area of coverage is often geographically divided into hexagonal cells. Each of these contains a BTS covering a small geographic area [5]. The BTS supplies the radio interface to mobile service users within its coverage area and is controlled by a switch.

The complete design of the mobile service terrestrial network includes the location of the switches, the allocation of BTSs to these locations, the interconnection layout and dimensioning of the links between them, and their connection to the public network. Optimization of the transmission infrastructure (access network) used to connect base stations with the switches can have considerable cost savings for a mobile telephone operator. One of the largest infrastructure expenditures is transmission - here, the costs are chiefly generated by the interconnection of BTSs with the rest of the terrestrial network and as such, this requires careful planning for optimization.

This work focuses on the process of planning the network and optimizing the interconnection of base stations with the other elements of the terrestrial network. This NP-hard problem has no exact algorithms that can achieve desired results in acceptable runtimes, and hence we engineer a heuristic such as Simulated Evolution (SimE) to achieve optimization goals, while conforming to design constraints imposed by the network planner.

Section 2 presents a survey of literature related to the cellular network design. In particular, the application of iterative algorithms to the above problem is reviewed. In Section 3, the cell-to-switch assignment is formulated as a multiobjective problem. The cost functions related to the length of link and hand–off are designed along with switch capacity constraints. Section 4 discusses the details of the proposed SimE algorithm and the setting of various parameters.

Other heuristics, such as Simulated Annealing and Tabu Search which are used for performance comparison purposes are also reviewed. Experimental results for varying sizes of data sets are discussed and compared in Section 5 along with the effects of different parameters on the quality of the final solution. The work and results are summarized in the conclusive Section 6.

## 2 Related Work

A reasonable amount of work has been published which addresses the problem of designing cellular networks; in particular the assigning of cells to switches. Merchant and Sengupta [6] attempted to solve the problem using deterministic algorithms and provided a basic formulation. They considered a scenario of assigning cells to the switches of a Personal Communication Services (PCS) network in an optimum manner and approached the problem from the perspective of Integer Programming. Their work proposed three heuristic solutions, two of which performed extremely well.

Pierre and Houeto [4] extended the above work, solving the same problem with varying sizes (in terms of number of cells and switches) using Tabu Search, a non-deterministic iterative algorithm. Their approach defines a series of moves applicable to an initial solution in order to improve the cost and establish its feasibility. For this purpose, they identified a gain structure with update procedures to efficiently choose the best solution in the current neighborhood. The implementation was tested with different parameters for Tabu Search. They also compared these results against those obtained by using Simulated Annealing [7], another popular non-deterministic iterative algorithm.

Menon and Gupta [6] improved upon the work of Pierre and Houeto [4] and obtained results in lesser time [5]. According to them, in the presence of capacity constraints at the switches, the problem of assigning cells to switches becomes a difficult one to solve, with all effective solution approaches being based on heuristic techniques. They presented a hybrid heuristic named Price Influenced Simulated Annealing (PISA), which integrates ideas from linear programming into a simulated annealing framework. Extensive computational results were presented comparing the performance of the heuristic with the lower bound obtained from linear programming relaxation. These results indicated that the PISA procedure is extremely efficient.

A Memetic Algorithm (MA) was recently proposed by Quintero and Pierre [8] for assigning cells to switches in cellular mobile networks. Its implementation was subjected to extensive tests, which confirm the efficiency and the effectiveness of MA in providing good solutions for moderate and large-sized cellular mobile networks, in comparison with Tabu Search and Merchant and

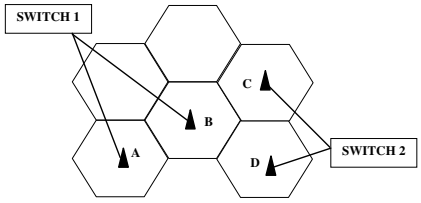



Fig. 2. Geographic division in a cellular network and hand–off between switches.

Sengupta's heuristics.

Shyu et al. [9] implemented an algorithm based on the Ant Colony Optimization (ACO) for solving the problem of cell assignment in PCS networks. It is a metaheuristic inspired by the foraging behaviors of ant colonies. The problem is modeled as a form of matching problem in a complete bipartite graph. Experimental results show that the proposed algorithm is an effective and promising approach with reasonable run times. Similar work related to the problem of cellular mobile network design has been carried out recently and the details are available in literature [10][11][12].

