Centralized Broadcast Scheduling in Packet Radio Networks via Genetic-Fix Algorithms

Chiu Y. Ngo, *Senior Member, IEEE,* and Victor O. K. Li, *Fellow, IEEE*

***Abstract*—An important, yet difficult, problem in the design of a packet radio network is the determination of a conflict-free broadcast schedule at a minimum cycle length. In this letter, we first formulate the problem via a within-two-hop connectivity matrix and then, by assuming a known cycle length, determine a conflict-free scheduling pattern using a centralized approach that exploits the structure of the problem via a modified genetic algorithm. This algorithm, called genetic-fix, generates and manipulates individuals with fixed size (i.e., in binary representation, the number of ones is fixed) and therefore, can reduce the search space substantially. We also propose a method to find a reasonable cycle length and shorten it gradually to obtain a near-optimal one. Simulations on three benchmark problems showed that our approach could achieve 100% convergence to solutions with optimal cycle length within reasonable time.**

***Index Terms*—Broadcast scheduling, genetic algorithms (GAs).**

I. Introduction

ONE distinguishing characteristic of a packet radio network is the broadcast nature of its radio channels. Transmission from a node in the network may be received by its neighboring nodes. Consequently, conflicts (or interference) may occur among the nodes. There are two types of conflict, namely, primary conflict and secondary conflict [1]. A primary conflict occurs when a node receives more than one transmission destined to it simultaneously. A secondary conflict occurs when a node, an intended receiver of a particular transmission, is also within the transmission range of another transmission intended for other nodes. In the context of time-division multiplexing (TDM), the problem of broadcast scheduling is to determine a conflict-free assignment of time slots to each individual nodes that satisfies the traffic requirements. Simultaneous transmissions among nodes are allowed as long as no conflict is produced and the collection of all distinct time slots forms a TDM cycle.

It has been shown that this problem is NP-complete in terms of optimal cycle length [2]. As a result, several approximate algorithms have been proposed. Our problem formulation follows a neural-network approach [3] whose objective is to obtain a conflict-free broadcast slot assignment where the cycle length is close to minimum. However, instead of using the one-hop connectivity matrix directly, we formulate the problem based on a within-two-hop connectivity matrix and propose a centralized scheduling algorithm using a modified genetic algorithm (GA), called the genetic-fix algorithm [4], [5]. Unlike the conventional GAs that generate subsets of all possible sizes, genetic-fix generates fixed-size subsets (i.e., in binary representation, the number of ones is fixed). This can greatly reduce the search space and subsequently, speed up the computation.

II. Broadcast Scheduling Problem

The packet radio network to be considered consists of n arbitrary nodes with an $n \times n$ symmetric one-hop connectivity matrix N_1 given by

$$(N_1)_{ij} = \begin{cases} 1, & \text{if nodes } i \text{ and } j \text{ can communicate with} \\ & \quad \text{each other and } i \neq j \\ 0, & \text{otherwise.} \end{cases}$$

Here, symmetric connectivity implies that any two connected or neighboring nodes can communicate with each other. We assume that the node locations are fixed and hence, the connectivity between nodes is known. Due to the inherent broadcast nature of radio channels, we assume that packets transmitted from a node can be received by all its neighboring nodes. Furthermore, fixed packet length is assumed, and the TDM scheme is deployed where time is segmented into slots, each of which equals a packet transmission time plus an appropriate guard time. For simplicity, we assume uniform traffic, i.e., each node has the same amount of externally generated traffic, and they are equally likely to be destined to all nodes. Without loss of generality, we assume one packet per node per cycle. In addition, we assume that all nodes are synchronized.

The optimal TDM broadcast scheduling involves the determination of the minimal TDM cycle length and the way to distribute the TDM slots among the nodes while satisfying the following traffic and conflict avoidance constraints: (C_1) each node uses only one slot per TDM cycle; (C_2) a node cannot send and receive a packet simultaneously; and (C_3) a node cannot receive more than one packet simultaneously. It is noted that constraint (C_2) is determined by the one-hop connectivity matrix N_1, whereas constraint (C_3) is determined by the two-hop connectivity matrix $N_2 = N_1^2$. Therefore, two nodes can transmit in the same slot without conflict only when they are more than two hops away. As a result, these two constraints can be completely determined by a within-two-hop 0-1 connectivity matrix N_* defined as

$$(N_*)_{ij} = \begin{cases} 1, & \text{if } (N_1 + N_2)_{ij} \neq 0 \text{ and } i \neq j \\ 0, & \text{otherwise.} \end{cases}$$

Note that if $(N_*)_{ij} = 1$, then node i can be reached by node j within two hops and vice versa.

