

Genetic Algorithm to Solve Optimum TDMA Transmission Schedule in Broadcast Packet Radio Networks

Goutam Chakraborty

Abstract— The problem of finding optimum conflict free transmission schedule for a broadcast packet radio network (PRNET) is NP-complete. In addition to a host of heuristic algorithms, recently neural network and simulated annealing approaches, to solve this problem, were reported. We showed that standard genetic algorithm, though able to solve small problems, performed poorly for large networks. This is because, classical crossover and mutation operations create invalid members, which flood the whole population, hindering the progress of the search for valid solutions. In this work, special cross-over and mutation operations are defined, such that the members of the population always remain valid solutions of the problem. Excellent results were obtained in a few generations, even for very large networks with 400 nodes. The results were compared with recently reported neural network and mean field annealing approaches.

I. INTRODUCTION

Packet Radio networks (PRNET) are widely used for wireless communication over a wide geographical area, where direct radio or cable connection is impractical. A host of papers, published during '84 to '87, contain nice survey about this problem and its variations [1], [2], [3], [4], [5], [6], [7].

The available radio frequency band can be allocated to the different nodes in the network in different ways: frequency division [6], [8], time division [9], code division, and spatial reuse. When a single radio channel is used, the communication can be facilitated either by broadcasting, or by activating a subset of network links in proper sequence [10], [11], [12]. When nodes transmit packets in broadcast mode using omnidirectional antennas, network management is simple if all nodes are tuned to the same channel frequency, and use time division and spatial reuse [13], [14], [15]. Spatial reuse means that the same channel is used simultaneously at different non-interfering parts of the network. Sometimes directional antenna or low transmission power is used to facilitate spatial reuse of channel [16], [17]. Since all nodes can not directly communicate, nodes act as store-and-forward repeaters facilitating multi-hop connection. The packet radio network (PRNET) model we assume here uses time division multiplex with spatial reuse.

A packet radio network can be modeled by an undirected graph, where the nodes represent the transreceiver stations. A link between two nodes is present when they can trans-

mit and receive packets directly. An example of five nodes network is shown in Fig. 1(a). Here, node-1 can communicate directly with node-2 and node-3, but not with node-4 or node-5. Node-1 can transmit a packet to node-5 say, using node-3 and node-4 as intermediate repeaters.

In the PR network, each station can transmit or receive, which is controlled by its control unit. When a node transmits, all its neighbors connected by direct link in the graph, can receive. The neighboring node/s could absorb the packet, if it is so designated. Else, it may store to transmit it later, in which case it acts as an intermediate repeater.

We consider a fixed topology PRNET, where a single wide band Radio channel is used by all nodes. A time division multiple access (TDMA) protocol is used [5]. The transmissions of packets are controlled by a single clock. The time is divided into distinct frames consisting of a fixed number of time-slots. A time-slot equals to the total transmission time required for a single packet to be transmitted and received by a pair of neighboring nodes. Many nodes may transmit simultaneously at the same time-slot without conflict, if they are far apart, i.e., there is no interference. In one TDMA frame, all the nodes must be able to transmit at least once. This is termed as *no-transmission* constraint.

The basic optimization objective is to get the smallest length TDMA frame, where many nodes are allowed to transmit simultaneously in a single time-slot in a conflict free manner. The secondary objective is to maximize the number of such transmissions for maximum utilization of the channel.

In addition to *no-transmission* constraint, there are other constraints, namely the *primary conflict* and the *secondary conflict*. *Primary conflict* says that a particular node can not transmit and receive in the same time-slot. In other words, two connected nodes can not transmit simultaneously. A *secondary conflict* occurs when two or more packets arrive at a node in a single time-slot. This will occur when two nodes at a distance of two hops are allowed to transmit simultaneously. Then, the intermediate node will receive two different packets from two directly connected nodes, at the same time slot. The transmission schedule in the TDMA cycle should avoid such *primary* and *secondary conflicts*.

Once the optimum transmission pattern for the TDMA frame is decided, the same frame is repeated over time. Thus, though the transmission is broadcast in nature, it is

The author is with the department of Software and Information Science, Iwate Prefectural University, Iwate ken, Takizawa Mura, Japan - 020-0193. E-mail: goutam@soft.iwate-pu.ac.jp .

centrally scheduled and collision free. We do not need any overhead like contention technique, as in ALOHA [18].

Fig. 1(d) is a valid TDMA frame for the 5-node network in Fig. 1(a). It satisfies *no-transmission* constraint, and *primary* and *secondary* conflicts. But it is not optimal. A solution with shorter frame length is shown in Fig. 1(e).

TDMA scheduling problem is proved to be NP-complete [19]. Several heuristics and other algorithms were proposed during last two decades. A trivial TDMA solution, which satisfies all the constraints but not optimized, is of frame length equal to the number of nodes. There, a single transmission for each of the, say N , nodes is allotted at N different time-slots. A formal introduction of this scheduling problem and the terms used in the paper are explained in section II. In section IV, we described how, starting from trivial solutions, optimal TDMA frames can be obtained using standard genetic algorithm. Before that, in section III, we give a brief introduction to genetic algorithm [20], [21], [22]. The results obtained by standard genetic algorithm were good for small networks only, when the number of nodes were 40 or less [23]. But it failed for larger nets. We proposed special crossover operator, suitable for this problem, which is explained in section V. This proposed modified genetic algorithm could deliver excellent results even for large nets. The details of simulation and results, discussions about setting of parameter values for genetic operators are in section VI. In section VII, we further proposed a randomized algorithm to generate solutions which are not trivial, but already optimized in frame length. In this case the genetic algorithm could improve the channel utilization index by a factor of 15 to 50%. Thus, the combination of the randomized algorithm with the proposed genetic algorithm delivers the best results. Comparison with works using neural network [24] and simulated annealing [25], are in section VII.C. Section VIII is the conclusion.

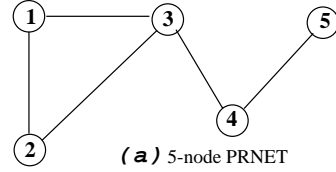
II. THE PROBLEM

PR network can be represented by a graph $G = (V, E)$, where $V = \{v_1, v_2, \dots, v_i, \dots, v_N\}$ is the set of nodes and $E = \{e_1, e_2, \dots, e_L\}$ is the set of undirected edges. The existence of an edge between two nodes means that both can directly receive packets transmitted from the other. The neighboring information, i.e., the connectivity among network nodes, are described by a $N \times N$ symmetric connectivity matrix \mathbf{C} , where the element

$$c_{ij} = \begin{cases} 1, & \text{if } v_i \text{ and } v_j \text{ are connected} \\ 0, & \text{otherwise} \end{cases}$$

The connectivity matrix for the 5-node network in Fig. 1(a) is shown in Fig. 1(b). When two nodes are directly connected, we say that they are one *hop* apart. The problem is to schedule constraints satisfied TDMA frame. The transmissions will follow the same TDMA frame repeated over time. We denote a TDMA frame by a $M \times N$ matrix \mathbf{T} , where its element

$$t_{mj} = \begin{cases} 1, & \text{if } v_j \text{ transmits in time-slot } m \\ 0, & \text{otherwise} \end{cases}$$



$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

(b) Connectivity matrix (c) Compatibility matrix

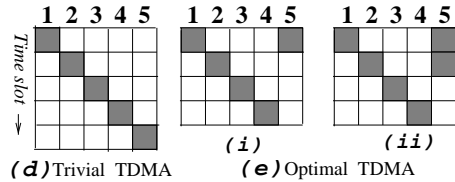


Fig. 1. Example of a (a) 5-node network, corresponding (b) \mathbf{C} matrix and (c) \mathbf{D} matrix, with (d) trivial TDMA schedule and (e) optimal TDMA schedule

When node v_i transmits a packet, none of its neighbors, i.e., nodes which are one *hop* away, are allowed to transmit simultaneously, as this would give rise to *primary conflict*. All nodes two *hops* away from v_i should also be disabled to transmit simultaneously with v_i , as this would result in *secondary conflict* of multiple reception at intermediate nodes. All these nodes which are one *hop* and two *hops* away from v_i form, what is called the *broadcasting zone* [19] of v_i . The set of these nodes we denote by B_i . It is clear that non-interference requires that none of the nodes in B_i should be allowed to transmit simultaneously with v_i . From this concept we can generate a $N \times N$ compatibility matrix \mathbf{D} , where its element

$$d_{ij} = \begin{cases} 1, & \text{if } v_j \in B_i \\ 0, & \text{otherwise} \end{cases}$$

The \mathbf{D} matrix for 5-node PRNET is shown in Fig. 1(c). It should be noted that even after removal of primary and secondary conflicts, a node may experience interference. But to obtain any meaningful algorithm, one has to assume the absence of such interference.