In this paper, we engineer Simulated Evolution (SimE) to address the problem of cells to switch assignment by optimizing the link cost and hand–off cost. The design approach for defining the goodness measure is the core of the algorithm, which is presented in Section 4.2. The results obtained from SimE are compared with other two heuristics, i.e., Simulated Annealing and Tabu Search.

## 3 Problem and Cost Function Formulation

In this section, the cell-to-switch assignment problem for cellular mobile networks is formulated. The objective is optimization of link cost, and hand–off cost, while the switch capacity is a constraint.

### *3.1 Problem Description*

In a cellular network, the area of coverage is often geographically divided into hexagonal cells. These cells are hierarchically set to reduce link costs, as illustrated in Figure 2. A certain number of cells are chosen to install switches that communicate with one another and serve as relays for communication between any pair of cells. For various reasons, particularly mobility, switches serving as relays to a given user could change if the user moves from his current cell. The operation that consists of detecting that a user has changed a cell and carrying out the required updates constitutes a hand–off.

When a hand–off occurs between two cells linked to the same switch, it is called a simple hand–off, because there are few necessary updates. On the other hand, a complex hand–off involves two cells associated with different switches. In this case the update procedures consume more resources than those required for a simple hand–off. For example, in Figure 2, a user who moves from cell B to cell A causes a simple hand–off. The network's database that keeps in memory the switch ID (managing each user) does not need an update. Only Switch 1 is used for this procedure and no other network entity intervenes. However, if a user moves from cell B to cell C, a complex hand–off occurs where Switches 1 and 2 have to exchange information on the user, and the database must also be updated. Furthermore, if Switch 1 is in charge of the billing, the hand–off cannot simply replace Switch 1 with Switch 2. Communications between the two switches continue to be relayed through Switch 1 even after the hand–off. Thus, we would have a connection to Switch 2, then to Switch 1, and finally to the network. Thus the cost of such a complex hand–off is higher. When the frequency of such hand–offs between cell B and cell A (Figure 2) is very high, while the hand–off frequency between cell B and cell C is low, it becomes reasonable to connect cells A and B to the same switch.

Thus, the problem of cell assignment could be summarized as follows:

*For a set of cells and switches (whose positions are known), assign the cells to the switches in a way that minimizes the cost function. The cost function integrates a component of link cost and a component of hand–off cost. The assignment must take into account the switch's capacity constraints that permit hosting only a limited number of calls.*

The following section gives a mathematical formulation to this problem of assigning cells to switches in a cellular network. This formulation is based on conventional methods where only the complex handover cost and link cost between cells and switches with respect to the maximum switch capacity constraints is considered.

### *3.2 Formulation Of Cost Function*

The formulation of cost function presented in this section is based on the work by Pierre and Houeto [4].

Let $n$ be the number of cells to be assigned to $m$ switches. Assume that the location of cells and switches are fixed and known. Let $H_{ij}$ be the cost per unit of time for a simple handover between cells $i$ and $j$ involving only one switch, and $H'_{ij}$ the cost per time unit for a complex handover between cells $i$ and $j$ $(i, j = 1, 2 \ldots, n \quad with \quad i \neq j)$ involving two switches. $H_{ij}$ and $H'_{ij}$ are

proportional to the handover frequency between cells $i$ and $j$ that could be measured or estimated. Let $C_{ik}$ be the amortization cost of the link between cell $i$ and switch $k$ ($i = 1, 2 \ldots, n \quad and \quad k = 1, 2, \ldots, m$) and $\lambda_i$ the number of calls per time unit destined to cell $i$. The capacity of a switch $k$ is denoted by $M_k$.

Let us define:

$$x_{ik} = \begin{cases} 1 \textit{ if cell } i \textit{ is related to switch } k \\ 0 \textit{ otherwise} \end{cases}$$

$$z_{ijk} = x_{ik}x_{jk} \qquad for \quad i, j = 1, 2 \ldots, n$$

$$and \quad k = 1, 2, \ldots, m \quad with \quad i \neq j$$

$$y_{ij} = \sum_{k=1}^{m} z_{ijk} \qquad for \quad i, j = 1, 2 \ldots, n \quad with \quad i \neq j$$

$z_{ijk}$ is equal to 1 if cells $i$ and $j$, with $i \neq j$, are both connected to the same switch $k$, otherwise $z_{ijk}$ is equal to 0. $y_{ij}$ takes the value 1 if cells $i$ and $j$ are both connected to the same switches and the value 0 if cells $i$ and $j$ are connected to different switches.