Conventional graph-theoretic approaches formulate the problem as a “minimum span” problem, i.e., given the

Paper approved by B. Jabbari, the Editor for Wireless Multiple Access of the IEEE Communications Society. Manuscript received April 15, 1999; revised June 15, 2002. This research was performed at the University of Southern California, Los Angeles.

C. Y. Ngo is with Samsung Information Systems America, San Jose, CA 95134 USA (e-mail: chiu.ngo@samsung.com).

V. O. K. Li is with the Department of Electrical and Electronic Engineering, University of Hong Kong, Hong Kong.

Digital Object Identifier 10.1109/TCOMM.2003.816950

$$
\begin{array}{cc|ccccccc|}
 & f_{jk} & \multicolumn{7}{c}{\text{Slot Number}} \\
 & & 1 & 2 & 3 & \dots & & & m \\
\hline
\text{Node Number} & 1 & 0 & 1 & 0 & \dots & 0 & 0 & 0 \\
 & 2 & 0 & 0 & 1 & \dots & 0 & 0 & 0 \\
 & 3 & 0 & 0 & 0 & \dots & 1 & 0 & 0 \\
 & \vdots & & & & \vdots & & & \\
 & n & 0 & 0 & 0 & \dots & 1 & 0 & 0 \\
\hline
\end{array}
$$

Fig. 1. Representation of a TDM broadcast schedule.

within-two-hop connectivity matrix N_* of an arbitrary n-node packet radio network, find a conflict-free TDM cycle schedule such that the total number of slots, or the TDM cycle length, is minimum. However, in practice, the determination of a conflict-free scheduling pattern is more important. Our formulation follows the neural-network approach in [3], which assumes fixed cycle length and formulates the problem as an unconstrained optimization problem.

We represent the solution space F as an $n \times m$ binary matrix where n is the number of nodes and m is the length of the TDM cycle. Each element f_{jk} in the matrix is either "1" or "0" such that

$$f_{jk} = \begin{cases} 1 \\ 0 \end{cases} \text{if slot } k \text{ is} \begin{Bmatrix} \text{assigned} \\ \text{not assigned} \end{Bmatrix} \text{ to node } j.$$

Diagrammatically, the admissible TDM broadcast schedule F can be described in an array form as shown in Fig. 1. Basic requirements for the scheduling problem are the traffic demand and the avoidance of conflicts. The first requirement imposes an allocation constraint on F, i.e., there is one and only one slot allocated to each node per cycle. This implies that only one "1" is allowed in each row of F. Mathematically, it means that if the slot assignment to node i violates the allocation constraint (C_1), then

$$\left(\sum_{k=1}^{m} f_{ik} - 1\right) \neq 0.$$

The second requirement is determined by the previously defined within-two-hop connectivity matrix N_*. If slot k is assigned to node i, then slot k cannot be assigned to any node within two hops from node i. Mathematically, it means that if the assignment of slot k to node j violates the constraints (C_2) and (C_3), then

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} (N_*)_{ij} f_{jk} > 0.$$

Therefore, a generic choice of cost function is given by

$$C(F) = \alpha \sum_{i=1}^{n} \left(\sum_{k=1}^{m} f_{ik} - 1\right)^2 + \sum_{i=1}^{n}\sum_{k=1}^{m} \left(\sum_{\substack{j=1 \\ j\neq i}}^{n} (N_*)_{ij} f_{jk}\right) f_{ik} \quad (1)$$

where $\alpha \epsilon R^+$. It is noted that $C(F)$ achieves its minimum of zero when all constraints are satisfied. Hence, our problem is to find an F such that $C(F)$ is zero. Compared to [3], our formulation is much simpler. Note that by restricting each row of F to have only one "1", the traffic requirement will automatically be fulfilled and hence, the cost function can be simplified to

$$C(F) = \sum_{i=1}^{n}\sum_{k=1}^{m} \left(\sum_{\substack{j=1 \\ j\neq i}}^{n} (N_*)_{ij} f_{jk}\right) f_{ik}. \quad (2)$$

In the following, we discuss how the genetic-fix algorithm can accomplish this task by generating and manipulating only those "useful" candidate solutions represented in the above array form.