The problem is to find shortest TDMA cycle, such that the following constraints are satisfied.

- Every stations should be scheduled to transmit at least once i.e. *no-transmission* constraint is satisfied.

$$\sum_{m=1}^M t_{mi} \geq 1 \quad \forall i \quad (1)$$

- A station can not transmit and receive packets in the same time-slot to avoid *primary conflicts*.
- A station can not receive two or more transmissions simultaneously i.e., *secondary conflicts* are to be avoided.

Formally,

$$\text{if } t_{mi} = 1, \text{ then } \sum_{j=1}^N t_{mj} d_{ij} = 0 \quad \forall m, i$$

i.e.,

$$\sum_{m=1}^M \sum_{i=1}^N \sum_{j=1}^N t_{mi} t_{mj} d_{ij} = 0 \quad (2)$$

The last two conflicts are avoided if the TDMA frame \mathbf{T} is scheduled satisfying Eq. (2). A trivial solution satisfying all the three constraints is a N -slot TDMA frame, where N different stations transmit in N different time-slots, as shown in Fig. 1(d).

The primary optimization criteria is to minimize the length of TDMA cycle, which means M should be as small as possible. A secondary optimization objective is to maximize the total number of transmissions. By allowing more nodes to transmit in a time-slot without interference, the channel utilization of the network is increased. In case of CSMA (carrier sense multiple access) with nodes transmitting packets of arbitrary length, there are collisions between packets due to hidden terminal problem [6]. In such a broadcast multihop PRNET, evaluation of throughput is difficult [26]. In the case of centralized TDMA schedule, as considered here, all transmissions are successful. One optimization criterion of the schedule is channel utilization index ρ , where

$$\rho = \frac{1}{M \times N} \sum_{m=1}^M \sum_{j=1}^N t_{mj} \quad (3)$$

is to be maximized. In section VI and VII, we will show how ρ is improved by genetic algorithm.

It is trivial that if the maximum degree of a node in the net is G , the tight lower bound for M would be

$$M \geq (G + 1) \quad (4)$$

As the problem is NP-complete, and there is no algorithm to find the optimum solution, we will use this tight lower bound ($G + 1$) as an empirical measure to judge the quality of solution. When $M = G + 1$, we know that the solution is optimal.

For the 5-node PRNET, the trivial schedule is shown in Fig. 1(d), satisfies all the constraints but does not optimize any of the optimization criteria. For this network $G = 3$. In Fig. 1(e)(i) the length of the TDMA frame is minimized to $G + 1$. In Fig. 1(e)(ii), ρ is maximized too.

A. Previous works

When the optimization criterion is only minimizing the length of the TDMA frame, and only *primary conflicts* are considered, the scheduling problem translates to simple graph-coloring problem. To include secondary conflicts, one need to consider the compatibility matrix \mathbf{D} as the connectivity matrix, instead of \mathbf{C} , in the graph coloring problem. As the graph coloring problem is NP-complete, so is

this TDMA scheduling. Formal proof of NP-completeness is available in [19], [25].

During the last two decades, several algorithms were proposed to solve this problem. Instead of broadcast scheduling, some considered a similar problem of optimal schedule of activating different links [10], [12]. For broadcast scheduling, the different algorithms could be classified depending on their objectives and approaches. Most of the earlier works were either centralized [9], [14], [27], [28], or distributed heuristic algorithms [19], [29]. Their optimization objective was to maximize transmission [19]. Those algorithms started with the trivial initial schedule, as shown in Fig. 1(d), and added transmissions to it to the maximum possibility without violating constraints. The length of the TDMA frame remained same, equal to the number of nodes, and therefore quite long for networks with many nodes.

In recent years, random algorithms using neural network [12], [24], [30], [31], and simulated annealing [25] were proposed. In those works, the main optimization objective was to minimize the length of the TDMA frame. These algorithms too can not reduce M from the initial setting. As there is no clue of what would be the optimum length, they usually started with $M = G + 1$, the tight lower bound in Eq. 4. When no solution could be found, M is increased in steps of 1. Every time the algorithm has to run from the beginning trying to find a valid solution. Depending on the problem complexity, many trials may be necessary. As simulated annealing and artificial neural network, implemented in conventional computers, are computationally heavy, these approaches could be quite slow.

III. INTRODUCTION TO GENETIC ALGORITHM

Genetic Algorithm (GA) is a search algorithm based on the mechanics of natural selection [32]. Compared to other approaches, they are superior, because of wide applicability. They make few assumptions from the problem domain, and are not biased towards local minimums. At the same time, GAs are very efficient to direct the search towards relatively prospective regions of the search space.

The first step in GA is to encode the solution of the problem in binary bit string. The solution in its original form is referred to as *phenotype*, whereas its binary encoded version is called *genotype* or simply *chromosome*. It is best to have a one-to-one mapping between the solution of the problem and the chromosome representation. As the TDMA schedule itself is in 0s and 1s, phenotype and genotype are same.

Next a pool of solutions of the problem, called initial population, is created. In general, these solutions are generated simply randomly, without any consideration to how good they are. A fitness function has to be defined to measure the goodness of these encoded solutions. Genetic operators *selection*, *crossover*, and *mutation* operate on the population to generate new population, i.e., new set of solutions, from the old ones. Good solutions are *selected* with greater probability to the next generation, in line with the idea of *survival of the fittest*. Standard *Crossover* opera-

tion recombines arbitrarily selected solutions pairwise, by interchanging portions of them, producing divergent solutions to explore the search space. An occasional *mutation* operation is performed on a chromosome by flipping a bit at random position of the encoded chromosome, to facilitate jumping of solutions to new unexplored regions of the search space. As the algorithm continues and newer generations evolve, the quality of solutions improve. The success of genetic algorithm is explained by schema theorem and building block hypothesis in [21].

Many strategies for fitness calculation, selection, crossover and mutation are proposed. The basic steps for the Standard Genetic Algorithm (SGA) are shown below. First the notations are explained.

g : generation number.

G : maximum generation.

P : population size.

$\Phi(g)$: set of chromosomes at generation g .

$\Phi''(g)$: set of chromosomes after selection.

$\Phi'(g)$: set of chromosomes after crossover.

Algorithm **SGA** ($g, G, \Phi(g), P$)

```

01 begin
02    $g = 0$ ;
03   Create  $P$  members of the initial population  $\Phi(0)$ ;
04   Calculate fitnesses of the members of  $\Phi(0)$ ;
05   while ( $g \leq G$ )
06      $g := g + 1$ ;
07      $\Phi''(g) \xleftarrow{\text{selection}} \Phi(g - 1)$ ;
08      $\Phi'(g) \xleftarrow{\text{crossover}} \Phi''(g)$ ;
09      $\Phi(g) \xleftarrow{\text{mutation}} \Phi'(g)$ ;
10   endwhile
11 end

```

Fig. 2. Algorithm for Standard Genetic Algorithm

The success of genetic search depends on balancing the two aspects of (1) population diversity for exploring the different regions of the search space, and (2) selection pressure to get to the optimum point fast. If the best few members of the initial population predominate the whole population in a few early generations, due to their much better fitnesses and high selection pressure, it would result in poor exploration and premature convergence to suboptimal solution. On the other hand, at later stage of the search, when high performance regions are identified, disruption of good chromosomes after crossover with bad ones would slow down the process of reaching the global optimum. A number of strategies were proposed [22](chapter 4 & 6), [33], [34], [35] to overcome this problem by setting a balance between diversity of solutions in the beginning and selection pressure to concentrate on the best few during later generations.

For constrained optimization problems, all encoded strings may not satisfy the different constraints. There are two possible approaches to overcome this problem:

1. Use the standard genetic algorithm and allow all encoded solutions, both valid and invalid. Invalid solutions

are penalized so that they may not survive (get selected) to the next generations. Valid solutions are assigned fitnesses according to how good they are, with respect to the optimization criteria.

