Improved Genetic Algorithm for Channel Allocation with Channel Borrowing in Mobile Computing

Somnath Sinha Maha Patra, Kousik Roy, Sarthak Banerjee, and Deo Prakash Vidyarthi

Abstract—This paper exploits the potential of the Genetic Algorithm to solve the cellular resource allocation problem. When a blocked host is to be allocated to a borrowable channel, a crucial decision is which neighboring cell to choose to borrow a channel. It is an optimization problem and the genetic algorithm is efficiently applied to handle this. The Genetic Algorithm, for this particular problem, is improved by introducing a new genetic operator, named pluck, that incorporates a problem-specific knowledge in population generation and leads to a better channel utilization by reducing the average blocked hosts. The pluck operator makes the crucial decision of when and which cell to borrow with the future consideration that the borrowing should not lead the network to chaos. It makes a channel borrowing decision that minimizes the number of blocked hosts and improves the long-term performance of the network. Efficacy of the proposed method has been evaluated by experimentation.

Index Terms—Cell, channel allocation, genetic algorithm, mobile communication, reuse.

1 Introduction

The widening availability of mobile information systems is being driven by the increasing demand to have information available for users anywhere and anytime. The load on available radio frequency resources increases with the increase in mobile devices and users. The radio frequency spectrum is limited and a need arises for the effective management of these radio resources. This work evaluates an approach for managing these resources such that their availability is maximized and, hence, the disruption to service is minimized [1].

The Genetic Algorithm (GA), useful for optimization problems, is based on the Darwin's theory of "survival of the fittest." Individuals, from the population of potential solutions, reproduce and solutions are refined successively over the number of generations. In the recent past, the application of GA has attracted the attention of researchers of numerous disciplines (e.g., operation research, economics, social sciences, life sciences, etc.) for problem solving [4], [13].

Researchers of mobile computing have used GA for the channel allocation problem. Zomaya and Wright used it for channel allocation and compared it to a greedy borrowing heuristic [1]. Kassotakis et al. proposed Hybrid GA for reusing isochronous channels in multiple access telecommunication networks [2]. An evolutionary genetic DCA for resource management is proposed by Asvial et al. that aims to provide optimum channel allocation for specified interference constraints using minimum cost as a metric [3].

The work described here exploits GA for the effective management of radio resources in dynamic channel allocation with channel borrowing. Introducing a new genetic operator named "pluck," an improved GA is proposed for the purpose. The principle of pluck is derived from the tea leaves plucking operation of the tea garden that sharply considers the present and future plucking.

The next section discusses the channel allocation problem with a few important issues in mobile computing. A few related models are also given. In Section 3, a short discussion over the Genetic Algorithm is made, followed by some GA-based channel allocation models. In Section 4, the proposed Improved GA (IGA) for channel allocation is described. Section 5 lists the experiments conducted for the channel allocation with their comparisons and, finally, a conclusion is drawn in Section 6.

2 Channel Allocation in Mobile Computing

Wireless networking greatly enhances the utility of portable computing devices. The technical challenges that mobile computing must surmount to achieve this potential are hardly trivial. The main issues stem from three essential properties of mobile computing, i.e., communication, mobility, and portability.

Wireless networks communicate by modulating radio waves or pulsing infrared light. Wireless networks are linked to the wired network infrastructure by stationary transceivers. The area covered by an individual transceiver's signal is known as cell. Cell radii vary from tens of meters in buildings to hundreds of meters in cities or tens of kilometers in the countryside. Cellular systems use small cells due to frequency reuse, less transmission power, and thin interference.

- *S.S. Maha Patra is with Network Management System, UTStarcom Inc., 7th Floor, Signature Towers-B, South City-I, Gurgaon, Haryana. India. E-mail: ssmp2001@yahoo.com.*
- *K. Roy can be reached at House Number: 343, Sector 22, Gurgaon, Haryana, India. E-mail: kushgrahi@yahoo.co.in.*
- *S. Banerjee is with Newgen Software Technologies Ltd., D-152 OKHLA, Phase 1, NewDelhi-110020, India. E-mail: sarthak@newgen.co.in.*
- *D.P. Vidyarthi is with the School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi, India. E-mail: dpv@mail.jnu.ac.in or dpvidyarthi2002@yahoo.com.*

Manuscript received 9 May 2004; revised 30 Sept. 2004; accepted 27 Mar. 2005; published online 16 May. 2006.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0156-0504.

The ability to change locations yet connected to the network increases the possibility of the volatility of some information. Certain data considered static for stationary computing becomes dynamic for mobile computing. Moving mobile hosts will use different network access point or "addresses" [10] and, so, needs more location sensitive information than stationary devices. The handover is a common problem that arises due to mobility [9].

Over the years, not only have the number of mobile devices increased, but the applications running on the mobile devices are also getting more data driven. Due to increasing load, the number of mobile hosts that could not connect to the destination is increased. There are two ways to solve this problem. One is to increase the number of channels (radio frequency) with the corresponding increase in cost. The other is to utilize the current infrastructure efficiently so that the best performance is achievable. Obviously, the second option is better and preferable.

In a mobile network, the number of wireless channels is usually limited and is reused. Channel reuse is defined as: If one channel is outside the interference range of other, it can reuse the same frequency and, thus, utilize the scarce resource. The efficient reusability of the channels improves the performance of the network. In Fixed Channel Allocation (FCA), the assignment of frequencies to a cell is fixed. This is inefficient if the traffic load on a channel varies from time to time and, thus, Dynamic Channel Allocation (DCA) is often used instead. A better method, in case of heavy load in one cell and light load in neighboring cell, is to borrow frequencies from neighbor cells. Cells with more traffic are dynamically allotted more frequencies. This scheme is known as Borrowing Channel Allocation (BCA) and is used in GSM systems; however, it requires careful traffic analysis. Other methods to deal with excess load in mobile networks in addition to channel borrowing are channel sharing and cell splitting [7]. Particularly, cell splitting is very common in many real cellular networks [8].

