Observations on Using Genetic-Algorithms for Channel Allocation in Mobile Computing

Albert Y. Zomaya, *Senior Member*, *IEEE*, and Michael Wright

Abstract—This paper highlights the potential of using genetic algorithms to solve cellular resource allocation problems. Generally, cellular resource allocation deals with how and when to allocate radio frequency channels to mobile hosts. The objective in this work is to gauge how well a GA-based channel borrower performs when compared to a greedy borrowing heuristic. This is needed to establish how suited GA-like (stochastic search) algorithms are for the solution of optimization problems in mobile computing environments. This involves the creation of a simple mobile networking resource environment and design of a GA-based channel borrower that works within this environment. A simulation environment is also built to compare the performance of the GA-based channel-borrowing method with the heuristic. To enhance the performance of the GA, extra attention is paid to developing an improved mutation operator. The performance of the new operator is evaluated against the heuristic borrowing scheme. For a real-time implementation, the GA needs to have the properties of a micro GA strategy [4]. This involves making improvements to the crossover operator and evaluation procedure so the GA can converge to a "good" solution rapidly.

Index Terms—Channel allocation, genetic algorithms, mobile computing, wireless networks.

1 Introduction

The widening availability of mobile information systems is being driven by the increasing demand to have information available for users at any time. As the availability of wireless devices increases, so will the load on available radio frequency resources. The radio frequency spectrum is limited, thus there will be a need to effectively manage these resources. This paper evaluates an approach for managing these radio resources such that their availability is maximized and, hence, the disruption to services is minimized [1], [3], [12].

A cellular network is a mobile network in which radio resources are managed in cells. A cell is a two-dimensional area bounded logically by an interference threshold. By exploiting the ability to borrow radio bandwidth, one can ensure that areas of high load are assigned sufficient resources. Further, resources can be assigned in a way that maximizes their availability at a future time. A number of decisions have to be made in the process of resource assignment. Considerations include where to borrow resources from and when to borrow them. Thus, a choice must be made intelligently that minimizes interruption to services.

2 The Channel Allocation Problem

Adding bandwidth to a mobile network is an expensive operation. When developing a mobile network's infrastructure, wasting bandwidth should be avoided because it is very costly. Thus, the objective is to try and get the most out of the minimum infrastructure. The same problem applies to a mobile network that is already installed, where it is cheaper to utilize the available resources more effectively than to add more bandwidth.

The channel allocation problem involves how to allocate borrrowable channels in such a way as to maximize the long term and/or short-term performance of the network [13]. The performance metrics that can be used to evaluate the solutions proposed will be primarily the number of hosts blocked and the number of borrowings. A host is blocked when it enters a cell and cannot be allocated a channel. Obviously, the more hosts that are blocked, the worse will be the performance of the network. The other major metric is the number of channel-borrowings. This should be minimized because channel borrow requests generate network traffic. There are other metrics that can be used to evaluate the performance of the solution such as the number of "hot cells" that appear in a cellular environment.

There are in fact a number of ways to deal with excess load in mobile networks in addition to channel borrowing, such as channel sharing and cell splitting [11]. In particular, cell splitting is commonly used in many real cellular networks [1]. Cell splitting works by breaking down cells into smaller cells. This is accomplished by having several different levels of cell coverage. These different levels are often called macrocells and microcells. A macrocell is essentially an umbrella over a set of microcells. When traffic becomes too great for a cell to handle and there is a microcell structure in place, the cell can be switched out and the microcells switched in. This enables the original cell site to handle more load. There are obvious drawbacks to this scheme that prevent it being implemented throughout the network. The obvious problem is, of course, cost. A secondary concern is the extra network traffic introduced by the additional cells.

- *A.Y. Zomaya is with the School of Information Technologies, The University of Sydney, Madsen Bld. F09, Sydney, NSW 2006, Australia. E-mail: zomaya@it.usyd.edu.au.*
- *M. Wright is with Motorola, 256 St. Georges TCE, Perth, WA 6000 Australia.*

Manuscript received 26 June 2001; accepted 8 Apr. 2002.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 116284.

2.1 Frequency Reuse

Frequency reuse is a technique used in mobile networks to make more effective use of the available radio resources. Put simply, frequency reuse is a way of assigning the same channel to more than one base station. By assigning a channel to more than one cell, we get two major benefits: lower transmitter power and increased capacity. Lower transmission power can result in longer battery life in mobile devices.

Channels that use the same frequencies in a cellular system are called cochannels. The distance between cochannels is determined primarily by the transmitter power used at each cell site. A quantity which is related to this distance, called the Carrier-to-Interference Ratio (CIR). The CIR is used to measure to what extent two cells that use the same frequencies interfere with each other.

Due to cochannel interference, the same radio frequencies are generally not used in neighboring cells. In both of the above networks, all of the available radio spectrum is divided into sevenths and assigned to each cell in a group of seven cells. This scheme demonstrates a classic trade-off. If the network has smaller cells, the channel reuse distance is also smaller. The advantage of using smaller cells is such that they will support more hosts, as the channel density is higher. However, it will cost more to use smaller cells because more of them are required to cover a service area. There is also an advantage to using larger cells because fewer are needed to cover a given area and, hence, the cost of implementing the network is less. However, to support larger cells more transmitter power is required and, hence, the mobile battery's life is lower.

2.2 Mobile Hosts

In a wireless environment, a cell is a bounded two-dimensional area that is created by a Mobile Service Station (MSS). A mobile host (MH) is an entity that exists in a mobile network and is free to move around. The definition of a mobile host is always changing as mobile devices become more advanced. A new paradigm is emerging whereby mobile hosts are containing more local intelligence. Examples of this are Personal Digital Assistants (PDAs) and many new mobile telephony devices. With increasing inbuilt computational power, a mobile network may yet emerge as a dynamic parallel computing platform.

In this paper, mobile hosts are treated as "dumb" entities that simply consume radio resources and follow some path. In a related work [11], mobile agents are used to convey load information around the network. These agents could potentially be executed on mobile hosts. This may reduce the load on the mobile service station in the future.

2.3 Load Coping Schemes

Limited radio bandwidth is a major issue in mobile networking [11]. When a cellular layout plan is designed, each cell is assigned a fixed number of channels able to handle ambient and predicted system loads. Situations can occur, however, where the load at points of the system increases beyond that which the system can handle. Specifically, when all channels in a cell are allocated to hosts, the cell is considered "hot." Hot cells must be avoided because mobile hosts will be denied a service upon entering the cell and hand-off requests will also be denied. There are a number of schemes available which can be used to cope with dynamic load. Some popular schemes are described next.

2.4 Channel-Borrowing With Locking

Channel borrowing with locking or simple borrowing (SB) is a scheme that allows neighbors to use each other's channels. This scheme uses cochannel locking to eliminate cochannel interference that occurs when a neighboring cell borrows a channel. A cell's neighbor can borrow a channel if it is not already allocated to a local host. There are a number of decisions to be made when borrowing a channel, including where to borrow from and when to borrow. A dynamic load-balancing algorithm must make these decisions.

Another more complex scheme exists, known as channel borrowing without locking (CBWL). This scheme uses power control to avoid locking the cochannels of the lending cell and is described in [11]. This technique uses the borrowed channel under less power.

The constrained power problem has also been researched as a method of improving channel utilization [5], [6], [18]. These particular research efforts suggest a dynamic resource allocation algorithm based on constrained power control. The goal is to minimize all transmitter power subject to a finite frequency spectrum and finite transmitter power.

There are some other models for mobile networks that differ from the classical fixed base station configuration. There is a great deal of research occurring in the area of robotics where each mobile host or robot is actually a mobile service station as well. This creates a new set of problems particularly to do with routing algorithms. Since there may be no underlying fixed network structure, message routing tables will be changing constantly. This requires an advanced dynamic mobile routing algorithm that can cope with the changing structure of the virtual network links.

3 Genetic Algorithms

A Genetic Algorithm (GA) is a search algorithm based on the principles of evolution and natural genetics. GAs combine the exploitation of past results with the exploration of new areas of the search space. By using *survival of the fittest* techniques combined with a structured yet randomized information exchange, a GA can mimic some of the innovative flair of human search.

A generation is a collection of artificial creatures (strings). In every new generation, a set of strings is created using information from the previous ones. Occasionally, a new part is tried for good measure. GAs are randomized, but they are not simple random walks. They efficiently exploit historical information to speculate on new search points with expected improvement [7], [8], [9], [13].