2. The chromosome representation, the crossover, and the mutation operations are defined such that invalid solutions are always avoided. This usually involves some extra computation during crossover and mutation. Higher fitnesses are assigned to chromosomes representing better solutions.

The above approach (1) is easier to implement and softer, which means problem independent. But, in practice they can produce optimum or near optimum results only for small problems. When the size of the problem and/or the accuracy of the solutions are increased, due to explosive increase in the size of the search space mostly crowded by invalid solutions, finding optimum or near optimum valid solution is almost impossible. The other problem is, how to decide to what extent the invalid solutions are to be penalized. Between two invalid solutions, to what extent they are violating the constraint is difficult to judge? The fitness calculation then heavily affects the efficiency of the algorithm, and the quality of the solution.

Approach (2) is harder, i.e., problem specific, and difficult to implement. One has to define the solution representation and/or genetic operators, so that the chromosomes always represent valid solutions. The algorithm also gets strongly associated with the particular problem. As the valid and invalid regions are interlaced, it is difficult to define genetic operations that make the offsprings always valid. Correcting invalid chromosomes to their nearby valid ones is computationally costly. Yet, because the search space is now restricted to valid solutions only, the algorithm is more efficient, and the quality of solutions much better.

In the following sections, we will show how two different approaches were implemented to solve TDMA scheduling problem, and their respective performances.

IV. STANDARD GENETIC ALGORITHM APPROACH

We first experimented with standard Genetic algorithm to solve the TDMA scheduling problem. It worked successfully for small networks, but failed when the network size is increased. In this section, we describe the standard GA approach. Corresponding experimental results are in section VI.B.

A. Coding the problem and the initial population

TDMA cycle is in binary. Therefore, TDMA frame itself could be the chromosome and no coding is necessary. For a N -node PR network, a chromosome could be a $N \times N$ binary matrix. A random initialization of the chromosome by 0s and 1s will not be a valid solution of the problem. A trivial valid solution is a TDMA cycle with N time-slots, where transmissions for different nodes are allocated in different time-slots. Then there will be no interference. This solution is far from optimal though. A number of such trivial solutions could be generated by randomly selecting different time slots for different nodes. This is done by

different random permutation of 1 to N . In our implementation, population size P remains same throughout all the generations.

B. Fitness evaluation and selection

While selecting member solutions to be carried to the next generation, we need to find proper evaluation function to judge the fitness values of different competing solutions. For a valid solution, its quality is judged by the length of the TDMA cycle, i.e. M , and the channel utilization index ρ . After crossover operation, it is possible that the offsprings created would violate the constraints and form invalid solutions. It is difficult to properly evaluate invalid individuals. We penalize such infeasible solutions by using a penalty function, $Pen(\mathbf{T})$, where \mathbf{T} denotes the invalid chromosome. With a heuristic penalty function $Pen(\mathbf{T})$, we can not actually evaluate the degree of invalidity. We can only apply subjective judgement to decide which one is better between the two, and rank them according to their goodness. We rank all the chromosomes in the population by comparing every pairs using the following algorithm.

Step 1: First we check if the two chromosomes are valid solutions or not.

Step 2: If both are valid solutions:
The chromosome with lesser number of time-slots (M) will have lower rank. For two chromosomes with equal number of time-slots, the one with higher channel utilization index (ρ) will have lower rank.

Step 3: If only one is a valid solution:
The valid solution will have lower rank.

Step 4: If both are invalid solutions:
Calculate a quadratic penalty function $Pen(\mathbf{T})$ for the chromosomes as follows.

$$Pen(\mathbf{T}) = (P1 \times (ntr + cf / (P2 \times N)))^2 \quad (5)$$

where,

ntr = the number of nodes failed to transmit
 cf = number of *primary* and *secondary* conflicts
 N = number of network nodes
 $P1, P2$ = parameters to tune $Pen(\mathbf{T})$
 Lower $Pen(\mathbf{T})$ value means lower rank.

All the chromosomes are serially arranged according to their ranks. The best solution is assigned rank 1, and the worst ranked P , the population size.

We use *tournament selection* [20](Chapter 24) for selecting chromosomes to the next generation. Here, some predefined number, say τ the tournament size, of chromosomes are randomly chosen from the whole population. The best among them goes to the next generation. This process is repeated P number of times to select P members of the next generation. Tournament selection is very efficient with time complexity $\mathcal{O}(P)$. Selection pressure is low when τ is small, and increases with τ . And therefore, it is very easy to control selection pressure by changing tournament size.

C. Crossover and Mutation

As time-slots are the time units of transmission, crossover operations are implemented on the rows of a

TDMA frame, not on the whole TDMA cycle. The total number of time-slots in the whole population is $P \times N$ initially. We select rows from different chromosomes (TDMA frames) using predetermined crossover probability, and gather them in the mating pool. For crossover between two strings selected randomly from the pool, a cut position is determined randomly and the classical crossover operation is done by swapping parts of the chromosome from the cut point. If, in a valid offspring, all entries for a row becomes 0, that row is erased from that chromosome. During crossover, violation of constraints are not examined. If an individual becomes an invalid solution, its fitness is penalized as discussed in the section IV.B.

Simple mutation is used where bit positions are selected for mutation with mutation probability. The binary bit at the selected position is flipped.

Simulation results, reported in section VI.B, show that the standard GA failed to solve complex problem with large number of nodes. In the next section V, we propose special crossover operation suitable for this problem, so that members of the population are always valid.

V. MODIFIED GENETIC ALGORITHM

We defined a new crossover operator for this problem, so that invalid solutions are not created. In section IV, constraints of the problem were used only to penalize the offsprings, when they were invalid. The new crossover operator is defined using knowledge of constraints of the problem. Obviously, the crossover operation is more involved, but the fitness calculation is easy and more meaningful as all the offsprings are always valid.

A. Coding the problem and the initial population

Coding of the TDMA schedule and creation of the initial population is done the same way as described in section IV.A. When we use trivial TDMA cycles of length N as members of the initial population, the proposed genetic algorithm could reduce the cycle length to a great extent, but could not reach the optimum length for large problems. A very high value of channel utilization index is achieved though. Simulation results are reported in section VI.C.

In another set of experiments, we used initial population generated by a randomized algorithm described in section VII.A. Though, those schedules are already short in cycle length, the ρ values are low, as there is only one transmission per node in the whole TDMA frame. On a population generated by the randomized algorithm, the proposed GA is executed to improve transmission efficiency. Corresponding simulation results are reported in section VII.B and VII.C.

B. Fitness evaluation and selection

In this case, the member solutions are always valid. The comparison of fitness among solutions is therefore easier. For any chromosome \mathbf{T}^i , the TDMA frame length M^i and channel utilization index ρ^i are used to evaluate it. The chromosome with shorter M^i is a better solution and will have lower rank. For two chromosomes with equal number

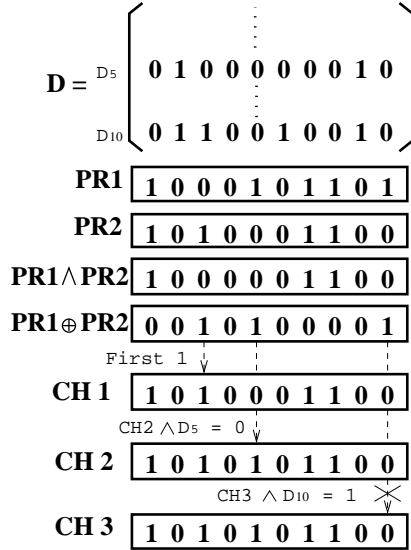


Fig. 3. An example of crossover operation, for a 10-node network

of time-slots, the one with larger value of ρ^i , i.e., with more transmissions, will have lower rank.

Here too we use tournament selection. Constant tournament size is used throughout all the generations.

C. Crossover and Mutation

Crossover is done on rows of TDMA cycle. Rows from members of the population, i.e., different TDMA frames, are selected using the predetermined crossover probability, and are marked to be members of the mating pool. A pair of rows are randomly selected from the pool for crossover. Let's name these two parents as $PR1$ and $PR2$. The objective of the crossover is to create an off-spring with a better schedule for a time-slot with more transmissions, by combining the parent schedules. By crossover operation only one child is created, which may or may not replace the parent/s, depending on how good it is. Maximum transmission schedule, combining the parents $PR1$ and $PR2$, is possible when $PR1$ and $PR2$ are logically *OR*ed to create the off-spring. But, this child may violate conflicts described by matrix D . The following algorithm creates a valid child (CH), which is a maximal of the transmissions from the two parents, and does not violate constraints.