The channel allocation problem involves how to allocate borrowable channels in such a way that it maximizes the long term and/or short-term performance of the network [6]. The performance metric in the present work, to evaluate the proposed IGA for channel allocation, is the number of blocked hosts. A host is blocked if it enters into a cell but cannot get a channel. It is obvious that more blocked hosts result in performance degradation of the network. Thus, reduction in blocked hosts is sought in the present work. The number of borrowings should also be as few as possible because more borrowings incur more network traffic. The other metric that can be used to evaluate the performance is the number of "hot cells," a cell without any free channel [1]. A cell is considered "hot" when all the channels in a cell are allocated to hosts. Hot cells are undesirable because mobile hosts will be denied a service upon entering the cell.

2.1 Related Models

Some relevant channel allocation models that use techniques other than channel borrowing are described below.

2.1.1 Cell Splitting

Cell splitting, used in many real cellular networks [8], works by breaking down cells into smaller cells. This is accomplished by having several different levels of cell coverage. These different levels are called macro and micro cells. A macro cell is essentially an umbrella over a set of micro cells. There are obvious drawbacks to this scheme as it prevents being implemented throughout the network because of cost involved. A secondary concern is the extra traffic introduced by the additional cells [1].

2.1.2 Least Interference

In the least interference method, the Access Point (cell) scans all available channels and selects the channel with the lowest interference power. If more than one channel shares the same lowest interference power, the channel used previously will be selected and, if none were used previously, the channel with the lowest number is selected [11].

2.1.3 Channel Segregation

In the channel segregation method, an ordered list is given to each Access Point (AP) and this list is updated according to interference conditions. The AP will scan the interference power for the highest priority channel in the ordered list. If the scanned interference power is below a threshold, this channel will be selected and the scanning process ends. The priority of each channel in the list is updated by a given function. The initial priority list can be arbitrary and is set to follow the channel number [11].

3 Genetic Algorithm

A Genetic Algorithm is a search procedure based on the principle of evolution and natural genetics. GA combines the exploitation of past results with the exploration of new areas of the search space. By using the survival of the fittest technique combined with a structured yet randomized information exchange, a GA can mimic some of the innovative flair of human search.

A GA 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 from good measure. GAs are randomized, but they are not simple random walks. They efficiently exploit historic information to speculate on new search points with expected improvement [1], [4], [5].

The majority of optimization methods move from a single point in the decision space to the next using some transaction rule to determine the next point. This method may be harmful as it can locate a false peak in multimodal (many-peaked) search spaces. By contrast, GA works from a database of points simultaneously (a population of strings), climbing many peaks in parallel [1].

In GA, we start with an initial population and then we use some genetic operators on it for appropriate mixing of exploitation and exploration. A simple genetic algorithm consists of an initial population followed by selection, crossover, and mutation [5]. Selection operation selects the best results among the chromosome through some fitness function. The idea of the crossover operation is to swap some information between a pair of chromosomes to obtain a new chromosome. In mutation, a chromosome is altered a little bit randomly to get a new chromosome. The simple structure of the GA is:

```
GA( )
{    Initialize population;
     Evaluate population;
     While termination criterion not reached
     {    select solutions for next population;
          perform crossover & mutation;
          evaluate population;
     }
}
```

3.1 Related GA-Based Channel Allocation Models

Wong and Wassell proposed a dynamic channel allocation model using GA for a broadband fixed wireless access network. In the proposed model, the aim is to allocate the channel so as to reduce the Signal to Noise Ratio (SNR) and, at the same time, meet the traffic demands. They compared their model with Least Interference and Channel Segregation models of channel allocation and are able to show that their GA-based model achieves an SNR gain as compared with the other two methods [11].

A hybrid genetic (HGA) approach for channel reuse in multiple access telecommunication networks is proposed by Kassotakis et al. [2]. They combined GA with a local search algorithm to ensure the reliability property of GA with the accuracy of the hill-climbing method. The performance of the proposed HGA is compared, via simulation, to that of the graph coloring algorithm (GCA) of [12] and it is concluded that GCA's performance is comparable to that of HGA at light/medium load, while HGA solutions massively outperform GCA once at heavy network load.

The evolutionary genetic DCA for resource management in mobile systems is proposed by Asvial et al. The proposed chromosome structure is the combination of traffic load and interference limited DCS. The constraints in interference used in the algorithm include the cosite interference, the co-spotbeam interference, and the adjacent co-spotbeam interference. Total interference is proposed to be minimized in the model [3].

Zomaya and Wright proposed a GA-based DCA model in which they modified the mutation genetic operator of GA for channel allocation problem. The fitness function used considers the hotcells, current borrowings, and present and future blocking. They compared their model with FCA and greedy borrowing heuristics for the average number of blocked channel metric and are able to show that the model works better than FCA and has a slight edge from the heuristic model [1].

4 Improved GA for Channel Allocation

In the proposed work, GA is applied for channel allocation in DCA with channel borrowings. We introduce a new genetic operator, "pluck," for improving the simple GA and call the proposed method an Improved Genetic Algorithm (IGA) for the purpose. The idea behind the pluck operation is to add some knowledge by selecting some chromosomes that may not be giving good results right now but may lead the system to a stabilized state for better results in the future. The various phases of the IGA are explained as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13

Fig. 1. Gene structure.