The majority of optimization methods move from a single point in the decision space to the next using some transition rule to determine the next point. This point-to-point method is dangerous as it can locate false peaks in multimodal (many-peaked) search spaces. By contrast, GAs work from a database of points simultaneously (a population of strings),

climbing many peaks in parallel. The probability of finding a false peak is reduced compared to methods that go point to point. The mechanics of a simple GA are surprisingly simple, involving nothing more complex than copying strings and swapping partial strings. Simplicity of operation and power of effect are two main attractions of the GA approach. The effectiveness of the GA depends upon an appropriate mix of *exploration* and *exploitation*. Three operators to achieve this are: *selection*, *crossover*, and *mutation*.

Selection according to fitness is the source of exploitation. The mutation and crossover operators are the sources of exploration. In order to explore, they must disrupt some of the strings on which they operate. The trade-off of exploration and exploitation is clearest with mutation. As the mutation rate is increased, mutation becomes more disruptive until the exploitative effects of selection are completely overwhelmed. More information is provided on these operators in [7], [13].

3.1 PGAPack

PGAPack is a general-purpose, data-structure-neutral, parallel genetic algorithm library written in C. PGAPack was chosen for implementing the genetic algorithm in this thesis. The library had many benefits, making it an attractive choice [10]. PGAPack allows multiple levels of access, which means that a user can provide their own mutation and crossover functions or use the defaults provided. This allows for a much smoother development process.

The other major benefit PGAPack provides is a Message Passing Interface (MPI) back-end [10]. MPI is a library written in C that allows processes to pass data between each other. The library is designed so that you only write one program, which has message passing constructs and is built in a client/server or peerless configuration. For example, the MPI library can then start a process on a pool of processors in a network-of-workstations. The result is that PGAPack can be parallelized with a simple option. Of course, some consideration still must be given to the parallel nature of the program, though this is reduced significantly [2], [8].

4 GA-Based Bandwidth Allocation

This section describes the design of the GA-based solution. This includes a description of the encoding schemes, chromosome seeding, fitness calculations, and crossover and mutation operators. The simulation environment used for the Genetic Algorithm method is based on the fixed channel allocation simulation.

The purpose of the Genetic Algorithm is to find channel-borrowing policies that give good network performance. The GA essentially takes the place of the channel borrowing heuristic described previously. The GA simulations are designed so that the GA is executed every iteration of the simulation (Fig. 1). Although this is quite computationally expensive, it is a sound starting point for testing the efficiency of the GA. The initial goal is to make the GA perform well for this execution frequency.

The other objective is to reduce the execution frequency and execution time, while still generating "good" channel allocation policies. Ideally, for practical implementation purposes, the goal would be to create a Micro GA [4], which has very small population and execution requirements.

4.1 Encoding Schemes

One of the most important aspects that control a GA's performance is the encoding method chosen. The encoding refers to the method by which the problem parameters are mapped into a chromosome. A particular encoding can be weak or strong. For example, a strong encoding exploits features of the solution space in the mapping. For example, the information presented to the GA could be encoded in two ways. One way could be to encode the real line as a binary representation in the chromosome. A second way could use the gray-code representation of real number line values [10]. The gray-code method exploits the continuous nature of the real line because solutions are close together from a bit inverting point of view.

A number of encoding schemes were chosen for the channel-borrowing problem, but one particular encoding scheme emerged as the most suitable from those sampled. The encoding method is a global approach to the problem. It is global because any chromosome has enough information to describe a set of channel-borrowings for the entire network.

A chromosome is composed in the following way. For every cell in the network there is a major chromosome slot or super-gene. Within each supergene there are six actual genes. These genes represent the six neighbors of a cell. In total, there are $6N$ genes for a network of size N. Since all of the simulations in this project are run on 100 cell networks, each chromosome consists of 100 supergenes and 600 actual genes in total.

The size of the solution space can be calculated by the formula $S = C^{7N}$. The size depends on the number of channels C available to each cell. This is because a borrow operation could borrow as many channels as there are available. Obviously, a lot of these borrowing operations would not be allowed as a consequence of other borrowing and cochannel locking. This, however, represents an upper bound for the search space size because the GA does not look for these inabilities to borrow. The GA simply tries to borrow a channel regardless. The search space can be computed for all the simulations in this work via $S = C^{7N}$. Since the network size is always 100 cells and each cell has 10 channels available the solution space size is 10^{700}. This is clearly an unworkable size for both brute force and random searches.

As described previously, the problem encoding is global approach. This is not a particularly powerful encoding scheme. The parameter space for the problem is decomposed into super-groups of seven cells. A consequence of this is that channel-borrowing operations directly affect a neighbors' channel-borrowing capacity. A problem arises from the fact that this encoding scheme linearly maps a two-dimensional network into a one-dimensional chromosome. While this mapping does not lend itself well to a simple crossover operator, there are properties of this encoding that are suited to it.

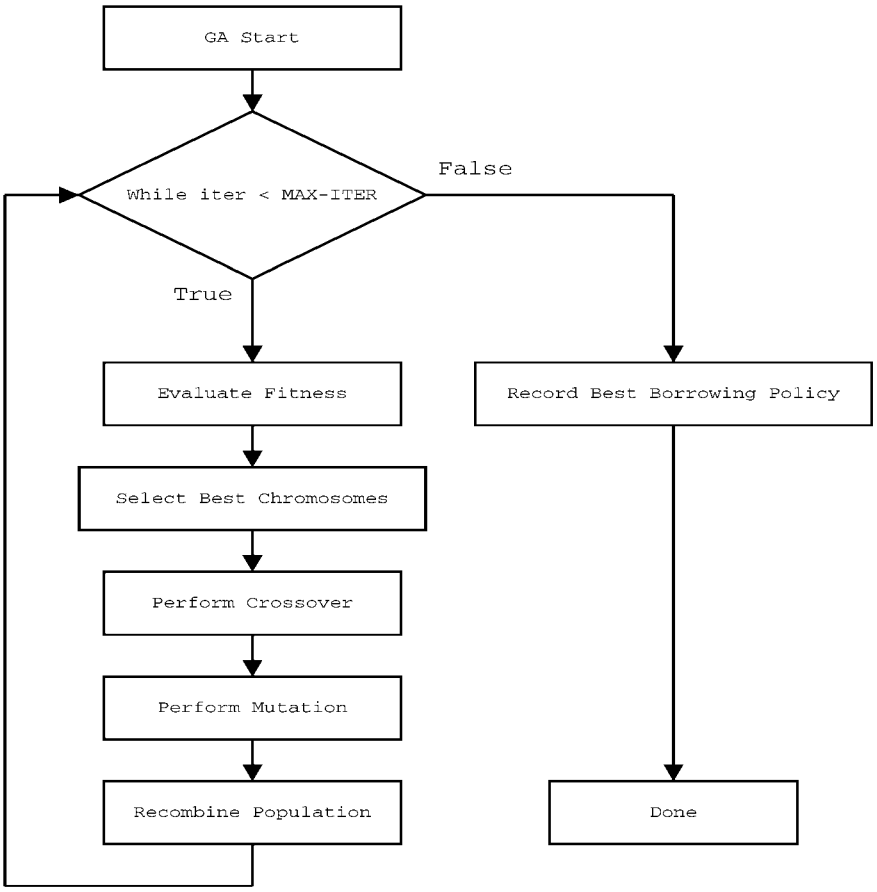

Fig. 1. GA flow.

There are cases where the crossover operator will accelerate the GA to a better solution. Suppose there are two highly loaded regions in a 100-cell network. If these regions are located towards the top and bottom of the network, the crossover operator will tend to recombine chromosomes that have good channel-borrowing strategies at the beginnings and ends of their chromosomes. Borrowing decisions that are distributed distantly and vertically are mapped to genes that are distant in a chromosome. Borrowing decisions that are distributed distantly and horizontally are mapped to much closer points in the chromosome. This lack of spatial generality is the weakness of the encoding.

A more powerful encoding could possibly be based on local cell information or a global/local hybrid scheme. The goal is to find an encoding which clusters solutions together and also minimizes the impact of borrowing decisions on other cells.

For a GA to be efficient, or at least useful, it must converge to a solution within the number of iterations given by $r^n/m - 1$ [4]. This equation is basically the size of the search space divided by the number of strings in the population m minus 1. The quantity r^n is simply the number of values a gene can take raised to the power of the number of genes in a chromosome. If the number of iterations needed to find a solution exceeds this quantity, a brute force search would be a better alternative because it would cover the solution domain once completely.

In related work [19], resource allocation is treated as a call admission problem. In this work, GAs are used to find "good" call admission policies. The chromosome is encoded using three genes in a group to describe a local call admission policy. The encoding is binary. The bits of the chromosome represent admit or reject decisions for a new call arrival, a hand-off request from a cell on the left, and a hand-off request from a cell on the right. The interesting result is that, when a two-dimensional network is packed into a linear chromosome, the GA performs better than the best heuristic hand-off policies available.