Step 1 First $PR1$ is logically *AND*ed with $PR2$ and written on CH , i.e., $CH \leftarrow PR1 \wedge PR2$, where \wedge denotes logical AND operation. It is trivial that CH does not violate any constraint, as $PR1$ and $PR2$ do not.

Step 2 $PR1$ is logically exclusive-ored with $PR2$. This $PR1 \oplus PR2$ string is used for possible addition of 1s to CH string.

Step 3 The $PR1 \oplus PR2$ string is scanned from left to right. The first 1 encountered is copied to the same position of CH string replacing 0. This is the new CH , shown as $CH1$ in Fig. 3. This addition of 1 would not violate any constraints, as CH would still be a subset of either $PR1$ or $PR2$, both of which are valid.

Step 4 We go further right on the $PR1 \oplus PR2$ string till

we encounter the next 1, say at i^{th} bit position. This 1 is temporarily copied to CH string, and logically *AND*ed with the i^{th} row of D matrix. If the resulting string after *AND* operation is all 0s, it is easy to see that there will be no *primary* or *secondary conflict* by replacing 0 with 1 at this i^{th} position of the CH string. This 1 then replaces 0 in CH , which becomes the current CH (shown as $CH2$ in Fig. 3). If the *AND*ing results in a non-zero string, this 1 at i^{th} position would violate *primary* or *secondary conflict*, and the 0 is kept unchanged. This is what happened at the rightmost bit in Fig. 3, when $CH3 \wedge D_{10}$ has a 1 in bit position 3. The procedure is repeated, till the end of the $PR1 \oplus PR2$ string is reached, when we get the final CH .

Step 5 In strings $PR1$, $PR2$ and CH , a 1 denotes transmission allowed for a particular station. Thus, corresponding to a string, say $PR1$, a set of stations are scheduled to transmit. We denote it by $\{PR1\}$. After step 4 is completed, there are three possible outcomes.

(1) $\{CH\} \supset \{PR1\}$ and $\{CH\} \supset \{PR2\}$.

In this case, as CH is a better schedule compared to both the parents, it replaces $PR1$ and $PR2$ in their original TDMA frames.

(2) $\{CH\}$ is superset of either $\{PR1\}$ or $\{PR2\}$, not both. In that case CH replaces only that parent which is its proper subset. The other parent remains unchanged.

(3) Neither of the above two cases are true. Do nothing.

Step 6 When a parent PR is replaced by the offspring CH , all rows of that TDMA frame, which are subsets of $\{CH\}$ are erased.

Simple mutation is done by flipping a bit. At every bit of all the members of the population, a random number between 0 to 1 is generated. If it is less than or equal to mutation probability, it is flipped, only when it does not violate conflicts. If the mutation creates an invalid frame, it is discarded.

Theorem: In the proposed GA, all chromosomes in every generations will satisfy all the constraints.

Proof: It is trivial that the TDMA frames, after crossover, will not violate *no-transmission constraint*, as the set of transmission of a child is always a superset of that of the parent it replaces. Further, a crossover and mutation operation are ensured not to create any chromosome that would violate *primary* or *secondary conflict* constraints. We have started with an initial population with all the members satisfying all the constraints. Therefore, the solutions after each generations will also satisfy them.

VI. SIMULATION AND RESULTS

The performance of the standard as well as the modified genetic algorithm were experimented for a large number of times, using connected networks of different sizes and degrees of connectivities. In section VI.B, we give results of experiments using standard GA. The results show that when the network size is over 40-nodes, the results are very poor. In section VI.C, we describe the results of experiments done with the modified crossover. Here, the simulations were done on large connected planar networks described in section VI.A. In section VI.B and VI.C, we used

randomly generated trivial TDMA frames, with $M = N$, as the initial population. We also did extensive experiments on how the efficiency and the quality of the results depend on the population size, tournament size, crossover, and mutation probabilities.

A. Problem Set up

As there is no standard benchmark problem set, we will perform simulations on networks similar to what were used in recently reported publications [19], [24], [25], and [29]. In practice, for PRNET, the degree of connectivity of nodes are usually low. Here, we will set up networks which are described by planar graphs. Only neighboring nodes are directly connected. Nodes in large networks are placed in the form of matrices, as was done in [24], [25].

To create large problems, a matrix of dots in X-Y plane. are arranged. Now, a node is randomly connected to a subset of its neighboring eight nodes (for nodes on the boundary it is less than 8). It is ensured that finally it is a connected network, so that any node can communicate with any other node, if sufficient number of hops are used. The average degree of connection is set to different values. Two such 100 nodes networks, with average degree of connections 4 and 6, are shown on the left side of Fig. 10 and Fig. 11 respectively.

B. Results with Traditional Genetic Algorithm

We created networks of randomly connected nodes with average node degree of 3. Networks with number of nodes 10, 20, 30, 40, 50, and 60 were used for the experiments. The probability of connection between two nodes was decreased exponentially with the distance between them. Each experiment was run 50 times and the average value of the results are shown in Table. I. Each time up to 1000 generations were executed, sufficient for the results to be stabilized. Different values of the parameters e.g., probability of crossover, mutation, and tournament size were tried. They were set at values, where the results were best.

The results in Table. I shows that the standard GA works somewhat well only for small networks of size up to 40, till when, though the length of the TDMA frame is not optimized, the channel utilization index reaches quite high value. Beyond that size, the results are little better than the trivial solution. That means, the initial population improved little through generations.

The most important reason for this failure is that, by allowing infeasible solutions the search space explodes. Most of the members in the population are then invalid. The search for good valid solution fails, even after penalizing the invalid ones. Moreover, it is impossible to define a single penalizing function, which could penalize all the invalid solutions according to their degrees of invalidity. Also, by uniform ranking we lose the information about the relative goodness (or badness) of different solutions, and the selection pressure becomes weak. Thus, it is important to keep all the member solutions valid throughout generations and define crossover and mutation accordingly. Experiments in

N	10	20	30	40	50	60
$G + 1$	6	5	6	6	7	7
M	6.0	9.6	21.1	28.0	49.2	59.9
ρ	0.183	0.208	0.208	0.169	0.023	0.017
Avg node degree = 3		Tournament size = 4		$G = \max.$ degree		
M : avg number of time-slots		$\rho =$ Avg channel utilization index		Maximum Generation = 1000		
Population Size = 100		Probability of crossover = 0.3		Prob. of mutation = 0.001		

TABLE I
RESULTS OBTAINED USING STANDARD GA

sections VI.C are with proposed crossover, where all member solutions generated are valid.

C. Results with new genetic operators on trivial initial population

Here we used large planar graphs as described in section VI.A. Smaller nets were same as used in [19], [24], [25], and [29].

In Table. II the results of these experiments are summarized. For all the problems, GA were run for 20 times, each time up to a maximum of 300 generations. Though not exactly same, the results are quite similar for 20 different runs. Short TDMA frames could be reached in a few generations. For large networks, frame lengths are longer than optimum lengths. This is because, even at early generations, lots of transmissions were added in the TDMA frame, and then it is difficult to reduce the frame length further, avoiding primary and secondary conflicts. On the other hand, very high channel utilization index is achieved for all the problems tried. Though the TDMA frame length reaches its stable value in a few generations, the number of transmissions and therefore the channel utilization index continue to improve for many more generations, especially for big problems with 100 or more nodes. To give a rough idea, the computation time for a run of 100 generations is noted in the last column of Table. II. The simulations were done in a DEC VT-alpha 600SNL machine (600Mhz. CPU with 512 Mbyte main memory). As the codes were not optimized, and included lots of reporting commands, the actual execution time for the GA would be much less. For larger networks, due to very limited size of the core memory, swapping caused long execution times.

In Fig. 4, we show typical examples of how the TDMA frame length and channel utilization index improves over generations. The results are for problems #4 and #5.

By changing the fitness function, assigning more priority to reduce the TDMA frame length, or restricting only single transmission per node, it is possible to achieve solutions with shorter frames. Instead, we used a randomized algorithm to obtain short length TDMA solutions, as the initial population. The algorithm is described in section VII.A, and results are reported in section VII.B and VII.C.