The cost per time unit $f$ could be calculated as:

$$f = \sum_{i=1}^{n}\sum_{k=1}^{m} c_{ik}x_{ik} + \sum_{i=1}^{n}\sum_{j=1, i\neq j}^{n} H'_{ij}(1 - y_{ij}) + \sum_{i=1}^{n}\sum_{j=1, i\neq j}^{n} H_{ij}y_{ij} \tag{1}$$

The first term of equation represents the link cost, while the second and third terms take into account the complex handover and the simple handover costs respectively. We should keep in mind that the cost function is quadratic in $x_{ik}$ as $y_{ik}$ is a quadratic function of $x_{ik}$.

The assignment of cells to switches is subject to a certain number of constraints. Each cell must be assigned to only one switch, which is translated to:

$$\sum_{k=1}^{m} x_{ik} = 1 \qquad for \quad i = 1, 2 \ldots, n \tag{2}$$

If $\lambda_i$ denotes the number of calls per time unit destined to cell $i$, the limited

capacity of switches imposes the following constraint:

$$\sum_{i=1}^{n} \lambda_i x_{ik} \leq M_k \qquad for \quad k = 1, 2 \ldots, m \tag{3}$$

According to the above constraint the total load of all cells which are assigned to the switch must be below the capacity of the switch. Finally, the constraints of the problem are completed by:

$$x_{ik} = \quad 0 \quad or \quad 1, \quad for \quad i = 1, 2 \ldots, n \quad and \quad k = 1, 2 \ldots, m \tag{4}$$

$$z_{ijk} = x_{ik} x_{jk} \quad and \quad i, j = 1, 2 \ldots, n \quad and \quad k = 1, 2 \ldots, m \tag{5}$$

$$y_{ij} = \sum_{k=1}^{m} z_{ijk} \qquad for \quad i, j = 1, 2 \ldots, n \tag{6}$$

The problem which is simplified to (1) under (2)–(6), cannot be solved using a standard technique such as linear programming as constraint (5) is not linear. Merchant and Sengupta [6] replaced it by the equivalent set of constraints:

$$z_{ijk} \leq x_{ik} \tag{7}$$

$$z_{ijk} \leq x_{jk} \tag{8}$$

$$z_{ijk} \geq x_{ik} + x_{jk} - 1 \tag{9}$$

$$z_{ijk} \geq 0 \tag{10}$$

The problem can be further simplified by proposing:

$$h_{ij} = H'_{ij} - H_{ij} \tag{11}$$

where $h_{ij}$ refers to the reduced cost per time unit of a complex handover between cells $i$ and $j$. Objective function (1) could be rewritten as follows:

$$f = \sum_{i=1}^{n} \sum_{k=1}^{m} c_{ik} x_{ik} + \sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} h_{ij}(1 - y_{ij}) + \underbrace{\sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} H_{ij}}_{constant}$$

The assignment problem thus takes the minimization of the following cost function subject to constraints (2)–(4) and (6)–(10):

$$f = \sum_{i=1}^{n} \sum_{k=1}^{m} c_{ik} x_{ik} + \sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} h_{ij}(1 - y_{ij}) \tag{12}$$

In this form, the assignment problem could be solved by usual programming methodologies. However, due to the complexity of the problem, exact methods cannot scale with instances involving more than a few cells and switches [6]. Consequently, non–deterministic heuristic methods are applied that search the solution space efficiently and within acceptable runtimes.

### *3.3 Inclusion of Additional Constraints*

Though the problem formulation is formally complete, an additional constraint is defined to address the limitations imposed by the number of ports available on each switch. This has immense practical significance as the MSC, in a real-life cellular mobile network, is limited not only by its call processing capability, but also by the number of ports present. In certain scenarios though the switch may have enough processing capability, it may have exhausted the number of available ports, thus effectively causing congestion.

If $P_k$ denotes the number of ports available on each switch $k$, then the constraint on the number of ports may be represented as follows:

$$\sum_{i=1}^{n} x_{ik} \leq P_k \qquad for \quad k = 1, 2 \ldots, m \tag{13}$$

where $x_{ik}$ is as defined in Section 3.2. For this modified model, the problem then is to solve Equation (12) subject to constraints (2)–(4) and (6)–(10) as well as (13).