III. Principle of Genetic-Fix Algorithm

GAs are stochastic, yet structured, iterative search procedures which mimic the evolution of biological genetics that favors the fittest individuals via selection, crossover, and mutation. Generally, GAs generate subsets of all possible sizes. However, there are some combinatorial optimization problems whose feasible solutions are fixed size (i.e., in binary representation, the number of ones is fixed) subsets. We develop the genetic-fix algorithm that can generate a fixed number of ones for each individual and preserve this property during the genetic operations. Of course, this requires special crossover and mutation operators that can maintain the property of a fixed number of ones.

A. Crossover in Genetic-Fix

Given two parents A and B, we create a first-in–last-out (FILO) stack to store the bit position k corresponding to opposite bit pair (A_k, B_k). A_k and B_k are said to be opposite if $A_k \oplus B_k = 1$ where $\oplus$ denotes the exclusive OR operator. The crossover is performed by first generating two crossover points c_1 and c_2 at random along the string length, such that $c_1 < c_2$ and then moving right from c_1 until an i is found, such that $A_i \oplus B_i = 1$. We push i into the FILO stack and continue the process until we find a j such that $A_j \oplus B_j = 1$. Then, we compare A_j with A_{s1} where $s1$ is the top element in the stack. If they are the same, we push j into the stack; otherwise, we swap the pair indexed by j with the pair indexed by $s1$ and pop $s1$ from the stack. The process continues until c_2 is reached. An example can be found in [5].

B. Mutation in Genetic-Fix

In order to balance the number of ones in an individual, the mutation operation must always be done in pair of opposite bits. This can be implemented as follows. Let b_i be the ith bit position of an individual. To mutate b_i, we need to find a random b_j such that $b_i \oplus b_j = 1$. Then, we swap b_i with b_j. In case of binary array representation, both b_i with b_j must be in the same row.

For details of these operatons, see [4] and [5]. When the "elitist" selection strategy is used such that the best individual survives with probability one, it has been shown the genetic-fix algorithm converges asymptotically [4].

IV. Application of Genetic-Fix to Broadcast Scheduling

By representing each individual as a unit-row binary array, as shown in Fig. 1, and preserving the unit-row property throughout the process using the genetic-fix operators, we can apply the genetic-fix algorithm to solve the scheduling problem (2) easily. However, one still has to determine the cycle length m. This number can be determined by the lower bound estimated by either a graph-theoretic method or some other heuristics. In the following, we present a stochastic heuristic method that can determine a near-optimal m. The method starts with a matrix N which equals the within-two-hop matrix N_* and an empty slot matrix S, and then proceeds as follows.

1) Select randomly a "nonone column," say column j, from N_*. (A "one column" is a column whose elements are all ones.)
2) Put index j into the first empty row, say r, of S and set the jth row of N_* to all ones to indicate that node j has been assigned to slot r.
3) Select randomly a zero, say indexed by i, from column j of N. (Zeros in column j are possible co-nodes of node j that can share the same slot r.)
4) Append index i to row r of S and set the ith row of N_* to all ones to indicate that node i has been assigned to slot r.
5) Repeat steps (3)–(4) until column j becomes a "one column."
6) Repeat steps (1)–(5) until N_* becomes a "one matrix."
7) Save the slot assignments and the corresponding cycle length.

This procedure is repeated until a reasonable minimum cycle length, say m', is achieved. Then, we apply the genetic-fix algorithm to solve the scheduling problem using a cycle length shorter than m'. Starting from $m' - 1$, we repeatedly decrease the cycle length by one until we cannot find a solution. The smallest feasible cycle length obtained thus far will be used as our m in the scheduling problem.