C.1 Effect of algorithm parameters

We studied how the efficiency of the algorithm and the quality of the results are affected by the choice of the ge-

Prob. no.	Problem specifications			TDMA frame length in no. of gen.	transmissions in no. of generations	corresponding channel uti. index ρ	Computation time for 100 generations
	No. of nodes	No. of links	av. Degree, and $G + 1$				
1	14	23	3.3, 6	6 in 5	17 in 26	0.202	0.71 Sec.
2	16	22	2.75, 5	5 in 6	17 in 14	0.212	0.85 Sec.
3	40	66	3.3, 8	9 in 13	72 in 47	0.200	5.6 Sec.
4	100	200	4.0, 9	16 in 46	248 in 80	0.155	5.5 mins.
5	100	250	5.0, 9	17 in 50	223 in 90	0.131	5.6 mins.
6	100	300	6.0, 9	18 in 60	206 in 83	0.114	5.8 mins.
7	200	400	4.0, 9	29 in 85	973 in 162	0.168	48 mins.†
8	300	600	4.0, 9	39 in 118	1955 in 247	0.167	2.4 hrs.†
9	400	800	4.0, 9	65 in 164	4628 in 299	0.178†	14.3 hrs.‡

TABLE II

SIMULATION RESULTS WITH TRIVIAL INITIAL SOLUTIONS. A POPULATION SIZE OF 100 WAS USED FOR PROBLEMS #1 TO #3. FOR THE REST OF THE PROBLEMS, THE POPULATION SIZE WAS 400. TOURNAMENT SIZE = 8, PROBABILITY OF CROSSOVER = 0.30, AND PROBABILITY OF MUTATION = 0.001 WERE SET FOR ALL THE EXPERIMENTS. † : THE TRANSMISSIONS WAS STILL IMPROVING, BUT WE STOPPED AFTER 300 GENERATIONS. ‡ : DUE TO SMALL CORE MEMORY, AND EXCESSIVE SWAPPING, THESE COMPUTATION TIMES WERE LONG AND CARRY LITTLE MEANING.

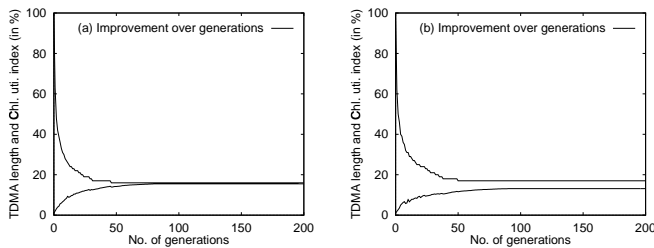


Fig. 4. Improvement of TDMA length and channel utilization over generations for 100-node network problems #4 and #5. The upper curve is for TDMA frame length and the lower one for channel utilization index (ρ).

netic algorithm parameters: probability of crossover (p_c), probability of mutation (p_m), population size (P), and tournament size (τ). It is known that exponentially decreasing mutation rate [36], or adaptively modifying the crossover rate [34] are more useful for genetic search. But, to make the implementation simple, we kept their values constant throughout all the generations.

The experiments were done with 100-nodes networks of different degree of connectivities, which are problems #4 to #6. While finding the effect of one parameter, its value is changed in steps, keeping other parameters fixed. For a particular setting of parameters, the algorithm is run 50 times and the average of these 50 runs are noted. Each run involves number of generations well beyond the convergence of search results. In all the experiments, the trend of the effect of parameters is independent of the network with which the experiment is done. The generic trends are reported as an useful guide for deciding the parameter values.

Probability of crossover (p_c): The commonly used p_c values vary over a wide range from 0.1 to 0.9. The proposed greedy crossover improves the transmission of the schedule quickly, and thereby hinders the search for short

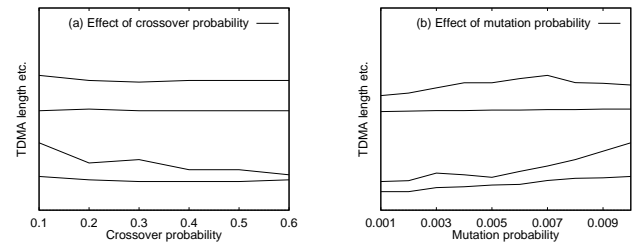


Fig. 5. Variation of performance with p_c and p_m . Top two curves are about the quality of the results. Topmost curve is for TDMA frame length. Second from top is channel utilization index. Bottom two curves are about the efficiency of the algorithm. Third from top is the number of generations required for the TDMA frame length to converge. The bottom curve is the number of generations required for the channel utilization index to converge.

length TDMA frames. Thus a high value of crossover is detrimental to find a short length TDMA frame. We varied p_c from 0.1 to 0.6 in steps of 0.1. The average result of 50 runs, for the TDMA frame length, channel utilization index, number of generations required to converge TDMA cycle length, and number of generations required to converge channel utilization index, for each setting of p_c value, were recorded. The results are shown in Fig. 5(a). The average results are normalized and then scaled, to put all the results in a single graph. It shows that the TDMA length reaches its minimum when $p_c = 0.3$. The channel utilization index value is almost unaffected for $p_c > 0.2$. The number of generations required for convergence is large for $p_c = 0.1$, but is almost same for higher values. The behavior is very similar for other networks too. Thus, we set the value of $p_c = 0.3$ for the rest of the experiments.

Probability of mutation (p_m): p_m values typically should be very low, as otherwise genetic search will behave like random search only. We varied p_m values from 0.001 to 0.01, in steps of 0.001. The average result of 50 runs, for the TDMA frame length, channel utilization index, number of

generations required to converge TDMA cycle length, and number of generations required to converge channel utilization index, for each setting of p_m value, were recorded. The results are shown in Fig. 5(b). The TDMA length reaches its minimum when p_m is lowest i.e., 0.001. The channel utilization index value is almost unaffected. The number of generations required for convergence, for both TDMA length as well as channel utilization index, increases with increase in p_m value. Considering all aspects, we set the value of p_m equal to 0.001 for all experiments.

Population size (P): The rule of thumb is to use as large a population size as system limitations (memory) and time constraints allow. All performance indexes of genetic search improves with increase in population size. But the improvement may be insignificant beyond certain size. We experimented with population size from 50 to 500, in steps of 50. The results are shown in Fig. 6(a). With increase in population (1) the TDMA length slightly improved, (2) the channel utilization index remained almost unchanged, number of generations required for (3) the TDMA frame length to converge was almost unaffected, and that (4) for channel utilization index was a little faster. Little improvements were observed beyond a population size of 400. So, for all the experiments we set $P = 400$. The biggest problem with large population is memory. To keep the previous and the present generation members for processing, the amount of memory required is $2 \times (P + 1) \times N(N + 1)$. For a 400-node network, and 500-member population, the required memory, just to hold these two population, is more than 160 Mbyte, considering 0s and 1s are saved in single bytes of memory. Thus, for large N , the population size has to be limited depending on the available memory.

Tournament size (τ): Tournament size $\tau = 1$ corresponds to no selection, which means that members are randomly picked up and copied to the next generation. With increasing τ , the selection pressure increases. For most applications the value of τ between 4 to 10 are recommended [20](chap. 24). Experiments were done with different tournament sizes, 2, 4, 6, 8, 10. The results are shown in Fig. 6(b). The TDMA cycle length was minimum when the tournament size is 8. The channel utilization index remained almost same for all settings of the tournament size. The number of generations required for convergence also reaches low value at $\tau = 8$. So, for all experiments we set tournament size at 8.

VII. NEW GENETIC OPERATIONS ON ELITE INITIAL POPULATION

In this section, first we propose a simple and fast randomized algorithm to find a pool of valid solutions of the problem. Though no optimization criterion is considered while generating these solutions, it is observed that the best in the pool is optimum in TDMA frame length, for all the problems where optimum TDMA length is known. The algorithm ensures only single transmission per node. The initial population is made up of such TDMA frames. As these solutions are already optimized in length, we call it an elite initial population. The modified GA, defined

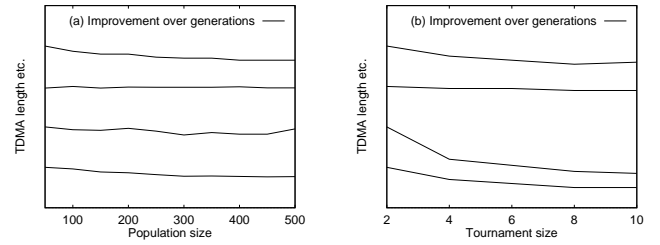


Fig. 6. Variation of performance with population size and tournament size. Top two curves are about the quality of the results. Topmost curve is for TDMA frame length. Second from top is channel utilization index. Bottom two curves are about the efficiency of the algorithm. Third from top is the number of generations required for the TDMA frame length to converge. The bottom curve is the number of generations required for the channel utilization index to converge.

in section V, is then run to improve the transmission efficiency. It is achieved in a few generations. The solutions thus produced are the best. In section VII.A, we explain the randomized algorithm. In section VII.B and VII.C, the results are shown.