4.1 The Encoding Scheme

One of the most important aspects that control GA's performance is the encoding method chosen. The encoding refers to the method by which the problem parameters are mapped into a chromosome.

There are number of ways to encode the chromosome. Here, a gene is created for each cell. Each gene contains the information about the number of hosts and the number of free channels in the cell. It also contains the information where to borrow, how much to borrow, and to whom to give and how much. The environment considers each cell with six neighbors and, for each of them, we have the information at borrow-side and lending-side. After encoding the gene for each cell, genes are combined into a super gene, which contains information of the whole network.

A gene is an array of length 14, as shown in Fig. 1. At the first location of the array, we keep the number of blocked hosts and the second one has the number of free channels. The next six locations contain the information about lending to six neighbors. The last six locations have the information about borrowing from six neighbors. A super gene is formed with the gene of a cell and the gene of its six neighboring cells. Hence, the super gene is a matrix of 7×14 and all GA operations are performed on this super gene, i.e., matrix.

4.2 The Pluck Operation

It has been found that incorporating problem specific knowledge in GA improves its performance [13], [14]. A new operation for channel allocation that incorporates knowledge and an idea which is borrowed from the plucking of tea leaves in a tea garden is introduced. It is observed that the performance of the channel allocation model is improved drastically by the new operator. This operation is added considering the future channel borrowings. The idea for the "pluck" is as follows:

In a tea garden, workers pluck tea leaves. They first verify the leaf. Is it a good leaf to pluck? If yes, will it be better the next day? If the leaf is good but it may be better the next day, the workers do not pluck it and leave it for a better result. In the case of another leaf, it is not as good as the previous one, but it may get worse the next day if the worker plucks it to lead to a better condition in the future.

The proposed pluck operation is similar. When one cell tries to borrow some channels from a neighbor cell, it checks whether it is the best time to borrow. If the cell finds that it may give better results for the next, considering the environment, then the cell does not borrow the channel from this neighbor. If it finds that it may lead to a bad situation if it is borrowed later, then the cell tries to borrow it at the appropriate time. This future information (i.e., the information about free and blocked hosts) is extracted from the cell itself. Thus, pluck, together with the encoding method, produces the population for other GA operations.

4.3 Crossover and Mutation

Supergenes are formed in a matrix as discussed earlier. Breeding between two matrices produces two new matrices as the offspring of the parental matrices. From each matrix, select two rows, one from the first and the other from the second. Take a cut point randomly, which is a point between 1 and the highest column number. Divide each row (on which we are applying the crossover) in two parts. The first part is one before the cut point and the second part is the elements in the rows after the cut point. When the two rows meet, these two parts act differently and generate two new rows of the offspring.

Elements before the cut point are swapped with each other. Suppose the chosen cut point is 3. If, in the first row before the cut point, the elements are 1 2 3 and, in the second row, the elements before the cut point are 5 6 7, then in the two offspring rows, generated from them, the first three elements of the first offspring will be 5 6 7 and the first three elements of the second row will be 1 2 3.

For the elements after the cut point, mating is done in a different fashion. We search the first parental row after the cut point in the first matrix for which the elements are common with the elements after the cut point of the first row in second matrix. Then, the order in which the common elements occur in the row of the second matrix is listed. Reposition the common elements in the order as they occur in the second matrix. For the elements, which are not common with the row of the second matrix, keep them in the same position as they are in the first parental row before the crossover. Thus, the elements after the first three elements of the offspring one are generated. Now, the entire row of the first offspring is formed. In the same manner, the elements of the second offspring are generated from the elements after the cut point of the second parental row and the elements after the cut point of the first parental row.

Thus, two new offspring rows forming two existing parental rows are generated and we have two more options to choose the best among them. By applying the procedure on each row of the matrix, two new offspring matrices are generated, making four genes to choose the best. The operation is elaborated by an example for the following two parental matrices:

$$\begin{array}{ccccccc} \multicolumn{7}{c}{\text{1st matrix}} \\ & & & \multicolumn{4}{l}{\downarrow \text{cut point}} \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 4 & 3 & 4 & 8 & 7 \\ 5 & 6 & 7 & 5 & 6 & 4 & 3 \end{array}$$

$$\begin{array}{ccccccc} \multicolumn{7}{c}{\text{2nd matrix}} \\ & & & \multicolumn{4}{l}{\downarrow \text{cut point}} \\ 4 & 5 & 6 & 5 & 7 & 4 & 6 \\ 4 & 3 & 5 & 8 & 7 & 9 & 4 \\ 1 & 2 & 2 & 6 & 6 & 3 & 4 \end{array}$$

The offspring generated from two parental matrices are:

$$\begin{array}{ccccccc} \multicolumn{7}{c}{\text{offspring 1}} \\ 4 & 5 & 6 & 5 & 7 & 4 & 6 \\ 4 & 3 & 5 & 3 & 8 & 7 & 4 \\ 1 & 2 & 2 & 5 & 6 & 3 & 4 \end{array}$$

$$\begin{array}{ccccccc} \multicolumn{7}{c}{\text{offspring 2}} \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 4 & 4 & 8 & 9 & 7 \\ 5 & 6 & 7 & 6 & 6 & 4 & 3 \end{array}$$

Here, the two offspring are generated from two parental matrices. Suppose the cut point taken is 3. Up to the third column, in each row of the two parental matrices, the elements are before the cut point. These parts of each matrix are swapped when the rows of the parental matrices meet. For example, in the first row of the first matrix, the elements are 1 2 3 and, in the second row, these are 4 5 6. In the offspring, they are swapped and, in offspring 1, the first three elements are 4 5 6 and, in offspring 2, they are 1 2 3.