## 4 Proposed Approach

### *4.1 Simulated Evolution (SimE)*

According to the theory of evolution the more an organism adapts to its environment, the better are its chances of survival. Hence, adaptation is seen as a form of optimization. This similarity has given rise to a new class of randomized, evolution-inspired, iterative algorithms such as *Genetic Algorithms, Simulated Evolution*, and *Stochastic Evolution*. For these heuristics, the cost function is an estimation of the degree of adaptation of a particular solution to the target objective. In this work, SimE is engineered to traverse the search space to find the optimal solution in terms of both minimal cost and runtime. The results are compared against other reported algorithm, with a particular focus on Simulated Annealing.

```
ALGORITHM Simulated_Evolution(B, Φ_initial, StoppingCondition)
NOTATION
B= Bias Value.   Φ= Complete solution.
m_i= Module i.   g_i= Goodness of m_i.
ALLOCATE(m_i, Φ_i)=Function to allocate m_i in partial solution Φ_i
Begin
Repeat
  EVALUATION:
      ForEach m_i ∈ Φ evaluate g_i;
      /* Only elements that were affected by moves of previous */
      /* iteration get their goodnesses recalculated*/
  SELECTION:
      ForEach m_i ∈ Φ DO
          begin
            IF Random > Min(g_i + B, 1)
            THEN
              begin
                P_s = P_s  ∪  m_i; Remove m_i from Φ
              end
          end
    Sort the elements of S
  ALLOCATION:
      ForEach m_i ∈ S DO
          begin
            ALLOCATE(m_i, Φ_i)
          end
Until  Stopping Condition is satisfied
Return Best solution.
End (Simulated_Evolution)
```

Fig. 3. Structure of the Simulated Evolution algorithm.

SimE is a general, non-deterministic, iterative meta-heuristic proposed by Kling and Banerjee in 1987 [13] to solve combinatorial optimization problems. The general SimE algorithm is illustrated in Figure 3 and comprises three main steps namely **Evaluation**, **Selection**, and **Allocation**. The algorithm starts with an initial solution or assignment, which it seeks to improve from one iteration or generation to the next by following an evolutionary based approach. It works with a single solution, that is referred to as the population, consisting of movable cells or elements. Each of these is associated with a certain *goodness* measure - a metric strongly correlated to the overall solution fitness, which is indicative of how optimally the cell is placed.

In the **Evaluation** step, the **goodness** of each cell, in the range $[0, 1]$, is measured at its current location. The goodness is an approximate indicator of how near a cell is to its optimum location. In **Selection** step unfit cells are selected probabilistically for relocation and this is based on their goodness value. Higher the goodness value, lower is the chance of it being selected. These selected cells, comprising the *selection set* $P_s$, are removed from the current solution and reassigned one at a time to new locations in a constructive **Allocation** step, thereby increasing the overall goodness. SimE executes these three basic steps or procedures in sequence until the average *goodness* of the population reaches a maximum value, or no significant improvement to the *goodness* is observed after a number of iterations.

Another deciding parameter crucial to the success of SimE is the bias value $B$

which is used in the *Selection* function. It is not possible to find the accurate goodness value for a cell because its exact optimum location is unknown. Further, goodness only gives the local view of cell, and in order to compensate the error in goodness calculation and to limit the size of selection set, a bias parameter is used [14], where a cell $i$ is selected if $goodness_i + B < Random$, where $Random$ is a uniformly distributed random number in the range $[0, 1]$, $B$ is the selection bias with its typical values in the range $[-0.2, 0.2]$, and $goodness_i$ is the goodness value of cell $i$. In this work, a bias value of 0.2 was chosen through extensive trials.

### *4.2 Goodness Function*

The goodness function, which is calculated for each cell in an assignment, is a core component unique to SimE and is strongly correlated to the overall fitness or cost optimization. The algorithm iteratively improves on the goodness of each cell, thereby navigating the search space towards better quality solutions.

A simple goodness function is defined for the SimE algorithm, which comprises the link cost and hand–off cost, following an approach similar to that used for the cost function in the problem formulation. It is formulated as:

$$goodness = \frac{O(C_l + C_h) + \varepsilon}{C(C_l + C_h) + \varepsilon}$$

where $C_l$ and $C_h$ are the link and hand–off costs respectively. Here, $O(C_l + C_h)$ is the optimal value for the cell while $C(C_l + C_h)$ is its current value in that particular iteration. The optimal link cost is the lowest cost among all possible links from the cell to the switches. The optimal hand–off cost is the lowest possible cost incurred between neighboring cells, which is zero (i.e., there is no hand–off). These optimal values are computed only once, while the current value for $C(C_l + C_h)$ is computed in every generation.