A. Randomized Algorithm to Create Elite Initial Population

For a N -node PRNET, we first create P permutations of the digits 1, 2, 3, ..., N , where of course $P < N!$. Suppose, an individual permutation sequence $\Pi^1 = \langle \pi_1^1, \pi_1^2, \dots, \pi_1^N \rangle$. For each different P permutations, the algorithm will create TDMA frames, $\mathbf{T}^1, \mathbf{T}^2, \dots, \mathbf{T}^p, \dots, \mathbf{T}^P$. Thus, \mathbf{T}^1 is created from the permutation Π^1 , \mathbf{T}^2 from Π^2 etc., by a simple fast algorithm explained below.

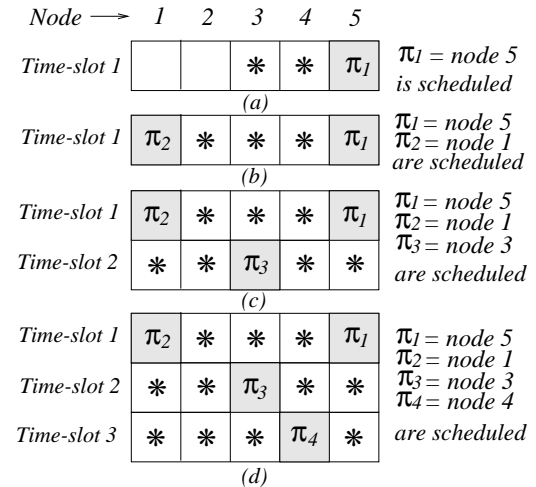


Fig. 7. Illustration of the randomized algorithm to create elite TDMA frame, when the permutation of 5 nodes is 5, 1, 3, 4, 2

The making of \mathbf{T}^i from Π^i is explained using the 5-node example of Fig. 1(a). For $N = 5$, let us take a permutation 5, 1, 3, 4, 2, i.e., $\pi_1 = 5, \pi_2 = 1, \pi_3 = 3, \pi_4 = 4, \pi_5 = 2$. For simplicity, we drop the superscript, as we are considering only one instance. The idea is to allot time-slots for different nodes in this permutation sequence, as shown in

Fig. 7. Here $\pi_1 = 5$. So time-slot 1 is allotted to node-5, i.e., node-5 transmits at time-slot 1. Now if we consult *compatibility matrix* \mathbf{D} , we know that node-3 and node-4 can not transmit simultaneously with node-5. We put a flag * at t_{13} , and t_{14} . The * flag means that the corresponding row (time-slot) can not be assigned to that column i.e., the node. The schedule status at this moment is shown in Fig. 7(a). Next the time-slot allocation is for node π_2 i.e., node-1. From Fig. 7(a) it is evident that node-1 or node-2 can still transmit in time-slot 1 without interference. So we allot time-slot 1 to π_2 (node-1), i.e., t_{11} is assigned for node-1's (π_2 's) transmission. From \mathbf{D} we know that node-2, node-3, and node-4 interfere with node-1. So we need to put * flag at t_{12} , t_{13} , and t_{14} , of which t_{13} and t_{14} are already marked with *. This status of the first time-slot, which is the first row of \mathbf{T} , is shown in Fig. 7(b). Time-slot 1 is now full. Next we have to allot time-slot to π_3 , which is node-3. As time-slot 2 is all free, we allot it to π_3 and change t_{23} to allotted status. Transmission from node-3 interferes with all other nodes. So t_{21} , t_{22} , t_{24} , and t_{25} , all are marked with the flag *. This status is shown in Fig. 7(c). Time-slot 2 is now full. The next in the permutation list is π_4 , which is node-4. As there is no free time-slot available in column 4 (for node-4) in the existing time-slots 1 and 2, we have to add a new time-slot 3 for scheduling transmission from π_4 . We changed t_{34} to allotted status. Matrix \mathbf{D} says that π_4 i.e., node-4 transmission interferes with all other nodes. So, t_{31} , t_{32} , t_{33} , and t_{35} , all are marked with *. This status is shown in Fig. 7(d). We still have to assign a time-slot for π_5 i.e., node-2. As there is no free time-slot available in column 2 in the existing three timeslots, another row (time-slot) is added to the TDMA frame, where node-2 is assigned for transmission. In the schedule it is assigned at t_{42} . Now each node has its transmission scheduled for once, and we have a 4-row TDMA schedule, which is not shown in Fig. 7. This solution is one row shorter than the trivial 5-row schedule. The algorithm is simple and fast. For a 100 nodes network, in a VT-alpha 600MHz machine with 512 Mbyte memory, it takes about 1 second to generate 1000 such solutions. It will take a few milliseconds in a new PentiumIV machine.

Depending on the permutation sequence, the length of the TDMA frames will be different. If P is sufficiently large, many TDMA frames with optimum length would be generated. By selecting required number of minimum length TDMA frames, one could create an elite initial population. The distribution of solutions of different lengths for 100-node network problem #4 is shown in Fig. 8. This result is obtained by generating 1 million TDMA frames using the above mentioned algorithm. We see that a little more than 1% of the solutions are of optimum length 9. So to create an initial population of 50 elite solutions, we need to generate about 5000 random TDMA solutions from 5000 different permutations of 1, 2, 3, . . . 100.

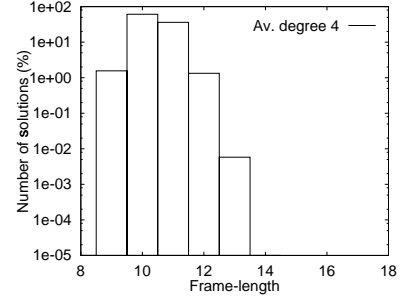


Fig. 8. Distribution of solutions of different TDMA length for 100-node network problem #4

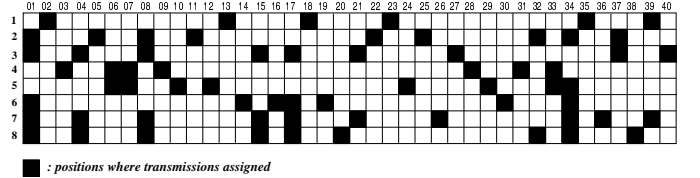


Fig. 9. GA solution for 40-node network problem #3

B. Results with New Genetic Operators on Elite Initial Population

In this last part of experiments, we use the same problem set as in the previous section VI.C. We also kept the members of the population always valid by using the proposed crossover and mutations operations. The initial populations were generated using the randomized algorithm described in section VII.A. The genetic algorithm improved the channel utilization index in a few generations. Though the TDMA frame length for all the members of the initial population are now short, the channel utilization indices are low, $(1/M)$, as there are only one transmission per node. In a few generations, the channel utilization index improved to a very high value, while the transmissions increased by 15-50% of their initial values. In Table. III the results of these experiments are summarized. In Fig. 9 to Fig. 10, final TDMA solutions for problem#3, problem#4 and problem#6 are shown.

If the TDMA frame length is optimized, the schedule is too tight for improving transmissions. With longer TDMA frames, much better transmission efficiency could be achieved. If we compare the channel utilization indices in Table. II and Table. III, it is evident that the channel utilization indices in Table. II are better, at the cost of lengthier schedules.