V. Simulations

The simulator used was called GENESIS_F (see [5]). Three benchmark problems were examined. Problem 1 was taken from [1], problem 2 from [6], and problem 3 from [3]. For each problem, 100 Monte Carlo runs were performed. In order to avoid the disappearance of the best individual, the "elitist" selection strategy was adopted so that the best individual always survives intact from one generation to the next. In addition, a local search routine described in [5] was used to improve the performance of the algorithm. Several parameters need to be set, including the maximum number of trials per run (T trials), the crossover probability (p_c), the mutation probability (p_m), the population size (N_s), the size of the penalty vector (n_p), and the counter for "igniting" the local search routine (K generations). Table I summarizes the characteristics of these three problems and their corresponding simulation parameters.

Simulations were performed on an HP Apollo 9000/700 workstation using our proposed genetic-fix algorithm in GENESIS_F. Table II summarizes the results. For ease of comparison, the corresponding neural-network performance in [3] is also included.

The results show that genetic-fix gives very good results. Compared with the low frequency of convergence in the neural-network approach, our algorithm gives 100% convergence in all three problems. Furthermore, in Problem 3, we find a shorter cycle length of nine instead of ten, given in [3].

TABLE I
Simulation Parameters

Problem	No. of nodes	No. of edges	Max. nodal degree	T	p_c	p_m	N_s	n_p	K
1	14	23	5	200000	0.95	0.01	10	14	1
2	16	23	4	200000	0.95	0.01	10	16	1
3	400	805	8	200000	0.95	0.001	10	100	10

TABLE II
Summary of Simulation Results

Problem	Neural Network		Genetic-Fix			
	m	Frequency of Convergence (%)	m	Frequency of Convergence (%)	No. of Trials	CPU Time (sec)
1	6	35	6	100	162.4 (± 157.4)	0.1384 (± 0.1216)
2	5	26	5	100	96.7 (± 87.1)	0.0946 (± 0.0731)
3	10	37	9	100	34885 (±16285)	4117 (±2636)

For a network with maximum nodal degree of n_d, the lower bound on the cycle length m will be $(n_d + 1)$ because the node and each of its neighbors requires one slot. With reference to the maximum nodal degrees in Table I, the cycle lengths determined by our algorithm are, in fact, optimum for all three problems. In addition, we found during our simulation that our heuristic used in determining m gave even better results than the neural-network approach in Problems 1 and 2.

VI. Conclusions

We have studied the problem of conflict-free TDM broadcast scheduling in packet radio networks. Using a within-two-hop matrix, we obtained a simple problem formulation. We proposed an approach based on a modified GA. This algorithm, called genetic-fix, generates and manipulates individuals with fixed size so as to reduce the search space substantially. Simulations on three benchmark problems showed that this algorithm could achieve 100% convergence to solutions with optimal cycle length within reasonable time. In one case, a cycle length shorter than that obtained by the neural-network algorithm [3] was found. Such significant results indicate that the genetic-fix algorithm is indeed a good method for solving the broadcast scheduling problem. Although uniform traffic is assumed, this algorithm can easily accommodate nonuniform traffic requirements (see [4]).

References

[1] A. Ephremides and T. V. Truong, "Scheduling broadcasts in multihop radio networks," *IEEE Trans. Commun.*, vol. 38, pp. 456–460, Apr. 1990.
[2] R. Ramaswami and K. K. Parhi, "Distributed scheduling of broadcasts in a radio network," in *Proc. IEEE Conf. Computer Communications, INFOCOM '89*, 1989, pp. 497–504.
[3] N. Funabiki and Y. Takefuji, "A parallel algorithm for broadcast scheduling problems in packet radio networks," *IEEE Trans. Commun.*, vol. 41, pp. 828–831, June 1993.
[4] C. Y. Ngo, "Genetic algorithms for discrete optimization and their applications to radio network design," Ph.D. dissertation, Dept. Elect. Eng.—Syst., Univ. Southern California, Los Angeles, CA, Aug. 1995.
[5] C. Y. Ngo and V. O. K. Li, "Fixed channel assignment in cellular radio networks using a modified genetic algorithm," *IEEE Trans. Veh. Technol.*, vol. 47, pp. 163–172, Feb. 1998.
[6] I. Cidon and M. Sidi, "Distributed assignment algorithms for multihop packet radio networks," *IEEE Trans. Comput.*, vol. 38, pp. 1353–1361, Oct. 1989.