This goodness function, though perceivably simple is validated and proven to be effective as demonstrated by Figures 4, 5 and 6. These figures, drawn from experimental results discussed later show the run-time trends exhibited by Simulated Evolution for a problem instance involving 100 cells and 5 switches. The first of these illustrates a gradual reduction in the size of the selection set (i.e., the number of cells selected for allocation) which reflects an increasing average goodness as shown in the second figure. This behavior along with the corresponding cost optimization in Figure 6 confirms the validity of the formulated goodness measure.

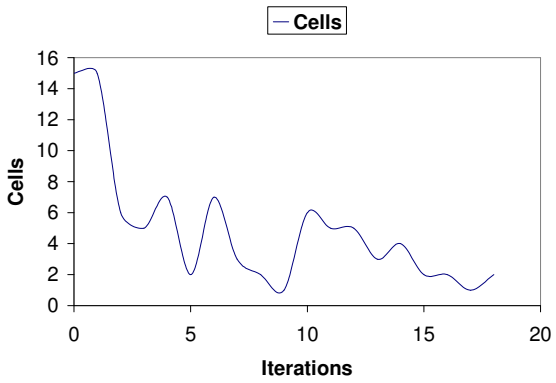



Fig. 4. Decrease in the Selection Set size for problem size (100–5).

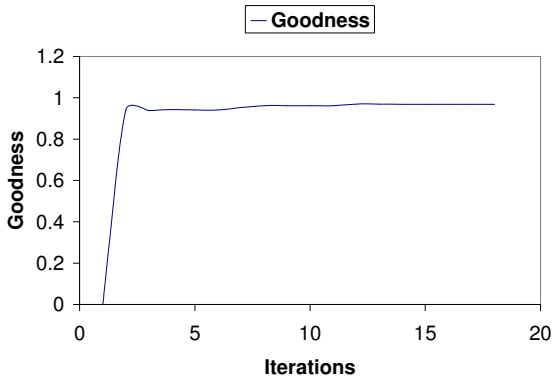



Fig. 5. Increase in the average cell goodness for problem size (100–5).

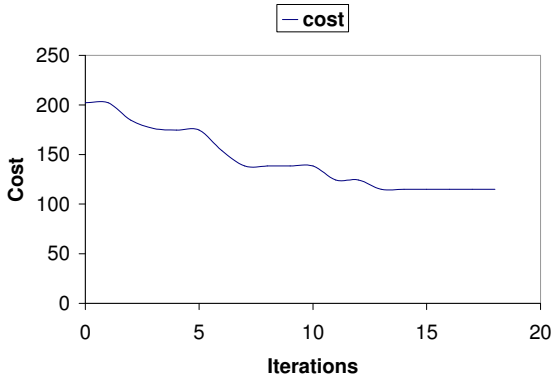



Fig. 6. Optimization of overall solution cost for problem size (100–5).

### *4.3 Simulated Annealing and Tabu Search*

Simulated Annealing is one of the most popular and general adaptive heuristics belonging to the class of *non-deterministic* algorithms [7]. It has been successfully applied for solving a vast number of combinatorial optimization problems. It is relatively easy to implement and also produces high quality solutions regardless of the choice of the initial configuration. The main components of the algorithm are the initial temperature $T_0$, the cooling rate $\alpha$,

constant $\beta$, and $M$, which represents the time until the next parameter update. The core procedure of the algorithm is the *Metropolis* function, which simulates the mettalurgical annealing process. SA transitions from an approximately random search process to a greedy, deterministic algorithm controlled by the changing temperature parameter and thus is able to navigate complex, multimodal search spaces and avoid local-optima. In-depth details and discussion of the general simulated annealing algorithm can be found in [7]. The values for these parameters were assigned after carrying out a number of trial runs for different parameter values and extensive tuning of parameters. The main parameters are initial temperature $T_0$=10000, number of iterations before temperature update $M$=10, and the constants $\beta$=1.0095 and $\alpha$=0.963.