C. Comparison with recent works

In [25] mean field annealing method was used, for networks with number of nodes up to 40. For the same 40-node network problem, we could improve the TDMA frame length by one time slot, and the channel utilization index by more than 3%. Large networks in our experiments were similar in structure and connectivity, as were used in [24]. Problem #7(100-nodes, 189-links), #8(100-nodes, 282-links), #10(196-nodes, 370-links), and #13(400-nodes,

Prob. no.	Problem specifications		TDMA frame length	Improvement in number of transmissions	Channel utilization index ρ	Required no. of generations	Computation time for 100 generations
	No. of nodes	av. Degree, and $G+1$					
1	14	3.3, 6	6	14→17	0.202	11	0.5 sec.
2	16	2.75, 5	5	16→17	0.212	5	1.1 sec.
3	40	3.3, 8	8	40→65	0.203	62	9.5 sec.
4	100	4.0, 9	9	100→133	0.148	36	4.3 min.
5	100	5.0, 9	10	100→118	0.118	22	4.3 min.
6	100	6.0, 9	11	100→115	0.104	23	4.3 min.
7	200	4.0, 9	9	200→267	0.148	60	17.5 min.‡
8	300	4.0, 9	10	300→453	0.151	81	42.8 min.‡
9	400	4.0, 9	10	400→598	0.149	96	79.7 min.‡

TABLE III

SIMULATION RESULTS WITH ELITE INITIAL POPULATION. A POPULATION SIZE OF 100 WAS USED FOR PROBLEMS #1 TO #3. FOR THE REST OF THE PROBLEMS, THE POPULATION SIZE WAS 400. $p_c = 0.3$, $p_m = 0.001$, AND $\tau = 8$ WAS SET FOR ALL SIMULATIONS. ‡: DUE TO SMALL AVAILABLE CORE MEMORY, AND EXCESSIVE SWAPPING, THESE COMPUTATION TIMES WERE LONG.

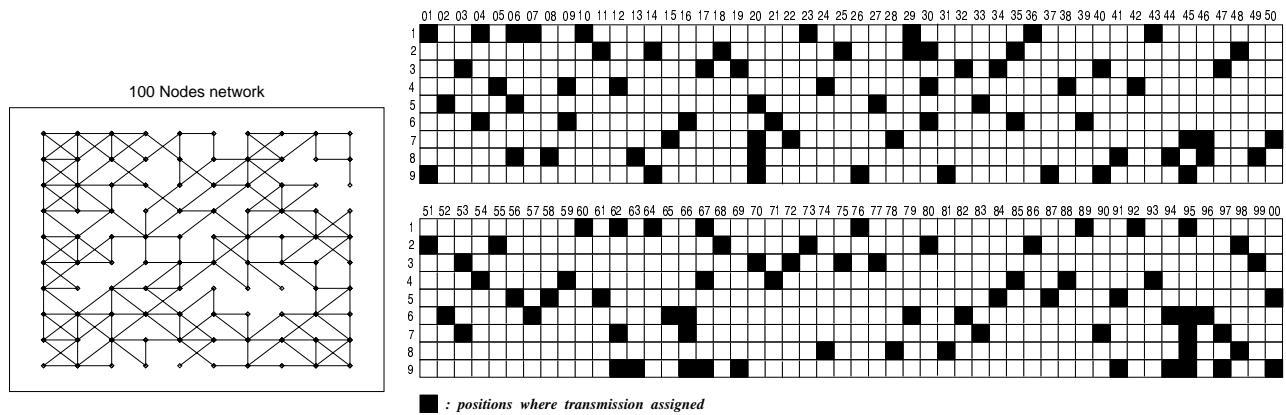


Fig. 10. GA solution for 100-node network problem #4 with degree of connection 4

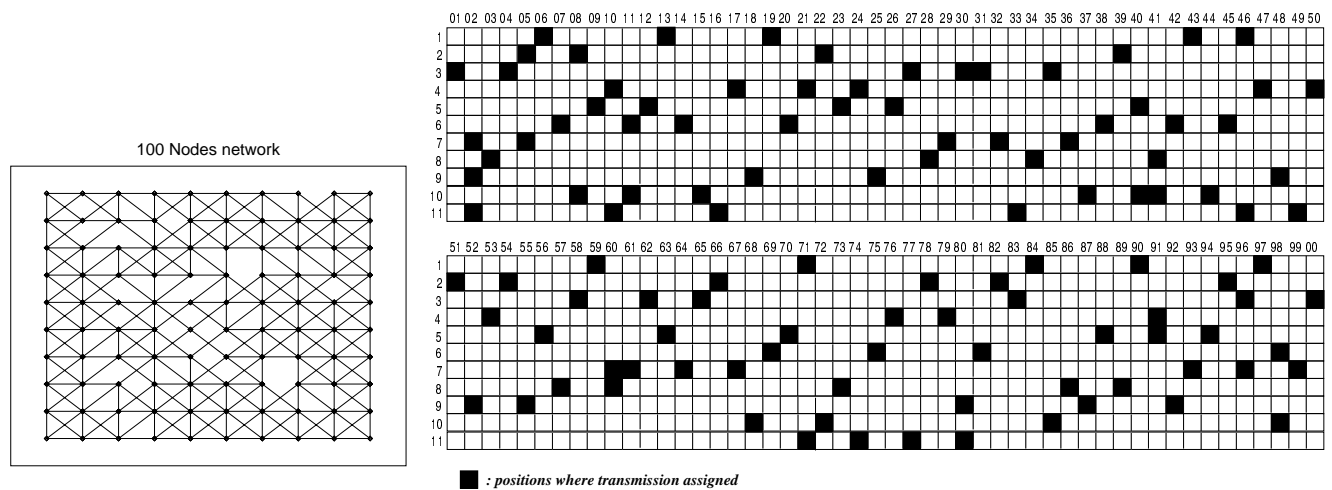


Fig. 11. GA solution for 100-node network problem #6 with degree of connection 6

Our solution				solutions reported in [24],[25]			
N	links, G+1	M	ρ	N	links, G+1	M	ρ
40	66, 8	8	0.203	40	66, 8	9	0.197
100	200, 9	9	0.148	100	198, 9	9	0.111
100	300, 9	11	0.104	100	282, 9	12	0.083
200	400, 9	9	0.148	196	370, 9	10	0.100
400	800, 9	10	0.149	400	805, 9	10	0.100

TABLE IV

COMPARISON OF RESULTS OBTAINED BY THE PROPOSED GENETIC ALGORITHM AND OTHER COMPETITIVE ALGORITHMS. WE PUT PROBLEMS OF COMPARABLE SIZE AND COMPLEXITY IN THE SAME ROW. THE FIRST ROW IS COMPARISON WITH WORK REPORTED IN [25]. THE LOWER 4-ROWS ARE COMPARISON WITH WORK REPORTED IN [24]

805-links) of [24] are similar to our problems #4(100-nodes, 200-links), #6(100-nodes, 300-links), #7(200-nodes, 400-links), and #9(400-nodes, 800-links) respectively. The first three problems are slightly more complex in our set up, as the number of connections are more. For problems #8 and #10 of [24], our TDMA schedule is better by one time slot. The schedules in [24] had only one transmission per node. So we could improve the channel utilization indices by 15% to 50%. Comparison of results on similar networks is summarized in Table. IV. In our previous work [23] using genetic algorithm, we used only small randomly connected networks. We did not use mutation operation, and therefore the improvement of ρ was slow. Generation of elite initial population and subsequent improvement of transmission index using genetic algorithm are also new in this paper.

VIII. CONCLUSIONS

The problem of transmission scheduling for PR network is NP-complete. Randomized algorithms like neural network and simulated annealing, to solve this problem, were recently reported. According to our survey, this is the first time genetic algorithm is used to solve this problem. Standard genetic algorithm with classical crossover operator gave good results for small problems only, and failed for any reasonable size network. We then proposed a new crossover operator, using some problem specific knowledge, to avoid invalid solutions. Excellent results were obtained for big networks (up to 400 nodes were experimented). The capability of the implemented algorithm to handle big networks is restricted only by the size of the available memory.

The algorithm quickly increases transmission, and therefore does not achieve the optimum length TDMA cycle, when trivial initial population is used. Trying variations in crossover operation and fitness evaluation would be an interesting extension of this work. It is also interesting to add a fairness term in the fitness function and check the outcomes. The fairness is to ensure that while increasing transmission they are uniformly distributed over the nodes.

In section VI.C.1, we find that the results are hardly affected by the setting of the parameter values, even when they are varied over wide ranges from the commonly used

settings. That is, the algorithm is robust against parameter values.

Adaptive tuning of crossover and mutation probabilities, larger values in the beginning and less at end generations, is an useful recipe for improving efficiency of GA. Another suggestion is that, the selection pressure too should be less in the beginning and increased slowly. To simplify the simulations we have not implemented such fine tuning of the algorithm.