4.2 Fitness Calculation

The fitness calculation is used to rank the quality of a chromosome. It is the job of the evaluation function to assign each chromosome a fitness value. The fitness value is used to determine how large a sector of the selection roulette wheel a chromosome should be allocated. The roulette wheel is only an analogy for what actually occurs, although, from a statistical point of view, the two are equivalent.

PGAPack scales the fitness value to a double precision floating-point quantity ranging from 0.0-1.0. PGAPack allows any fitness values to be presented to it, as long as the algorithm for evaluation is consistent across evaluations. The actual evaluation procedure is outlined in "pseudocode" below:

```
double evaluate( chromosome c ) {
  make a copy of MSSCollection
  make a copy of MHCollection

  for each cell i {
    for each neighbor j {
       value = getAllele( i*6+j )
       for each value b
         borrow a channel from (j -> i)
       next b
    next j

    assign all unassigned hosts in cell i
  next i
  blocked = number of hosts that are blocked
  fitness = blocked
  return fitness
}
```

The above evaluation function sets the fitness value to the number of hosts that would be blocked if the borrowing scheme represented by the chromosome were applied to the network. A number of other fitness functions were tested and it was found that a suitable fitness function could have the form given below.

$$fitness = \beta * borrows + \gamma * hot\_cells + \alpha_0 * blocked\_now + \alpha_1 * blocked\_later. \quad (1)$$

The most important aspect of (1) is the term $\alpha_1 * blocked\_later$, which represents the number of hosts blocked at some future time. This could be built into the fitness function to allow the GA to find chromosomes that perform well under future load as well as current load.

4.3 Seeding

Seeding is the process of setting the initial population to some initial configuration. If a population is seeded properly, the performance of a GA can be vastly improved. Since a GA works by probabilistically mutating and combining parameter encodings, the number of iterations can be significantly reduced if the population is initially preset to a good solution.

Two different methods of seeding the population were implemented. The first was to simply set every gene to zero. This initializes each run with no borrowing decisions. The GA is then used to find a "good" borrowing policy from "scratch." The second method was to seed the population with the output from the channel-borrowing heuristic. The channel-borrowing heuristic simply iterates through every cell in the network and calculates how many channels would be needed to support the number of hosts arriving at the time of GA invocation.

4.4 Crossover and Mutation

PGAPack provides a default crossover operator, which is used in all the simulation studies in this work. As explained previously, the crossover operator should find good solutions for groups of hot spots that are separated by at least one cochannel interference distance from each other. For other scenarios, the default crossover operator will have weaker performance. This can be overcome if the mutation operator works well and the population size is sufficient to allow enough diversity in the genetic population.

The mutation operator is responsible for introducing new genetic material into a population. PGAPack provides a default mutation operator that is based on a coin with a head probability equal to the probability of the mutation rate. A coin is thrown for each gene. If the result is a head, the gene will be mutated. The gene is mutated to a value chosen from a uniform random variable scaled to the minimum and maximum gene range. The performance of the default mutation operator is the subject of (5).

The mutation operator has several weaknesses as a result of its generality. The operator's main weakness is that it can make borrowing decisions ahead of time that are nonoptimal. These decisions can be nonoptimal for two reasons. First, their effectiveness is not measured in the fitness function. This tends to make the genes random and mostly useless. Second, these decisions degrade the future quality of service. The degradation comes about because, for every borrowed channel, seven more are locked out from use. The objective should be to keep borrowing to a minimum.

There is a problem with trying to minimize borrowing when using the default mutation operator. Since the operator is applied probabilistically over the entire chromosome length, it is difficult to minimize borrowing away from load while at the same time attempting to do optimal borrowing in and around loaded areas. To alleviate these weaknesses, a replacement mutation operator was developed.

The new mutation operator only mutates the borrowing policy in cells that need channels. Two variations of this mutation idea were implemented. The first simply mutates genes that could possibly reduce blocking. The second variation takes into account how many borrowings are actually needed to satisfy the requirement so as to prevent over allocation. The extra condition in the second mutation operator is actually implemented in the fitness function. The excess channel borrow requests are summed over the length of the chromosome. This total is then used to de-rate the fitness of the chromosome linearly.

The new mutation operator has a number of other advantages. The solution space size is no longer fixed to the maximum given by equation $r^n/m - 1$. Instead, it lies somewhere between zero and the maximum, depending on how many cells have unassigned hosts. Another advantage for this solution is the emergence of clustering-like behavior. The new mutation operator will often create situations where the GA needs only to optimize over groups of six genes. This is an advantage because the solutions that the permutations of the six gene groups give are closer together in terms of fitness.

A small change in the mutation probability or rate interpretation is required to operate the GA more efficiently. Because the fitness function takes into account the number of excess channel borrows, the mutation rate tends to be desensitized. This is also due to the nature of the problem and fact that the new mutation operator limits the borrowing decisions. In other words, the mutation rate can be varied significantly while still maintaining a good

convergence rate. Simulation 8 examines this behavior in more detail.

5 Simulation of Channel Allocation Schemes

This section will investigate the performance of a fixed channel allocation scheme and a simple channel-borrowing scheme. The two will be compared and some inferences will be drawn. This is particularly useful to examine because it is desirable that the GA-based solution approaches the performance of the heuristic borrowing scheme and possibly surpass it in terms of long-range performance.

5.1 Simulation Environment

All of the simulations to follow are based on examining transient loads imposed on ambient load configurations. A simple way of generating ambient loads is required. A number of simulations are performed using different loads. Some of these loads are test cases, some are random, and some are characteristic of physical configurations. In the case of the random loads and the loads that imitate some realistic phenomenon, a load map is generated.

5.1.1 Mobile Network Environment

The first objective for this paper was to develop an environment for conducting the channel allocation simulations. Two classes were developed to facilitate the simulations. The mobile service station and mobile host classes are the main two classes that contain the state information for the base stations and the hosts in the network.

5.1.2 Mobile Service Stations or Cells

The mobile service station or cell is a fixed entity in the network. A simple model of a mobile service station was developed to suit the needs of the simulations in this project. A mobile service station is a model of a fixed transmitter/receiver that has a number of channels assigned to it. It also has the ability to inherit more channels if demand requires. The radius of the station defines a cell region. To model a cell, a number of elements are required (state information): unique cell identification number, the X and Y position of the station in the 2D plane, transmission radius, a list of neighboring cells, a list of local and borrowed channels, and a list of cochannels.

As described previously, the cells are arranged in a hexagonal pattern. This means every cell has exactly six neighbors. A neighbor is defined as a cell that is adjacent, or more formally, a cell that is one cell diameter away. Every simulation conducted in this paper uses a seven-cell frequency reuse pattern. This implies that there are six cochannel cells for every cell.

5.1.3 Mobile Hosts

A mobile host is modeled in this project as an entity that has an ability to move in a straight line with a certain velocity. Each host can consume one channel from the cell that it is currently located in. There is a possibility for a host to be in one of three different cells if located at the corner point of the three cells. This is not an unrealistic assumption, as real mobile devices can switch cells when the signal strength becomes low or the same between cells. This can happen in GSM and CDMA mobile networks [15], [17].

A handoff will occur when a mobile host experiences a stronger signal from a cell not including the one it is in. For simplicity, the simulation environment treats signal strength and distance from a base station as two linearly related but opposing quantities. This implies that a handoff will occur when a host becomes closer to a base station other than the one it is in.

To support the notion of a handoff, each mobile host must store the identifier on the cell it is in and the cell it is currently assigned to. In summary, the state information stored by each mobile is: unique mobile host identification number, the X and Y position of the host in the 2D plane, the X and Y components of the hosts velocity, the cell the host is located in, the cell the host is assigned to, and the channel the host is assigned to.

5.1.4 Load Maps

A load map is simply a matrix of values that specifies how many hosts are in each cell of the network. The load map can be thought of as a gray-scale image map. The only complication arises from the fact that the mobile network simulations use hexagonal shaped cells and the load map is square.

This problem can be overcome by a simple translation. A hexagonal cellular network can be represented as a structure composed of squares. There are two ways hexagonal networks can be aligned—horizontally or vertically. In the simulations to follow, all of the cells are aligned horizontally. To transform this into a square map, the rows are numbered starting from zero at the bottom. Every even row can then be translated half a cell radius to the left. Once the network is square, values can be taken from the corresponding load map vertices and assigned as a traffic load.

A load map can be constructed in a number of ways. Manual construction, samples from real data, and random construction are three such ways. The map itself can be most easily visualized as a gray-scale image. For simplicity, the map elements are normalized to a range from 0.0 to 1.0. A white area indicates a load of 1.0 and a black area indicates a load of 0.0. In this paper, the value 1.0 in a load map indicates a hot cell. That is, the number 1.0 multiplied by the maximum number of hosts in a cell is equal to the number of channels in a cell. Additionally, the value 0.0 represents a cell with no load.