**Tabu Search (TS)**: Tabu Search starts from an initial feasible solution and carries out its search by making a sequence of random moves or perturbations. A tabu list is maintained which stores the attributes of a certain number of previous moves. This list prevents taking the search process back to recently visited states [7]. In each iteration, a subset of neighbor solutions is generated by making a certain number of moves and the best move (the move that resulted in the best solution) is accepted, provided it is not in the tabu list. If the said move is in the tabu list, it is accepted only if it leads to a solution better than the best found so far (aspiration criterion). In each iteration, a number of neighbor solutions are generated by perturbing the solution: two cells are selected randomly and their assignments are interchanged. The number of such neighbor solutions generated in each iteration depends on the problem size. The size of tabu list also depends on the problem size and is usually kept at 10% of the total number of cells. In this work, short-term memory element was used for TS implementation. A straightforward aspiration criterion was employed where, if the current best solution is the best seen so far i.e., better than the global best, it is then accepted and the tabu restriction is overridden.

## 5 Experimental Results & Comparison

This section presents the experimental results achieved with the SimE heuristic and compares it against Simulated Annealing, Tabu Search and Pierre and Houeto's SA-P algorithm [4]. The data sets used were generated as explained in [4][6] and comprise of problem instances with number of cells and switches varying from 15 to 500 and 2 to 12 respectively. Twenty such sets were generated and a series of test runs were conducted to measure SimE performance against other algorithms, particularly in terms of the cost minimization achieved.

### 5.1 Comparison of Solution Cost and Runtimes

Table 1 compares the performance of SimE with Simulated Annealing as well as other algorithms in literature such as Tabu Search and SA-P [4]. The results demonstrate the algorithm's ability to navigate the search space more effectively, thereby achieving better quality (lower-cost) solutions, particularly for large size problems. The actual performance increment is tabulated in Table 2 in terms of the percentage improvement with SimE over the other three algorithms. The heuristic achieves 29-55% gain over SA-P, and in the range of 11-29% over TS. SimE also consistently outperforms SA, especially for the larger problem sets with a maximum gain of slightly over 32%.

Simulated evolution's performance over annealing can be attributed to the dynamics of each heuristic. SA spends most of its initial iterations approximating a random search process, while slowly gravitating towards a greedier approach. SimE, however, with its *goodness evaluation*, *selection* and *allocation* steps can be engineered to reach consistently higher quality solutions from the first few iterations onwards.

Another interesting comparison between SimE and SA is how the two algorithms scale with problem sizes. Figure 7 shows how the runtime for annealing increases exponentially with larger problem sizes, while SimE reports almost constant, and much lesser computation time to reach its target fitness.

Table 1
Comparison of SimE and SA with TS and Pierre's (SA-P) heuristics in terms of best cost achieved.

| Cells | Switches | SA-P | TS | SA | SimE |
|---|---|---|---|---|---|
| 15 | 2 | 123 | 118 | 97.8072 | 86.8706 |
| 30 | 3 | 405 | 324 | 257.1508 | 229.1357 |
| 50 | 4 | 851 | 580 | 486.8911 | 432.3942 |
| 100 | 5 | 1999 | 1234 | 1157.1383 | 1055.906 |
| 150 | 6 | 3240 | 2010 | 1943.6049 | 1701.811 |
| 200 | 7 | 5550 | 2768 | 2966.2921 | 2453.9762 |
| 250 | 8 | - | - | 3852.7231 | 3329.5575 |
| 300 | 9 | - | - | 5190.4683 | 4004.1733 |
| 350 | 10 | - | - | 6574.7308 | 4624.6573 |
| 500 | 12 | - | - | 11550.5805 | 7812.2196 |

### 5.2 Comparison with Additional Constraints

The results presented in the earlier section addressed SimE performance without including the effect of limitation imposed by number of ports available on

Table 2
Percentage improvement in SimE in relation to TS, Pierre's (SA-P), and SA heuristics.

| Cells | Switches | SimE vs. SA-P | SimE vs. TS | SimE vs. SA |
|---|---|---|---|---|
| 15 | 2 | 29.37 | 26.38 | 11.18 |
| 30 | 3 | 43.42 | 29.27 | 10.89 |
| 50 | 4 | 49.18 | 25.44 | 11.19 |
| 100 | 5 | 47.17 | 14.43 | 8.748 |
| 150 | 6 | 47.47 | 15.33 | 12.44 |
| 200 | 7 | 55.78 | 11.34 | 17.27 |
| 250 | 8 | * | * | 13.57 |
| 300 | 9 | * | * | 22.85 |
| 350 | 10 | * | * | 29.66 |
| 500 | 12 | * | * | 32.36 |
| | Average | 45.39 | 20.36 | 17.01 |