The last four elements of a row are elements after the cut point. For the two parental matrices, it is verified that if there are common elements after the cut point in the two corresponding rows, those common elements of the first parental matrix are in which order in the corresponding row of the second parental matrix. In offspring 1, the common elements of the first parental matrix are kept in the same order as they are in the corresponding row of the second parental matrix. The uncommon elements are kept in the same position in the offspring as they are in the parental matrices. This is done so as to preserve the necessary information as much as possible.

In the example parental matrices, the elements after the cut point in the first row of the first parental matrix are 4 5 6 7 and, for the second parental matrix, they are 5 7 4 6. So, the elements of offspring 1, generated from these two rows, are 5 7 4 6 and, for offspring 2, are 4 5 6 7.

The second rows of parental matrices after the cut point are 3 4 8 7 and 8 7 9 4. So, when offspring is generated from these elements, the uncommon elements are kept in the same position and common elements change according to the order of the corresponding row of the parental matrix. So, the elements for offspring 1 are 3 7 8 4 and the elements for offspring 2 are 8 4 9 7. Complete rows of the offspring are generated from the parental matrices and offspring matrices are formed.

The mutation operator in GA is used to introduce new genetic material. As a result of its generality, it has several weaknesses. The main weakness is taking borrowing decisions ahead of time that may result in nonoptimality for two reasons. First, their effectiveness is not measured in the fitness function and, second, these decisions degrade the future quality of service [1]. The probability of applying mutation operation is often very small. In the proposed channel allocation model, the probability of applying mutation is assumed to be zero because of the same reason.

4.4 The Fitness Function

The fitness function is used to rank the quality of a chromosome. A fitness value is assigned to a chromosome by a fitness function and a chromosome is evaluated with this value for survival. The fitness function, used in this problem, is as below:

$$\begin{aligned} Fitness = {} & \alpha_c * current\_blocked + \alpha_n * next\_blocked + \\ & \beta * next\_free + \mu * hotcell + \lambda * transaction. \end{aligned} \tag{1}$$

Here, α_c, α_n, β, μ, and λ are constant values to suit the environment.

Next_blocked and next_free are calculated by the behavior variable of the cell and the host. Hotcell is taken into account when the cell has no free channel. Transaction is the number of blocked or free hosts of the cell.

The fittest gene, in this case, is the gene with the lowest fitness value.

4.5 Repair

This operation is done for keeping the cell information relevant. It is applied on the object after the crossover operation to recalculate the cell information.

Check the "hostnumber" that keeps the value of the number of the host. If the hostnumber is greater than the available free channel, then there will be no further available free channel and the free channel number is updated to be zero.

There will also be change in the lending part of the gene. As the cell with no free channel cannot spare an extra channel to any neighbor, this part is updated to be zero. For those cells from which the current cell will borrow the channels, their information is also to be updated. If it borrows from some neighbor, the neighbor's corresponding lending column of the gene should be updated.

For the situation when a host number is less than the number of currently free channels, it need not borrow from its neighbor. So, its borrowing part becomes zero. Now, it can distribute its extra free channels over its neighbors. So, for the neighbors that take an extra channel from it, their borrowing part information is updated in the same way. Taking the current neighbor and the neighbor index to identify to which it gives its extra channels and updating their borrowing information does it. It also updates its lending part information.

4.6 The Algorithm

The above operations of the IGA are summarized in the given algorithm. These are near to the programming language, specifically Java, in which the software is written.

1. Define the cell array, host array, and 3D float array called supergene. Cell array is for the cells, host array is to represent the mobile hosts in the network, and supergene is a collection of genes defined in cell. Supergene contains genes of a particular cell and its neighbors and is used for genetic operations.
2. Input number of mobile_hosts and clock_pulse.
3. Assign all the hosts in a single cell OR distribute it uniformly among the cells.
4. For each cell enter,
    a. average number of hosts in a cell,
    b. increase rate of hosts if number of hosts are above average,
    c. increase rate of hosts if number of hosts are below average,
    d. decrease rate of hosts if number of hosts are above average, and
    e. decrease rate of hosts if number of hosts are below average.
5. Set clock_pulse_index = 0.
6. In the beginning total_blocked_hosts = 0.
   Repeat Steps 7 to 18 until clock_pulse_index is equal to clock_pulse.
7. Calculate the number of free_channels and blocked_hosts of each cell.
8. Resource_Deallocation( ).
9. Calculate the number of free_channels and blocked_hosts for current modified information.
10. Create Initial_Population using Pluck( ).
11. Perform Crossover( ).
12. Perform Repair( ).
13. Calculate Fitness( ) from (1).
14. Select the best result and set the corresponding supergene as the current gene.
15. Calculate again the free_channels and blocked_hosts of each cell.
16. Show the result, i.e., the number of blocked_hosts found during current pass.
17. clock_pulse_index = clock_pulse_index + 1.
18. total_blocked_hosts = total_blocked_hosts + blocked_hosts.
19. Calculate average_blocked_hosts = total_blocked_hosts/number of clock pulses.
20. Show average_blocked_hosts.

4.7 Resource_Deallocation( )

At the start of the present iteration, it may happen that the needy cells of the last iteration need to borrow channels from its neighbors and, so, this cell will return the borrowed channels. On the other hand, if the donor cell of the past iteration needs the channels for its own hosts in the present iteration, it will take the donated channels back from the corresponding neighbor.

Other functions used in the algorithm, e.g., encode, crossover, repair, etc., are discussed above.