5.1.5 Simulation Information

The simulations all have a finite runtime and a variable time-step. The time-step is variable so that it can be made small enough to ensure that no handoffs are missed. A handoff occurs when a mobile host changes to a new cell. If the time-step is too small a mobile host may "skip" through a cell before a data point can be recorded. Both the velocities of the hosts and the sampling rate must be taken into account when constructing a simulation.

There is a case when a mobile host passes through a cell extremely close to one of the six corners. In fact, as a host's path tends towards the corner of a cell, the time-step must be made progressively smaller. A trade-off arises between

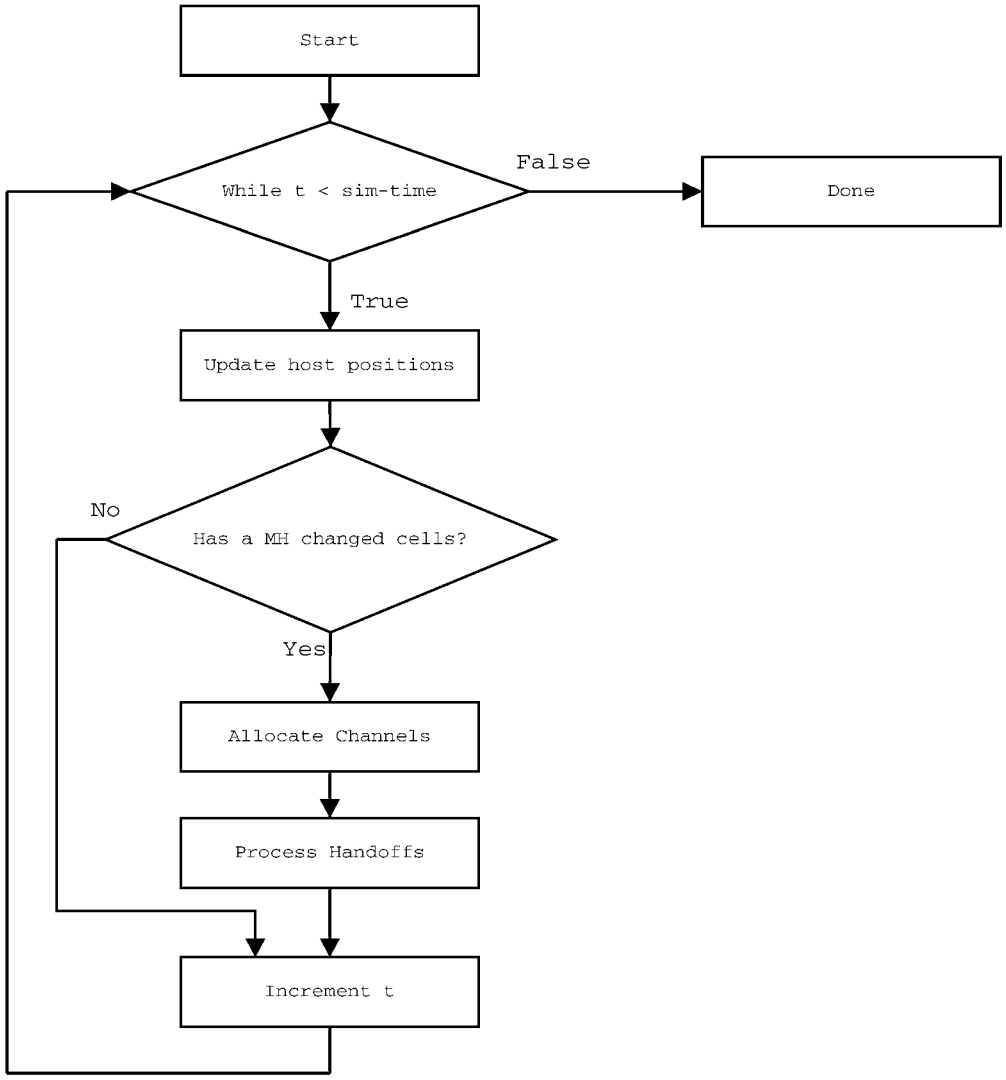

Fig. 2. Simulation flow diagram.

the number of sample points generated and the shortest path through a cell that can be sampled. Generally, if a host is passing through a cell very close to a corner, it does not matter which cell it is assigned to because the signal strength is nearly the same from the three cells that make up the corner. This limitation should be taken into consideration when designing simulations.

5.1.6 Simulation Flow Diagram

The general flow of simulation is given in Fig. 2. The actual performance is measured in a number of ways. The main performance metric used is the number of hosts blocked verses time.

For this simulation, a host is considered blocked when upon entering a new cell it is denied a service because there are no free channels available. A new data point will be calculated each time there is a change in network state. This will occur whenever there is a handoff. The simulations involve a number of parameters such as the number of mobile hosts and mobility. Mobility is a simple parameter that describes the ratio of the hosts that are stationary. This allows an ambient load to be set up in the network and a transient load to be examined.

For both simulations, the load map in Fig. 3 will be used for the ambient load. The simulation is set up to test the number of mobile hosts that are blocked from service each time there is a change in network configuration. As explained previously, a network configuration change occurs when there is a handoff. The simulator attempts a handoff whenever a mobile host enters a new cell. At this point, the simulator attempts to reassign all hosts to a

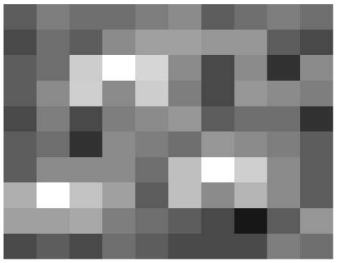

Fig. 3. Simulation load map.

TABLE 1
Simulation Parameters

Parameters	
Network size	10×10 (100 cells)
Cell radius	5 units
Number of channels per cell	10

channel in the cell they are in when there is a change in the network configuration.

The simulation uses the same ambient load over three different mobility loadings: 0.13, 0.23, and 0.37. These mobility factors are a consequence of choosing 64, 128, and 256 actual mobile hosts. Since each cell has 10 channels and the maximum number of stationary hosts per cell is set to 10, the result is 433 stationary hosts. This is derived by multiplying each point in the load map by the maximum number of mobile hosts in a cell.

There are a number of parameters that are constant over the following simulations (Table 1). It should be noted that the simulation environment is wrapped. This means that once a mobile host leaves the edge of the network it reappears on the opposite edge. The cell neighbor information, however, does not wrap over to the opposite side. Doing this would invalidate the model's "real-world" resemblance to some degree, since there is no global "real-world" mobile network that is controlled by one entity.

5.2 Fixed Channel Allocation Scheme

The fixed channel allocation scheme is the simplest scheme to implement and maintain. This is because it does not fully utilize the available radio resources [11]. Under this scheme, the network is set up to take advantage of frequency reuse. As described previously, the fixed channel scheme does not allow any channel borrowing of any kind. The simulation environment will be set up to show a number of different performance metrics.

TABLE 2
Parameters of Simulation 1

Parameters	
Load map	LoadMap 1
Number of mobile hosts	64, 128, 256
Max number of stationary hosts in a cell	10
Simulation time	100 seconds
Simulation resolution	0.05
Number of stationary hosts	433
Mobility	0.13, 0.23, 0.37
Total time steps	2000

5.2.1 Simulation 1

Simulation 1 is designed to show how the fixed channel allocation scheme performs under different loads or mobility factors. The simulation is set up to show what happens to the network when a large group of mobile hosts move across the network in one direction. The direction, velocity and initial position chosen for the simulation is arbitrary but in this instance it is 030°, $\sqrt{3}$ units/second, and cell (4,3), respectively.

The simulation will be subject to the ambient load given by the load map in Table 1. The parameters for the simulation are given in Table 2 and the corresponding simulation results in Fig. 4.

The results for simulation 1 are reasonably obvious. The variations in the number of mobile hosts blocked are directly proportional to the underlying ambient load determined by the load map. Table 3 presents an average of the three series in Fig. 4.