As it is impossible to precisely specify an objective function for fitness calculation of the members of the population, ranking of the members is the only option available. In this case, the objective function is a subjective preference only. Additionally, by rank-based selection, we avoid the problem of fitness scaling. We used tournament selection. There are many other methods of selection based on ranking. For example, one possibility is to use a nonlinear probability function (chap.4 of [22]),

$$prob(rank) = q(1 - q)^{rank-1}$$

where, $prob(rank)$ is the probability that an individual with $rank = rank$ will be selected to the next generation. Here, q controls the selection pressure. Other selection methods as above, are possible experimental extension.

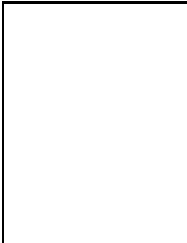
ACKNOWLEDGEMENTS

We are thankful to the reviewers for their valuable comments. This work has been partially supported by the Government of Japan, Ministry of general affairs (Soumushou) project *Research on surveillance framework for next generation ubiquitous network*.

REFERENCES

- [1] F. Tobagi, R. Binder, and B. M. Leiner, "Packet radio and satellite networks," *IEEE Comm. Mag.*, November 1984.
- [2] J. Jubin, "Current packet radio protocols," in *INFOCOM85 Proceedings*, March 1985.
- [3] A. Ephremides, and T. V. Truong, "Scheduling broadcasts in multiple radio networks," *IEEE Trans. Communication*, vol. COM-38, no. 4, pp. 456-460, April, 1985.
- [4] B. M. Leiner, D. L. Nielson, and F. A. Tobagi, "Issues in packet radio network design," *Proceedings IEEE*, vol. 75, no. 1, pp. 6-20, January 1987.
- [5] L. Kleinrock and J. Silvester, "On the behavior of packet radio networks," *Proceedings IEEE*, vol. 75, No. 1, pp. 156-167, January 1987.
- [6] F. Tobagi, "Modeling and performance analysis of multihop packet radio networks," *Proceedings IEEE*, vol. 75, No. 1, pp. 135-155, January 1987.
- [7] N. Shacham and J. Westcott, "Future directions in packet radio architectures and protocols," *Proceedings IEEE*, vol. 75, No. 1, pp. 83-99, January 1987.
- [8] M. B. Pursley, "The role of spread spectrum in packet radio networks," *Proceedings IEEE*, vol. 75, No. 1, pp. 116-134, January 1987.
- [9] F. A. Tobagi, "Multi-access protocols in packet communication systems," *IEEE Trans. Comm.*, vol. COM-28, pp. 46 488, April 1980.
- [10] I. Chlamtac, "Fair algorithm for maximal link activation in multihop radio networks," *IEEE Trans. Comm.*, vol. COM-35, no. 7, pp. 739-746, July 1987.
- [11] N. Funabiki, and S. Nishikawa, "A binary neural network approach for link activation problems in multihop radio networks," *IEICE Trans. on comm*, vol. E-79B, no. 8, pp. 1086-1093, August, 1996.

- [12] C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "A neural network approach to solving the link activation problem in multihop radio networks," *IEEE Trans. Comm.*, vol. COM-43, no. 2/3/4, pp. 1277-1283, Feb/Mar/April 1995.
- [13] R. Kahn, S. Gronemeyer, J. Burchfiel, and R. Kunzeman, "Advances in packet radio technology," *Proc. IEEE*, vol. 66, no. 11, pp. 1468-1496, Nov. 1978.
- [14] R. Nelson and L. Kleinrock, "Spatial TDMA: A Collision-free multihop channel access protocol," *IEEE Trans. Comm.*, vol. COM-33, no. 9, pp. 934-944, September 1985.
- [15] B. H. Davies and T. R. Davies, "The application of packet switching techniques to combat net radio," *Proceedings IEEE* vol. 75, No. 1, pp. 43-55, January 1987.
- [16] T.-C. Hou and V. O. K. Li, "Transmission range control in multihop packet radio networks," *IEEE Trans. Comm.*, vol. COM-34, no. 1, pp. 38-44, Jan. 1986.
- [17] H. Takagi and L. Kleinrock "Optimal transmission ranges for randomly distributed packet radio networks," *IEEE Trans. Comm.*, vol. COM-32, no. 3, pp. 246-257, March 1984.
- [18] H. Takagi and L. Kleinrock "Throughput-delay characteristics of some slotted-ALOHA multihop packet radio networks," *IEEE Trans. Comm.*, vol. COM-33, no. 11, pp. 1200-1207, Nov 1985.
- [19] A. Ephremides, and T. V. Truong, "Scheduling broadcasts in multihop radio networks," *IEEE Trans. Comm.*, vol. COM-38, no. 4, pp. 456-460, April, 1990.
- [20] *Evolutionary computation 1: Basic algorithms and operators*, Edited by Thomas Back, David B Fogel, and Z. Michalewicz, Institute of physics publishing, Bristol and Philadelphia, 2000.
- [21] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.
- [22] Zbigniew Michalewicz, *Genetic Algorithms + Data Structure = Evolution Programs*. Springer-Verlag, 1995.
- [23] Goutam Chakraborty, Y. Hirano, "Genetic Algorithm for Broadcast Scheduling in Packet Radio Networks," *Proceedings of IEEE World Congress on Computational Intelligence*, pp. 183-188, Alaska, May 4-9, 1998.
- [24] N. Funabiki, and Y. Takefuji, "A parallel algorithm for broadcast scheduling problems in packet radio networks," *IEEE Trans. on comm.*, vol. 41, no. 6, pp. 828-831, June, 1993.
- [25] G. Wang, and N. Ansari, "Optimal broadcast scheduling in packet radio networks using mean field annealing," *IEEE journal on selected areas in communication*, vol. 15, no. 2, pp. 250-260, Feb, 1997.
- [26] A. Kershenbaum, R. Boorstyn, and Mon-Song Chen, "An algorithm for evaluation of throughput in multihop packet radio networks with complex topologies," in *IEEE journal on selected areas in communication*, vol. 5, no. 6, pp. 1003-1012, July 1987.
- [27] J. A. Silvester, "Perfect scheduling in multi-hop broadcast network," in *Proc. 6th Int. Conf. on Computer Communication*, pp. 449-454, London, England, 1982.
- [28] W. C. Fifer and F. J. Bruno, "The low-cost packet radio," *Proceedings IEEE*, vol. 75, No. 1, pp. 33-42, January 1987.
- [29] I. Cidon and M. Sidi, "Distributed assignment algorithms for multihop packet radio networks," *IEEE Trans. Computers*, vol. 38, no. 10, pp. 1353-1361, October, 1989.
- [30] J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "A neural network approach to routing without interference in multihop radio networks," *IEEE Trans. Comm.*, vol. COM-42, no. 1, pp. 166-177, January 1994.
- [31] G. Wang and N. Ansari, "A neural network approach to broadcast scheduling in multihop radio networks," in *Prod. ICNN'94*, pp. 4699-4704, June 1994.
- [32] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI., 1975.
- [33] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Systems, Man and Cybernetics*, vol. SMC-16, No. 1, pp.122-128, Jan./Feb. 1986.
- [34] M. Srinivas and L. M. Patnaik, "Adaptive Probabilities of crossover and mutation in Genetic algorithms," *IEEE Trans. Systems, Man and Cybernetics*, vol. SMC-24, No. 4, pp.656-666, April 1994.
- [35] Goutam Chakraborty, and Kayo Hoshi, "Rank Based Crossover - A new technique to improve the speed and quality of convergence in GA", *Proceedings of the 1999 Congress on Evolutionary Computation*, Washington, USA, pp.1595-1602, July 6-9, 1999.
- [36] T. Back, *Evolutionary algorithm in theory and practice*, Oxford university Press, New York, 1996.
- [37] Kenneth A. De Jong, William M. Spears, "Using Genetic Algorithms to Solve NP-Complete Problems," *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 124-132, Morgan Kaufmann Publishers, CA, 1989.



Goutam Chakraborty received his Ph.D. in 1993 from Tohoku University, Japan. Presently he is a Professor in the Department of Software and Information Science, Iwate Prefectural University, Japan. His research interests are Soft Computing methods and their applications to solve different pattern recognition, classification, prediction, scheduling and optimization problems including applications to computer Networks, mobile networking and data mining.