5 Experiments

The performance of the proposed method has been evaluated by obtaining the average number of blocked hosts in the network. We investigate the performance of a fixed channel allocation scheme also. The performance is further evaluated against the results obtained by Zomaya and Wright [1] for the number of blocked hosts.

All the cases have a finite runtime and a variable time-step. Basic input data for the experimental environment are

- Total number of cells: 19.
- Channel per cell: 10.
- Crossover probability: 1.
- Number of hosts in the network: 50, 100.
- Number of time units in seconds: 10, 20, 50, 100, and 150.

In the beginning, all hosts are assumed to be in cell no. 1. During various cases, a host may move only to its neighbor cell. A host may change its cell depending upon three conditions: the host's desire to change its cell, the outgoing cells' desire to leave the host, and the incoming cells' desire to take. The host's desire and which neighbor it is willing to go to are generated randomly and the other two depend on the values supplied by the user. In the performance graphs, the x-axis represents instances of time and the y-axis represents the number of blocked hosts.

5.1 Experiment 1

We performed an experiment with 50 hosts and 10 units of time. The performance of the FCA scheme and the GA-based scheme is observed.

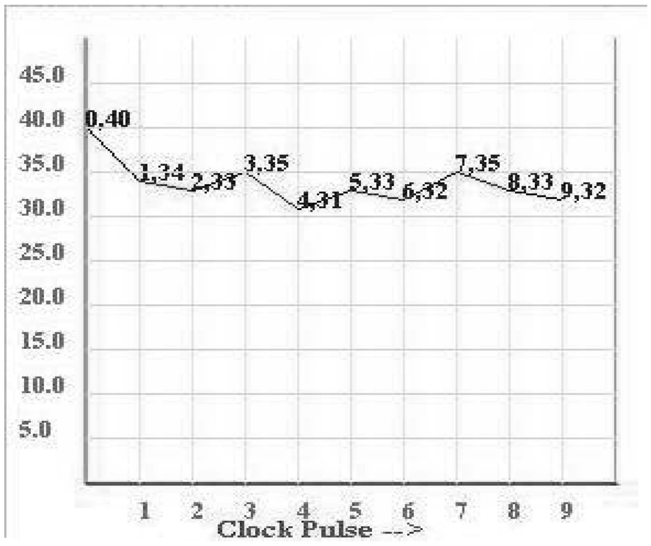

Fig. 2. FCA with 50 hosts.

5.1.1 *Case 1*

Case 1 is designed to see the performance of the FCA scheme with 50 hosts during 10 units of time. The performance graph is given in Fig. 2.

Point 1 indicates the situation after one unit of time and the second number, in the graph, is the number of blocked hosts on that instance and so on. Here, the average blocked host is 34.1, i.e., 34 hosts will not get service on average in the FCA scheme.

5.1.2 *Case 2*

Case 2 is designed to see the performance of IGA scheme with 50 hosts during 10 units of time. The performance graph is given in Fig. 3.

Here, the average blocked host is 9.7, i.e., on average, 10 hosts will not get service in the IGA-based scheme.

Results indicate that IGA is incomparable with FCA.

5.2 Experiment 2

The experiment is performed with same number of hosts, i.e., 50 hosts over 20 units of time to observe the number of average blocked hosts over time.

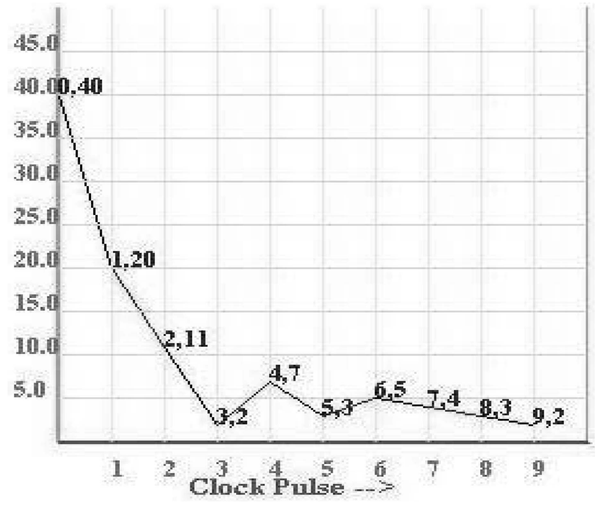

Fig. 3. IGA with 50 hosts.

5.2.1 *Case 3*

Case 3 is designed to see the performance of the FCA scheme with 50 hosts during 20 units of time. The performance graph is given in Fig. 4.

Here, the average blocked host is 29.7, i.e., 30 hosts will not get service on average in the FCA scheme.

5.2.2 *Case 4*

Case 4 is designed to see the performance of the IGA scheme with 50 hosts during 20 units of time. The performance graph is given in Fig. 5.

Here, the average blocked host is 5.05, hence, five hosts wouldn't get service in the GA scheme.

Both experiments 1 and 2 show that the performance of IGA is much better than FCA for 50 hosts. Further, we observe that, over the time period (from 10 to 20), the number of average blocked hosts reduces. Reduction is much sharper in IGA.

5.3 Experiment 3

Now, experiments with 100 hosts and 10 units of time are performed. The performance of the FCA scheme and the GA-based scheme is observed.