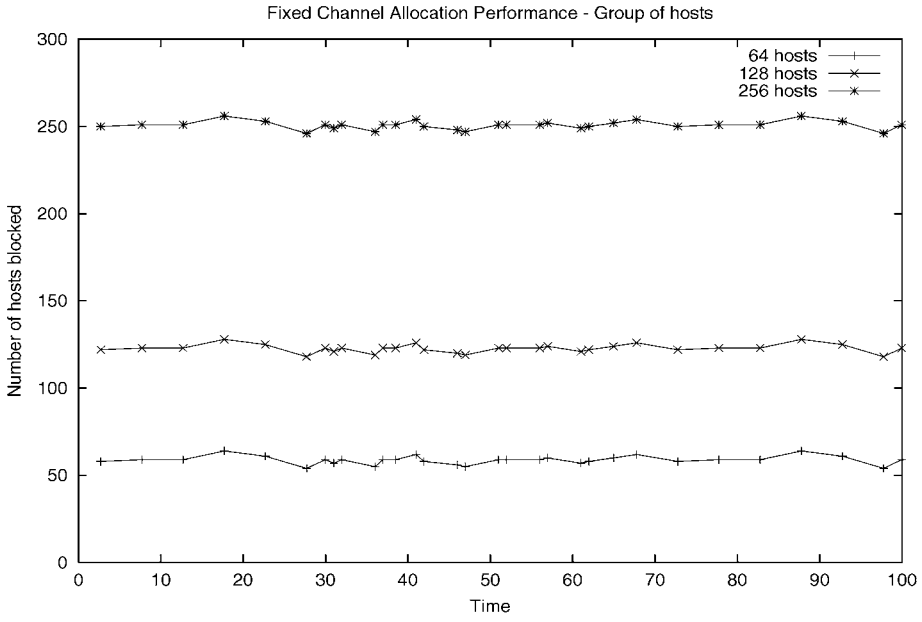

Fig. 4. Simulation 1.

TABLE 3
Simulation 1 Average Blocking

Number of hosts	Average blocked
64	58.8
128	122.8
256	250.8

TABLE 4
Simulation 2 Average Blocking

Number of hosts	Average blocked
64	5.86
128	20.1
256	69.4

5.2.2 Simulation 2

Simulation 2 is also designed to show how the fixed channel allocation scheme performs under different loads or mobility factors. This simulation is set up so that the mobile hosts all start in the same cell and then move away in random directions with random velocities. The number of hosts blocked should be large initially but then drop off afterwards to some average. The simulation will be subject to the ambient load given by the load map in Table 1. The parameters for the simulation are given in Table 2. Three different results are recorded in Fig. 5.

The results clearly show that there is an initial sloped region followed by some average number of mobile hosts blocked from service. This is consistent with what was suggested in the description above. This behavior is due to the initial concentrated load spreading out randomly. The sloping region can be omitted and an average taken which can be used as metric for this scheme (Table 4).

5.3 Channel-Borrowing Allocation Scheme

The channel-borrowing scheme examined is a simple scheme as described in [11]. To make this scheme near optimal but nonrealistic, it was decided that every hand-off request should trigger a channel borrow operation if it is possible. This makes the scheme a reasonable lower bound for the GA method. This decision is slightly unrealistic in a practical sense since the goal is to reduce blocking and this scheme would still produce service interruptions. The goal is to have bandwidth available in anticipation for future network load. This is what is required in a durable solution.

The channel-borrowing scheme is simulated in much the same way as the fixed channel scheme above. The number of hosts blocked will be the major metric used. In addition to this, we will look at the number of channels borrowed. The results of this simulation should correlate with the network load experienced.

5.3.1 Simulation 3

Simulation 3 is designed to show how the channel-borrowing scheme performs under different loads or mobility factors. The network is set up to have a group of hosts moving together across an ambient load given by the load map in Table 1. The parameters for the simulations are given in Table 2.

The simulation is configured to show what happens to the network when a large group of mobile hosts move across the network in one direction. The direction, velocity, and initial position chosen for the simulation is arbitrary but in this instance it is 030°, $\sqrt{3}$ units/second, and cell (4,3), respectively. Fig. 6 shows how the performance of the channel-borrowing scheme scales as the number of mobile hosts is increased. Table 5 shows the average blocking for each case.

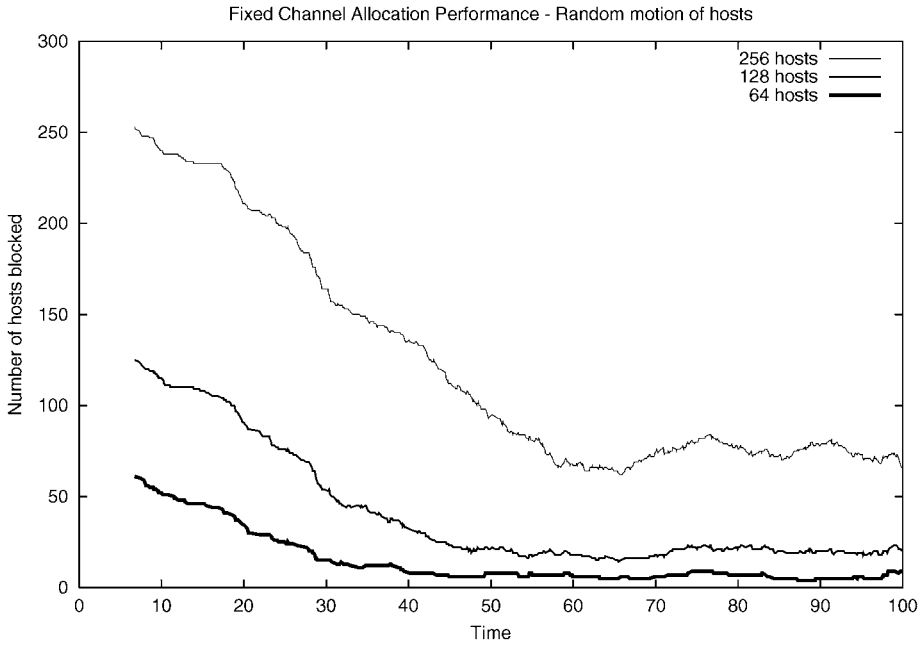

Fig. 5. Simulation 2.

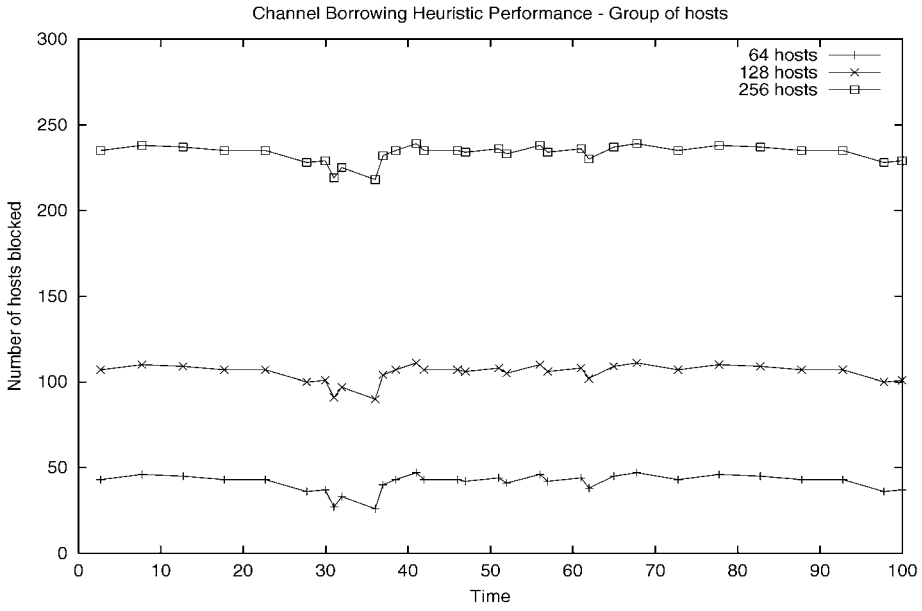

Fig. 6. Simulation 3.

5.3.2 Simulation 4

Simulation 4 is also designed to show how the channel-borrowing scheme performs under different loads or mobility factors. This case differs in the way the mobile hosts move. In the previous simulation, all the mobile hosts move in one large group. In this simulation, all of hosts start in the same cell but then move away in random directions. This would suggest there a higher number of hosts blocked early in the simulation followed by some near constant average.

The simulation is set up to have an ambient load given by the load map in Table 1. The parameters for the simulation are given in Table 6 and the results of the simulations in Fig. 7.

As mentioned previously, all of the mobile hosts start initially in the same cell and move randomly outwards. This would suggest a large amount of blocking initially which falls off to some average. If the sloped region of the graph is omitted and the remainder is preserved (about 60 seconds onwards) an average can be taken. This was done for all three cases and is presented in Table 7. In addition, the results for 512 and 1,024 hosts are also included in the table for completeness.

The Ratio column in Table 7 is approaching 1.0 as the number of mobile hosts is increased. This is a measure of how saturated the network is. In other words, as the number of mobile hosts becomes significant the number of hosts actually admitted to the network turns out to be insignificant.