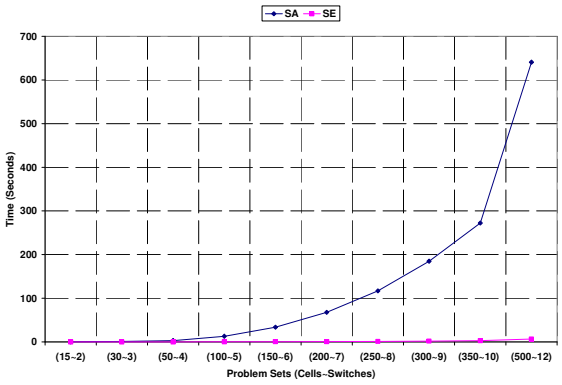
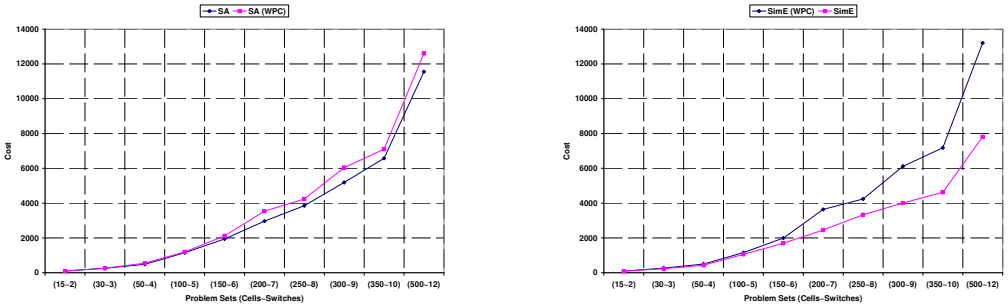



Fig. 7. Runtimes for SA and SimE with increasing problem sizes.

a switch. As mentioned previously, the processing capability of an MSC is not only limited by its own capacity, but also by this port constraint. This section reports on the performance of SimE and SA when this additional factor is addressed.

Table 3 compares the solution costs and the corresponding runtimes achieved with and without port constraint for both SA and SimE. The fitness optimizations achieved, as determined by the cost attained, are reasonably similar for both SA and SimE. However, the difference in runtimes is significant, with SimE requiring a fraction of that needed by Annealing.

However, another perspective to be considered is the overall effect of the additional constraint on each heuristic's behavior. Looking at the difference in the solution quality achieved with and without the port constraint for each algorithm, SimE is seen to be more severely affected by the addition of new constraints and parameters as compared to Annealing. This behavior is evident from Figure 8, where the SimE(WPC) search process attains much higher




(a) Simulated Annealing (b) Simulated Evolution

Fig. 8. Effect of the additional port constraint on the performance of Annealing and SimE for different problem sizes.

solution costs as compared to its performance without the port constraint.

Table 3
Comparison between the performance of SA and SimE when taking the port constraint (WPC) into consideration.

| | Simulated Annealing | | | | Simulated Evolution | | | |
|---|---|---|---|---|---|---|---|---|
| Cells-Switches | SA | Time(sec) | SA (WPC) | Time(sec) | SimE | Time(sec) | SimE (WPC) | Time(sec) |
| 15-2 | 97.8072 | 0.272 | 102.4707 | 0.284 | 86.8706 | 0.005 | 97.8072 | 0.005 |
| 30-3 | 257.1508 | 0.995 | 260.9837 | 1.025 | 229.1357 | 0.013 | 265.5235 | 0.013 |
| 50-4 | 486.8911 | 2.934 | 548.1965 | 3.066 | 432.3942 | 0.033 | 505.3431 | 0.687 |
| 100-5 | 1157.1383 | 12.909 | 1198.347 | 14.801 | 1055.906 | 0.133 | 1162.0067 | 0.133 |
| 150-6 | 1943.6049 | 33.718 | 2116.6357 | 34.149 | 1701.811 | 0.348 | 1991.3575 | 0.348 |
| 200-7 | 2966.2921 | 67.753 | 3540.3312 | 68.433 | 2453.9762 | 0.719 | 3639.9645 | 0.729 |
| 250-8 | 3852.7231 | 116.987 | 4242.4876 | 118.911 | 3329.5575 | 1.184 | 4240.2983 | 1.193 |
| 300-9 | 5190.4683 | 184.926 | 6042.9186 | 187.413 | 4004.1733 | 1.863 | 6119.7368 | 1.869 |
| 350-10 | 6574.7308 | 272.424 | 7099.5593 | 274.690 | 4624.6573 | 2.740 | 7179.7658 | 2.739 |
| 500-12 | 11550.5805 | 640.933 | 12600.5318 | 643.328 | 7812.2196 | 6.392 | 13203.5318 | 6.401 |