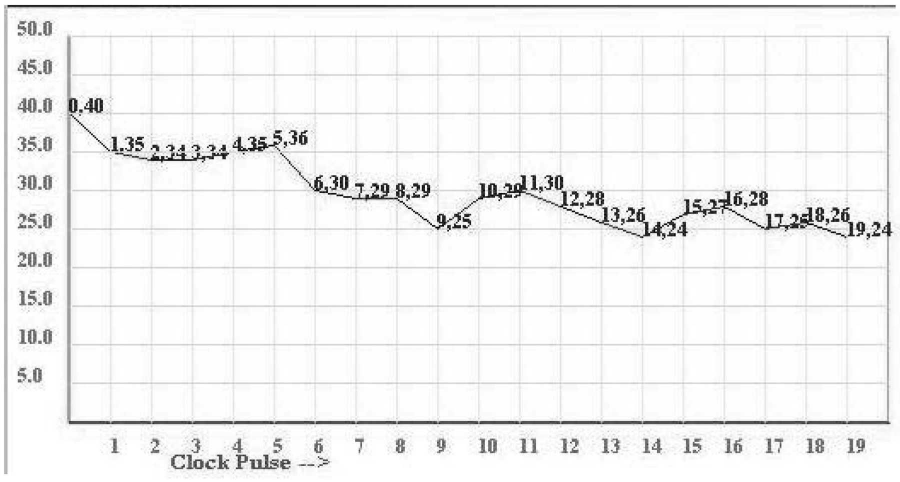

Fig. 4. FCA with 50 hosts

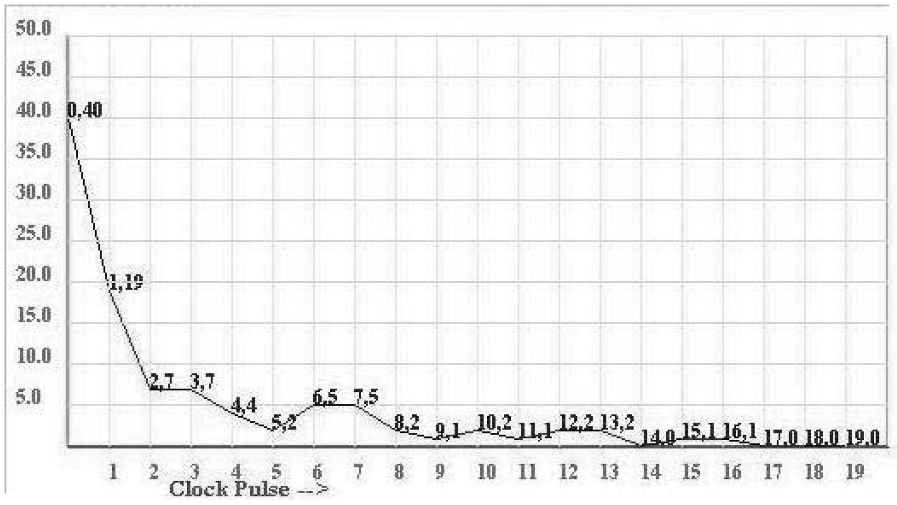

Fig. 5. IGA with 50 hosts.

5.3.1 *Case 5*

Case 5 is designed to see the performance of the FCA scheme with 100 hosts during 10 units of time. The performance graph is given in Fig. 6.

Here, the average blocked host is 75.6 in the FCA scheme.

5.3.2 *Case 6*

Case 6 is designed to see the performance of the IGA scheme with 100 hosts during 10 units of time. The performance graph is given in Fig. 7.

The average blocked host is 21.7, which means 22 on average in the GA scheme.

5.4 Experiment 4

Finally, experiments with 100 hosts and 20 units of time are performed and the performance of the FCA scheme and the GA-based scheme is evaluated.

5.4.1 *Case 7*

Case 7 is designed to see the performance of the FCA scheme with 100 hosts during 20 units of time. The performance graph is given in Fig. 8.

Here, the average blocked host is 70, which means 70 hosts are blocked.

5.4.2 *Case 8*

Case 8 is also designed to see the performance of the IGA scheme with 100 hosts during 20 units of time. The performance graph is given in Fig. 9.

Here, the average blocked host is 13.3, which means 13 hosts are blocked.

Another observation that can be made from the graph in Fig. 9 is that, over the time period, the average number of blocked hosts stabilizes (see point 8 to 15 in Fig. 9 and point 14 to 19 in Fig. 5).

5.5 Comparison of Results

For a cumulative comparison, an experiment is carried out for 50, 75, and 100 hosts over 50, 100, and 150 time periods. Results are derived and compared in Fig. 10.

From the above graph, it is evident that IGA outperforms FCA. If the numbers of hosts are more in the mobile network, the average number of blocked hosts reduces sharply. Further, it is also inferred that, over the time

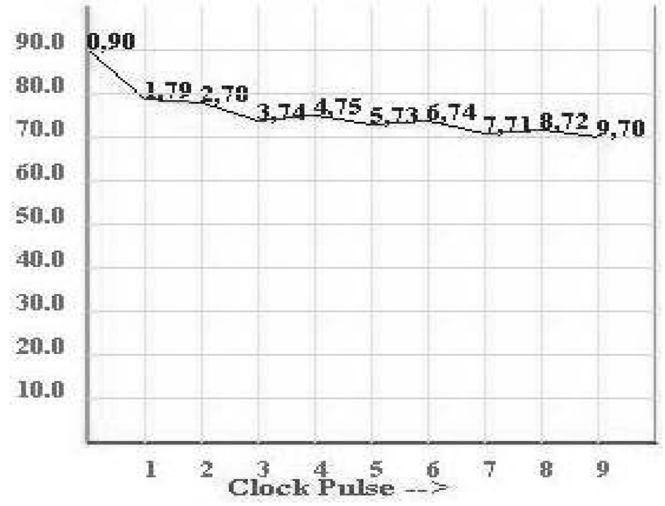

Fig. 6. FCA with 100 hosts.