TABLE 5
Simulation 3 Average Blocking

Number of hosts	Average blocked
64	41.2
128	105.2
256	233.3

5.4 Comparisons

Four simulations were reported in the preceding sections. The first two demonstrated two important load characteristics in a fixed channel allocation environment. The second two demonstrated the same two characteristics in a channel-borrowing environment. These simulations are important because real-life traffic can be analyzed by superposition of these two characteristics. For example, freeway traffic can be modeled in large groups and city business districts can be modeled as random motions.

In particular, the results demonstrate a particular weakness of the channel-borrowing scheme. The weakness that emerges is the potential for the channel-borrowing schemes' performance to degenerate to that of the fixed channel scheme in terms of the number of hosts block from service. To illustrate this, the results of simulations 1 and 3 should be compared and the results of simulations 2 and 4 should be compared (Tables 8 and 9).

Table 8 indicates that the channel-borrowing scheme is performing particularly well against the fixed scheme. In this case, both of these simulations attempt to record the number of hosts blocked each time there is a network

TABLE 6
Parameters of Simulation 4

Parameters	
Load map	LoadMap 1
Number of mobile hosts	64, 128, 256
Max number of stationary hosts in a cell	10
Simulation time	200 seconds
Simulation resolution	0.05
Number of stationary hosts	433
Mobility	0.13, 0.23, 0.37
Total time steps	4000

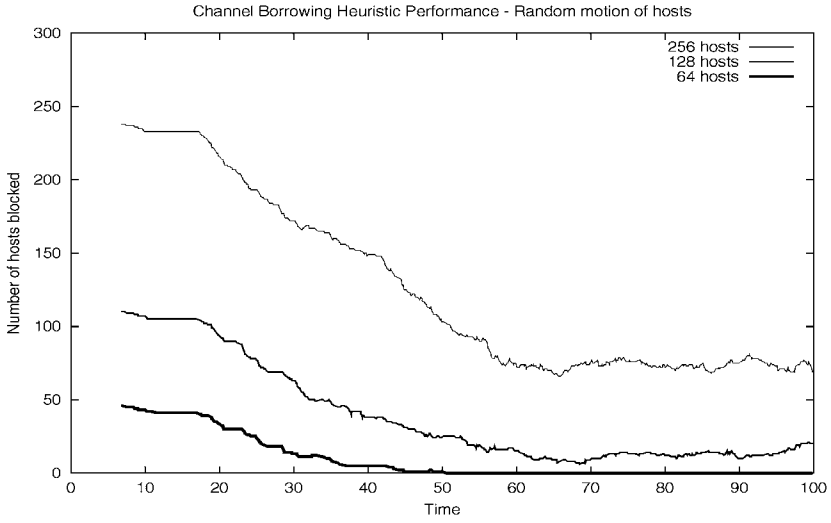

Fig. 7. Simulation 3.

configuration change. The simulation itself consists of a group of mobile hosts moving across a stationary landscape of hosts defined by the loadmap. This indicates the channel-borrowing scheme does well for this type of load characteristic.

The situation is quite different in Table 9. This table shows that the channel-borrowing scheme has approximately the same or worse performance than the fixed channel scheme. This highlights the weakness of the channel-borrowing scheme in an essentially random environment. The performance, in terms of the number of mobile hosts that receive no service, degenerates to that of the fixed channel scheme.

6 Simulation of GA-Based Channel Allocation

This section demonstrates a number of different Genetic Algorithm simulations. Most of the simulation results compare the heuristic and GA channel allocation schemes. Each simulation tests various refinements to the GA scheme. The simulation parameters are given in Table 10.

TABLE 7
Simulation 4 Average Blocking

Number of hosts	Average blocked	Ratio
64	0.1	640.0
128	15.1	8.48
256	71.6	3.57
512	256.9	1.99
1024	765.0	1.33

TABLE 8
Simulation 1 and 3 Average Blocking (Group Motion)

Number of hosts	Simul. 1 Avg. blocked	Simul. 3 Avg. blocked
64	58.8	41.2
128	122.8	105.2
256	250.8	233.2

TABLE 9
Simulation 2 and 4 Average Blocking (Random Motion)

Number of hosts	Simul. 1 Avg. blocked	Simul. 3 Avg. blocked
64	0.1	5.86
128	15.1	20.1
256	71.6	69.4

TABLE 10
Parameters of GA-Based Simulation

Parameters	
Network size	10×10 (100 cells)
Cell radius	5 units
Number of channels per cell	10
Load map	LoadMap 1
Maximum number of stationary hosts in a cell	10
Simulation time	100 seconds
Simulation resolution	0.05
Total time steps	2000

TABLE 11
Parameters of Simulation 5

Parameters	
Number of mobile hosts	128
Population size	100
GA iterations	100
Crossover probability	0.85
Modified mutation rate	0.5
Mobile host configuration	Group motion
Mobility	0.23

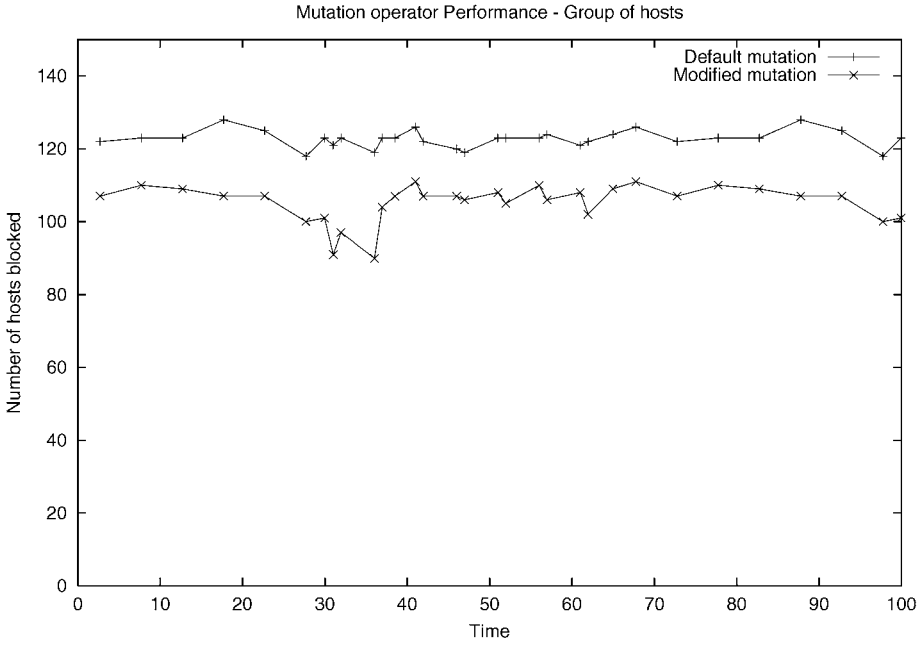

Fig. 8. Simulation 5.

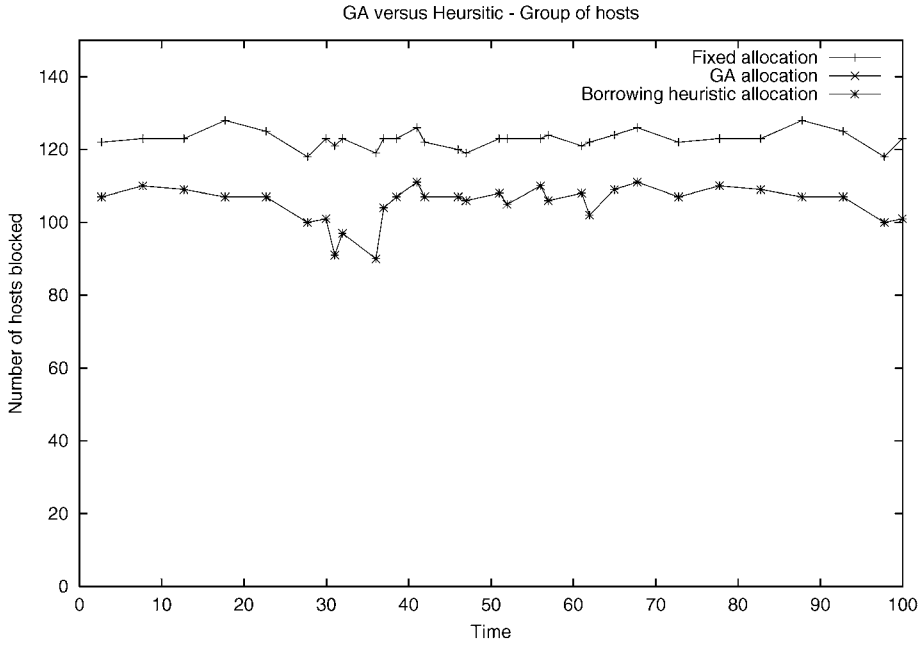

Fig. 9. Simulation 6.