## 6 Conclusion

The design and planning of cellular mobile networks in today's rapidly expanding and highly competitive telecommunications industry is a crucial factor in reducing operations costs while adhering to quality issues and performance constraints. The inherent complexities involved have to be addressed through reasonable and valid simplifications. There is a compelling requirement for software that would aid in this decision-making process, providing cost-optimization algorithms, while adhering to technology and designer constraints. The focus is on a methodology that would make it possible to rapidly assess various design alternatives and carry out studies on the sensitivity of solutions to variations in demand or in the unit cost of equipment or circuits.

In this paper, a solution is proposed for one such area of mobile network design - the cell to switch assignment problem. Simulated Evolution is engineered

to solve this problem and its performance is compared against that of other algorithms. An effective goodness function is formulated that guides the search process towards lower-cost solutions through an iterative application of the *Evaluation*, *Selection* and *Allocation* steps.

One of the main conclusions that can be drawn from the work is that the performance of any non–deterministic iterative heuristic is closely related to its interaction with the problem in general and the elements of the problem in particular. The algorithm parameters that can be closely related to the problem elements are the key entities in deciding the performance of the algorithm and the quality of solution produced. This is evident in the application of SimE, where the goodness function is derived from a detailed knowledge of the problem and optimization objectives. This then provides a certain guided randomness to the search process, thereby reaching higher quality solutions than those attained by SA and TS, which often rely too heavily on non-determinism alone.

## Acknowledgments

The authors thank King Fahd University of Petroleum & Minerals for all support. Thanks are also due to S. Pierre and F. Houeto for providing their data generation code.

## References

[1] F. S. A. Camara, A. B. Santos, G. P. Lasheras, F. F. Cuesta, Planning of the switching and control network in mobile telephony, Telefonica Moviles: Special Issue On Mobile Services.

[2] P. Kriens, Cellular network management, http://www.aqute.biz.

[3] I. T. Union, IMT-2000, http://www.itu.int/home/imt.html, 2000.

[4] S. Pierre, F. Houeto, Assigning cells to switches in cellular mobile networks using taboo search, IEEE Transactions on Systems, Man and Cybernetics, Part B, 32 (3) (2002) 351–356.

[5] S. Menon, R. Gupta, Assigning cells to switches in cellular networks by incorporating a pricing mechanism into simulated annealing, IEEE Transactions on Systems, Man and Cybernetics, Part B 34 (1) (2004) 558–565.

[6] A. Merchant, B. Sengupta, Assignment of cells to switches in pcs networks, IEEE/ACM Transactions on Networking 3 (5) (1995) 521–526.

[7] S. M. Sait, H. Youssef, Iterative Computer Algorithms and their Application to Engineering, IEEE Computer Society Press, December 1999.

[8] A. Quintero, S. Pierre, A memetic algorithm for assigning cells to switches in cellular mobile networks, IEEE Communications Letters 6 (11) (2002) 484–486.

[9] S. Shyu, B. Lin, T. Hsiao, An ant algorithm for cell assignment in PCS networks, IEEE International Conference on Networking, Sensing and Control 2 (2004) 1081–1086.

[10] I. Demirkol, C. Ersoy, M. Caglayan, H. Delic, Location area planning and cell-to-switch assignment in cellular networks, IEEE Transactions on Wireless Communications 3 (3) (2004) 880–890.

[11] Y. Wu, S. Pierre, A new hybrid constraint-based approach for 3G network planning, IEEE Communications Letters 8 (5) (2004) 277–279.

[12] S. Salcedo-Sanz, X. Yao, A hybrid hopfield network-genetic algorithm approach for the terminal assignment problem, IEEE Transactions on Systems, Man and Cybernetics, Part B, 34 (6) (2004) 2343–2353.

[13] R. M. Kling, P. Banerjee, ESP: A new standard cell placement package using simulated evolution , Proceedings of 24th Design Automation Conference (1987) 60–66.

[14] R. M. Kling, P. Banerjee, ESP: Placement by Simulated Evolution, IEEE Transactions on Computer-Aided Design 8 (3) (1989) 245–255.