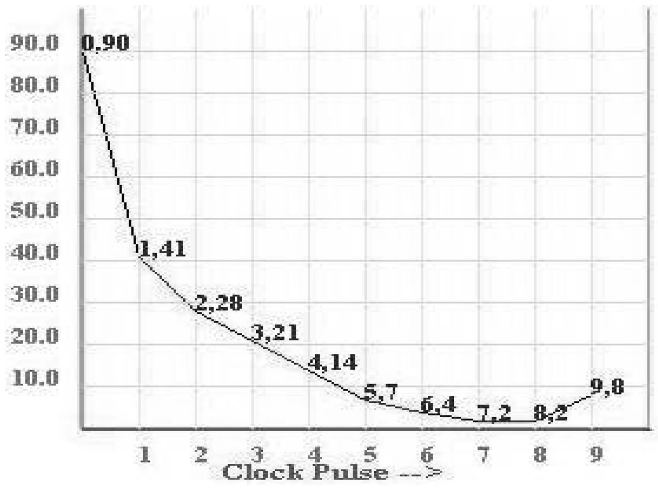

Fig. 7. IGA with 100 hosts.

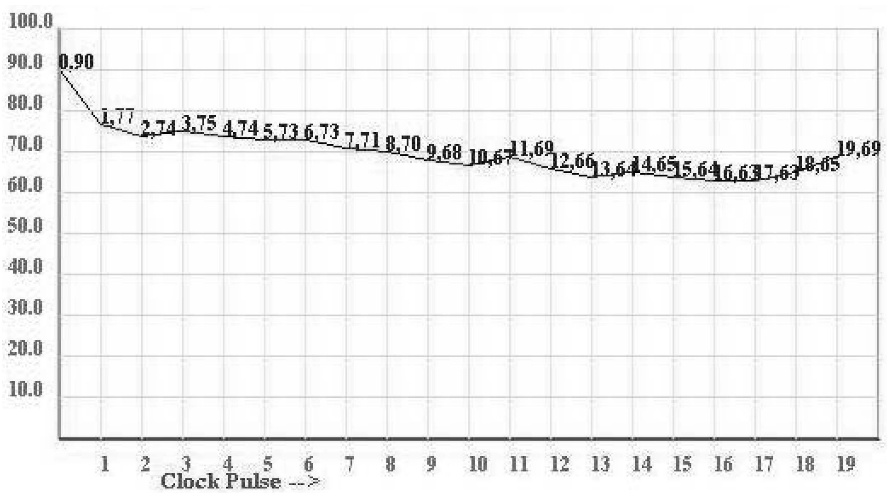

Fig. 8. FCA with 100 hosts.

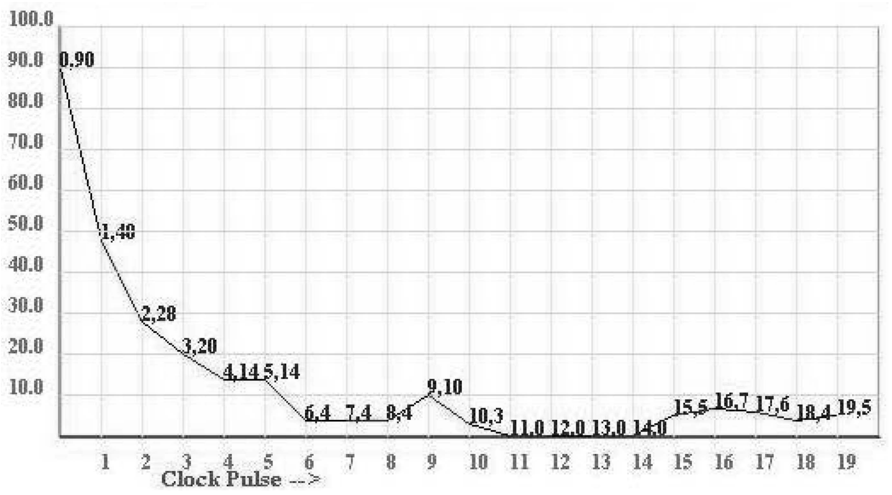

Fig. 9. IGA with 100 hosts.

period, the network reaches a more stable state with the reduction in blocked hosts.

Results obtained by IGA are compared with those obtained by Zomaya and Wright [1] in which the mutation operation is modified for reducing the blocked hosts (simulation 7 of [1]). A graph is shown in Fig. 11 for the number of blocked hosts to compare FCA, modified mutation GA, and IGA.

The performance of IGA over the modified mutation GA is conspicuous. It is obvious that both IGA-based and modified mutation GA-based channel allocation schemes outperform FCA, however, IGA has an edge over the modified mutation GA. Further, up to 20 time units, FCA was giving better results than modified mutation GA, but IGA was performing better since the beginning. This supports our idea that IGA is intended to perform better under all conditions.

6 Conclusion

This work aims at exploring the GA and improving it to solve the cellular resource allocation problem. It presents a different approach of handling the resource allocation problem by incorporating problem specific knowledge. The method basically depends on discriminatory channel borrowing. The study treats mobile hosts, consuming radio resources, as a trivial entity. The hosts roam in a mobile network, which is cellular and regular. When a host enters into a new cell, a hand-off is made for a channel. If there are channels available, it can get a free channel immediately, otherwise, a channel borrowing operation is endeavored. A decision arises concerning from which neighboring cell to borrow a channel from.