6.1 Simulation 5

Simulation 5 is designed to illustrate the differences in performance of the default PGAPack mutation operator versus the improved operator. The simulation environment is set up so that a group of hosts start in cell 24 and move together across the network. The direction of travel is set to 030°.

The simulation is configured to run the GA every time there is a change in network configuration. It is also set up to run the GA with 100 iterations. The parameters for simulation 5 are summarized in Table 11.

Fig. 8 displays a simulation series for the default mutation operator and the modified mutation operator. The simulation demonstrates the performance of the two mutation schemes for a group motion characteristic.

In this simulation, it can be seen that the default PGAPack provided mutation operator yields performance that was similar to the fixed channel-borrowing scheme. The modified mutation operator performs significantly better.

6.2 Simulation 6

Simulation 6 is designed to show the performance of the GA with the modified mutation operator compared to the channel-borrowing heuristic and the fixed channel allocation scheme. This simulation draws on results from simulations 1 and 3. Three data series are generated. In

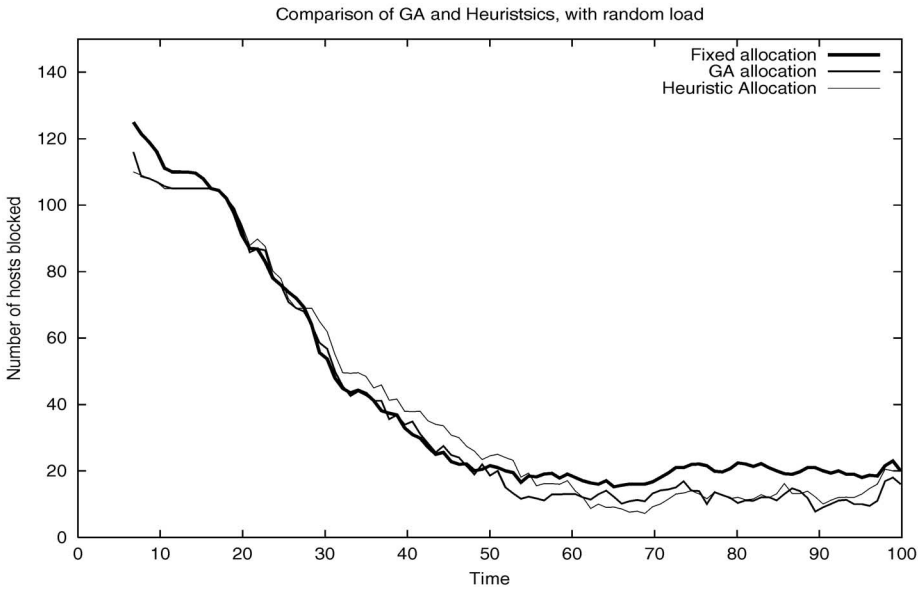

Fig. 10. Simulation 7.

each case, the number of mobile hosts is set to 128. The load is set up to have a group characteristic as before. The parameters for simulation 6 are similar to those in Table 11.

Fig. 9 shows that the GA is performing as well as the channel-borrowing heuristic. There is actually one data series towards the top and two below that. In fact, all of the points of the GA and borrowing heuristic data series are coincident. This is a good result because it implies that any further improvements to the GA are likely to be increasing the long-term performance of the channel allocations.

The simulation demonstrates that the GA reaches the performance of the channel-borrowing heuristic with 100 iterations per invocation. The actual speed of convergence for a number of different GA parameters will be examined in simulation 8.

6.3 Simulation 7

Simulation 7 is designed to show the performance of the GA with modified mutation operator compared to the channel-borrowing heuristic and the fixed channel allocation scheme. The load in this instance is set to start in cell 24 and diffuse randomly.

This simulation draws on results from simulations 2 and 4. Three data series are generated. In each case, the number of mobile hosts is set to 128. The load is set up to become random as described previously. The parameters for simulation 6 are similar to those in Table 11.

Fig. 10 shows that the GA is performing as well as the channel-borrowing heuristic. The simulation demonstrates that the GA reaches the performance of the channel-borrowing heuristic with 100 iterations per invocation. The actual speed of convergence for a number of different GA parameters will be examined next.

TABLE 12
Parameters of Simulation 8

Parameters	
Number of mobile hosts	30
Population size	50, 100
GA iterations	200
Crossover probability	0.85
Modified mutation rate	0.01, 0.1, 0.5, 0.75, 1.0
Mobile host configuration	3 groups of motion

6.4 Simulation 8

Simulation 8 is designed to show how rapidly the GA converges for different genetic parameters. The simulation is conducted for a range of different mutation rates over two different population sizes. Generally, if the population is larger, then the result will be better.

The actual scenario under simulation is set up with a load-map for the ambient network load. In addition to this, three cells are loaded with 10 extra hosts each. The hosts all move at once at an angle of 030° with the same velocity. The results are taken from the first GA run that is invocated to find a good channel-borrowing policy for these hosts. The hosts are placed in cells 21, 24, and 74. The parameters for simulation 8 are summarized in Table 12.

Figs. 11a and 11b reveal some of the characteristic behavior of the GA. As expected, the GA converges faster and, in lesser iterations, when the population is larger. In fact, when the population size is 100, the GA reaches a fitness level of 17 that, in this case, indicates the number of mobile hosts blocked of service. For a population size of 50, the same GA reaches a fitness level of 29.

Figs. 11a and 11b show that a modified mutation rate of 0.1 yields the best convergence for this problem. An interesting result in both figures is that, if the modified mutation rate is 0.5 or larger, the GA rapidly converges in about 40 iterations. The final fitness value it converges on is not as low as the result of GA runs with lower mutation rates. This is most likely due to the reduced resolution of the

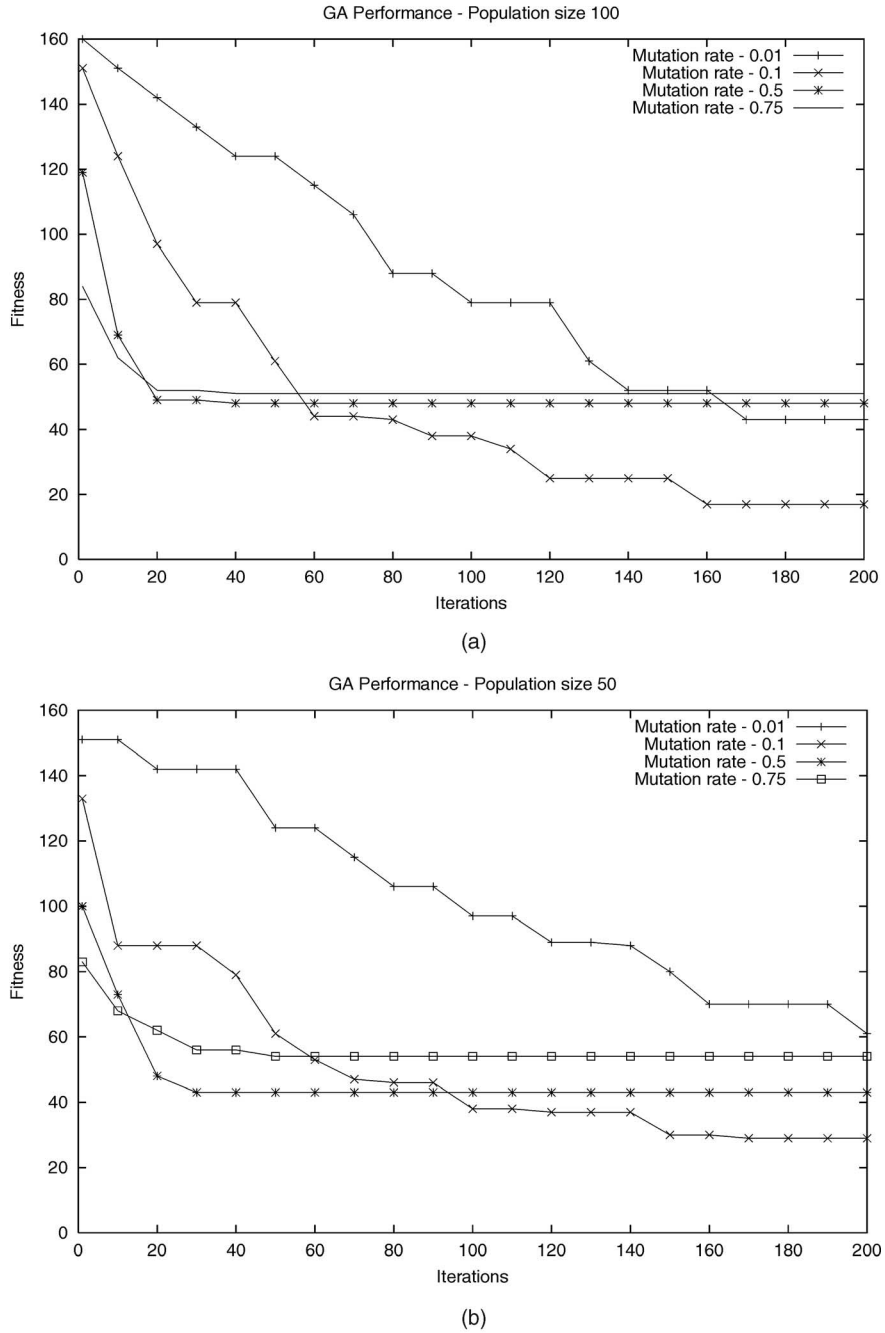

Fig. 11. Simulation 8.

operator at this mutation rate. It is clear that the operator is tolerant below 0.5, however.