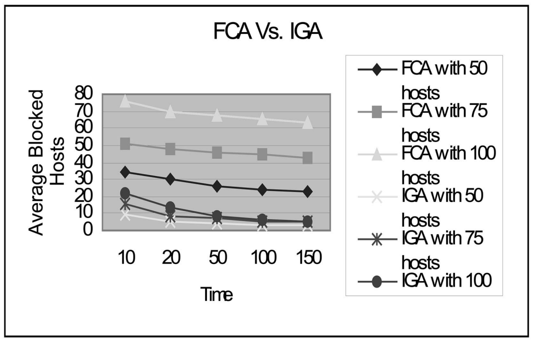

Fig. 10. FCA versus IGA for varying hosts and time units.

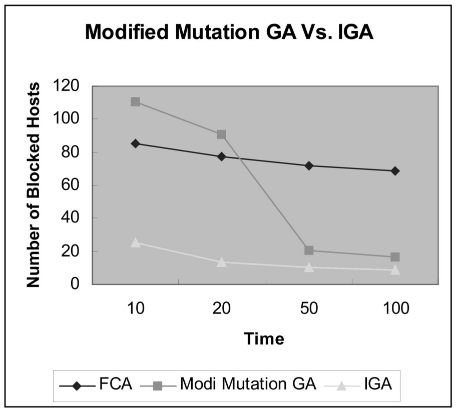

Fig. 11. Modi mutation GA versus IGA.

A new operation named pluck is proposed. It borrows cells and/or channels with the future consideration derived from past allocation and the cell properties. It always considers those cells or channels that never lead to a worse condition and, at the same time, will try to result in a better situation as far as the host blocking is concerned. This study attempts to improve the GA by adding a pluck operation to help make channel borrowing decisions that minimize the number of blocked hosts and maximize the long-term performance of the network. The quantification of the pluck operation is set to be the future course of work.

The comparison of results with the model proposed by Zomaya reveals that the IGA model has an edge for improving the channel utilization. Thus, the overall study suggests that the proposed allocation model will outperform the earlier channel allocation models in terms of reduction in average blocked hosts. The consequence is better channel utilization.

Acknowledgments

This work is undertaken as part of the project work carried out by the MSc students of the Department of Computer Science, Banaras Hindu University, Varanasi. The authors are thankful to the Department of Computer Science, BHU, Varanasi, for extending support in carrying out the experiments. The authors would also like to accord their sincere thanks to the anonymous reviewers for their valuable comments and suggestions which improved the quality of the paper drastically.

References

[1] A.Y. Zomaya and M. Wright, "Observation on Using Genetic Algorithms for Channel Allocation in Mobile Computing," *IEEE Trans. Parallel and Distributed Systems,* vol 13, no. 9, pp. 948-962, Sept. 2002.

[2] I.E. Kassotakis, M.E. Markaki, and A.V. Vasilakos, "A Hybrid Genetic Approach for Channel Reuse in Multiple Access Telecommunication Networks," *IEEE J. Selected Areas in Comm.,* vol. 18, no. 2, pp. 234-243, Feb. 2000.

[3] M. Asvial, B.G. Evans, and R. Tafazolli, "Evolutionary Genetic DCA for Resourse Management in Mobile Satellite Systems," *IEE Electronics Letters,* vol. 38, no. 20, pp. 1213-1214, Sept. 2002.

[4] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1999.

[5] M. Mitchell, *An Introduction to Genetic Algorithms.* London: MIT Press, 1999.

[6] 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.

[7] 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.

[8] G. Calhoun, *Digital Cellular Radio.* Artech House, 1988.

[9] J. Schiller, *Mobile Communications.* Pearson Education, 2003.

[10] G.H. Forman and J. Zahorjan, "The Challenges of Mobile Computing," *IEEE Trans. Computers,* vol. 43, no. 4, pp. 38-46, Apr. 1994.

[11] S.H. Wong and I. Wassell, "Dynamic Channel Allocation Using a Genetic Algorithm for a TDD Broadband Fixed Wireless Access Network," Laboratory for Comm. Eng., Univ. of Cambridge, U.K. 2002.

[12] N.F. Huang and H.I. Liu, "A Study of Isochronous Channel Reuse in DQDB Metropolitan Area Networks," *IEEE/ACM Trans. Networking,* vol. 6, no. 4, pp. 475-484, 1998.

[13] A.K. Tripathi, D.P. Vidyarthi, and A.N. Mantri, "A Genetic Task Allocation Algorithm for Distributed Computing System Incorporating Problem Specific Knowledge," *Int'l J. High Speed Computing,* vol. 8, no. 4, pp. 363-370, 1996.

[14] J.J. Grefenstelle, "Incorporating Problem Specific Knowledge into Genetic Algorithm," *Genetic Algorithm and Simulated Annealing,* Morgan Kaufman, 1987.

Somnath Sinha Maha Patra received the MSc degree in computer science from Banaras Hindu University, Varanasi, India, in 2004. Currently, he is working as an Engineer R&D (Network Management System) for UTStarcom Inc., Gurgaon, India. His research interests include mobile computing and network management.

Koushik Roy received the MSc degree in computer science from Banaras Hindu University, Varanasi, India, in 2004. His research interests include mobile communication and the Genetic Algorithm.

Sarthak Banerjee received the MSc degree in computer science from Banaras Hindu University, Varanasi, India, in 2004. He is working as a software engineer at Newgen Software Technologies Ltd., New Delhi. His research interest is in mobile communication.

Deo Prakash Vidyarthi received the master's degree in computer application from the MMM Engineering College, Gorakhpur, India, in 1991 and the PhD degree in computer science from Jabalpur University (with work done at Banaras Hindu University (BHU), Varanasi, India) in 2002. He has taught UG and PG students of the Computer Science Department of BHU, Varanasi, India, for almost 13 years. Currently, he is working as an associate professor in the School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi. His research interests include parallel and distributed systems, mobile computing, and evolutionary computation.