7 Conclusions

This work aimed at exploring the possibility that GAs can be used to solve the problem of cellular resource allocation. The research that followed indicated that the problem was well explored but the application of GAs to the problem was not [11], [19]. This paper presented a different approach of handling the resource allocation problem. The method was based on selective channel borrowing. The study treats mobile hosts as trivial entities that consume radio resources. The hosts roam in a mobile network which is simplified somewhat. The structure is cellular and regular. When a host enters a new cell, a hand-off request is made. If there are channels available, the host is assigned immediately to a free channel. If no channels are available, a channel-borrowing operation is attempted. A decision arises concerning which neighboring cell to borrow a channel from.

The channel allocation problem is about how to allocate borrrowable channels in such a way that maximizes the long-term and/or short-term performance of the network. The performance was measured by the number of mobile hosts that are blocked of a service when there is a network configuration change. The study attempted to apply GAs to help make channel-borrowing decisions that minimize the

number of denial-of-service conditions and maximizes the long-term performance of the network. To evaluate the performance of the GA borrowing method, a heuristic was developed that simply borrows a channel from wherever it can. The objective here was to create a GA channel borrower that exhibited a level of performance greater than or equal to that of the heuristic borrower. The performance was evaluated in two types of simulations that represented different load characteristics.

The first load characteristic represented a group of hosts moving together with the same velocity. This is analogous to a group of freeway traffic, for example. Firstly, the heuristic performance was evaluated against a no-borrowing policy. The channel-borrowing heuristic performed better than the no-borrowing policy. The results of this simulation were somewhat obvious. When the GA channel borrower was used, the performance was as poor as the no-borrowing policy. It was determined that the mutation operator needed to be improved. A new mutation operator was developed and the performance of the GA was on a par with that of the channel-borrowing heuristic.

The second load characteristic represented a large load diffusing random motion. The results were interesting in that the channel-borrowing heuristic did not perform much better than the no-borrow policy. This demonstrated that trying to execute channel borrowing in an environment of randomness does not yield any great increase in performance. The GA also exhibited the same behavior, but, interestingly, it improved on the heuristic at some points. This indicated that the new mutation operator performed well.

Finally, it was determined that the GA method with improved mutation operator worked well even for a small number of iterations and population. The results show that a modified mutation rate of 0.1 produced the best results. The results show promise for this class of problems.

ACKNOWLEDGMENTS

Professor Zomaya's work was supported by the Australian Research Council.

REFERENCES

[1] G. Calhoun, *Digital Cellular Radio.* Artech House, 1988.
[2] A. Chipperfield and P. Fleming, "Parallel Genetic Algorithms," *Parallel and Distributed Computing Handbook,* A. Zomaya, ed., pp. 1118-1193, 1996.
[3] K. Feher, *Wireless Digital Communications.* McGraw-Hill, 1995.
[4] M.A.C. Gill and A.Y. Zomaya, *Obstacle Avoidance in Multi-Robot Systems.* World Scientific, 1998.
[5] S. Grandhi, J. Zander, and R. Yates, "Constrained Power Control," *Int'l J. Wireless Personal Comm.,* vol. 2, pp. 257-270, 1995.
[6] S. Grandhi, R. Yates, and D. Goodman, "Resource Allocation for Cellular Radio Systems," *IEEE Trans. Vehicular Technology,* vol. 46, no. 3, pp. 581-587, 1997.
[7] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, Mass.: Addison-Wesley, 1989.
[8] D.E. Goldberg, "Sizing Populations for Serial and Parallel Genetic Algorithms," *Proc. Third Int'l Conf. Genetic Algorithms,* pp. 70-79, 1989.
[9] J.H. Holland, *Adaptation in Natural and Artificial Systems.* Michigan: Univ. of Michigan Press, 1975.
[10] D. Levine, "Users Guide to the PGAPack Parallel Genetic Algorithm Library," Technical Report ANL-95/18, Argonne Nat'l Laboratory, 1996.
[11] K.Y. Lim, M. Kumar, and S.K. Das, "Message Ring-Based Channel Reallocation Scheme for Cellular Networks," *Proc. Int'l Symp. Parallel Architectures, Algorithms, and Networks,* pp. 426-431, 1999.
[12] W.C.Y. Lee, *Mobile Cellular Telecommunications Systems.* McGraw-Hill, 1989.
[13] I. Katzela and M. Naghshineh, "Channel Assignment Schemes for Cellular Mobile Telecommunication Systems: A Comprehensive Survey," *IEEE Personal Comm.,* vol. 31, no. 3, pp. 10-31, June 1996.
[14] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs,* second ed. Berlin: Springer-Verlag, 1994.
[15] A.H.M. Ross, "CDMA Topics," http://www.whiterabbit.com/~ahmrphd/Topics.html, 1996.
[16] S. Salleh and A.Y. Zomaya, *Scheduling in Parallel Computing Systems: Fuzzy and Annealing Techniques.* Mass.: Kluwer Academic, 1999.
[17] J. Scourias, "Overview of the Global System for Mobile Communications," http://ccnga.uwaterloo.ca/~jscouria/GSM/gsmreport.html, 1999.
[18] A. Sampath, P.S. Kumar, and J.M. Holtzman, "Power Control and Resource Management for a Multimedia Wireless CDMA System, Personal, Indoor and Mobile Radio Communications," *Proc. Int'l Symp. Personal, Indoor, and Mobile Radio Comm.,* vol. 1, pp. 21-25, Sept. 1995.
[19] A. Yener and C. Rose, "Genetic Algorithms Applied to Cellular Call Admission Problem: Local Policies," *IEEE Trans. Vehicular Technology,* vol. 46, no. 1, pp. 72-79, 1997.
[20] *Solutions to Parallel and Distributed Computing Problems: Lessons from Biological Sciences,* A.Y. Zomaya, F. Ercal, and S. Olariu, eds., New York: Wiley, 2001.

Albert Y. Zomaya is currently the CISCO Systems Chair Professor of Internetworking in the School of Information Technologies, The University of Sydney. Prior to that, he was a full professor in the Electrical and Electronic Engineering Department at the University of Western Australia, where he also led the Parallel Computing Research Laboratory from 1990-2002. He served as associate-, deputy-, and acting-head in the same department, and held visiting positions at Waterloo University and the University of Missouri-Rolla. He is the author/coauthor of five books, 150 publications in technical journals and conferences, and the editor of three books and seven conference volumes. He is currently an associate editor for the *Journal of Interconnection Networks*, the *International Journal on Parallel and Distributed Systems and Networks*, *Journal of Future Generation of Computer Systems*, and the *International Journal of Foundations of Computer Science.* He is also the founding editor of the Wiley Book Series on Parallel and Distributed Computing. He is the editor-in-chief of the *Parallel and Distributed Computing Handbook* (McGraw-Hill, 1996). He is the chair of the IEEE Technical Committee on Parallel Processing (1999-Present). He is also a board member of the International Federation of Automatic Control (IFAC) committee on Algorithms and Architectures for Real-Time Control, and serves on the executive board of the IEEE Task Force on Cluster Computing. He has been actively involved in the organization of national and international conferences. Professor Zomaya is a chartered engineer, a fellow of Institution of Electrical Engineers (UK), a senior member of the IEEE, and member of the ACM. He received the 1997 Edgeworth David Medal from the Royal Society of New South Wales for outstanding contributions to Australian Science. In September 2000, he was awarded the IEEE Computer Society's Meritorious Service Award. His research interests are in the areas of high performance computing, parallel and distributed algorithms, mobile computing, and networking.

Michael Wright received the BE degree (majoring in information technology) from the University of Western Australia in 2001. He is currently a software developer with Motorola in Perth, Western Australia. His research interests are the Internet, networking, and parallel computing.