PAPER

A Greedy Genetic Algorithm for the TDMA Broadcast Scheduling Problem*

Chih-Chiang LIN[†], *Nonmember and* **Pi-Chung WANG**[††a)], *Member*

SUMMARY The broadcast scheduling problem (BSP) in wireless ad-hoc networks is a well-known NP-complete combinatorial optimization problem. The BSP aims at finding a transmission schedule whose time slots are collision free in a wireless ad-hoc network with time-division multiple access (TDMA). The transmission schedule is optimized for minimizing the frame length of the node transmissions and maximizing the utilization of the shared channel. Recently, many metaheuristics can optimally solve smaller problem instances of the BSP. However, for complex problem instances, the computation of metaheuristics can be quite time and memory consuming. In this work, we propose a greedy genetic algorithm for solving the BSP with a large number of nodes. We present three heuristic genetic operators, including a greedy crossover and two greedy mutation operators, to optimize both objectives of the BSP. These heuristic genetic operators can generate good solutions. Our experiments use both benchmark data sets and randomly generated problem instances. The experimental results show that our genetic algorithm is effective in solving the BSP problem instances of large-scale networks with 2,500 nodes.

key words: *broadcast packet radio networks, broadcast scheduling problem, optimum transmission schedule, time division multiple access, genetic algorithm*

1. Introduction

A broadcast packet radio network is a group of geographically distributed nodes, which are connected through a common radio channel. The radio channel is shared by time-division multiple access (TDMA). To avoid packet collisions, broadcast scheduling allows only one node transmission in each collision domain [1], and each node must be assigned at least one time slot in each time frame. In addition, the utilization of the shared channel can be further improved by assigning more than one transmitting node in a time slot. The aim of the broadcast scheduling problem (BSP) is to schedule the node transmissions in the shared channel with a time frame whose length is minimum and utilization is maximum.

The BSP can be treated as a combinatorial optimization problem known to be NP-complete [2]. Several scheduling algorithms have been proposed. These algorithms include exact methods [3], heuristic methods [4]–[11], and meta-heuristic methods [12]–[15]. The exact methods for the BSP (e.g., *brand−and−bound*) can optimally solve smaller problem instances. However, for large problem instances, the computation of the exact methods is quite time and memory consuming. Although the existing heuristic and metaheuristic methods have better scalability than the exact methods, the existing literature only solves problem instances with up to several hundred nodes. Currently, many applications of large-scale wireless sensor networks have been developed, e.g. battlefield surveillance and environment monitoring [16], where the number of nodes can be as large as several thousand nodes. An efficient solution for large BSP instances is thus getting important.

In this paper, we propose a greedy genetic algorithm for the BSP. In our algorithm, we propose three greedy genetic operators, including a crossover and two mutation operators. The proposed crossover operator uses a greedy method to generate offsprings. Both mutation operators use greedy approaches, one to minimize timeframe length and the other to maximize channel utilization With these genetic operators, the proposed genetic algorithm can generate good results of TDMA broadcast schedule. Our experiments show that the new scheme can solve large problem instances with 2,500 nodes. The numerical results also demonstrate that our algorithm is not only effective and efficient regardless of the network sizes but also significantly reduces computation time.

The organization of this paper is as follows. The next section describes the TDMA broadcast scheduling problem. Section 3 addresses related work. In Sect. 4, we describe the framework of our genetic algorithm and the proposed heuristic genetic operators for the BSP. Section 5 presents our performance analysis. Finally, we conclude our work in Sect. 6.

2. The Broadcast Scheduling Problem

The broadcast scheduling problem is based on a network graph, $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_N\}$ denotes the set of N network nodes, and $E = \{e_1, e_2, \ldots, e_M\}$ denotes the set of M transmission links. If there exists an edge between two nodes i and j in V, then the nodes i and j are one-hop apart and can receive the transmitted packet from each other. If they transmit packets at the same time slot, a direct collision occurs. In another case, nodes i and j are two-hop apart. If both nodes transmit packets to their intermediate node at

Manuscript received August 30, 2011.
Manuscript revised June 23, 2012.

[†]The author is with the Department of Computer Science and Engineering, National Chung Hsing University, Taichung 402, Taiwan, R.O.C.
[††]The author is with the Institute of Networking and Multimedia and the Department of Computer Science and Engineering, National Chung Hsing University, Taichung 402, Taiwan, R.O.C.

[*]This work is supported in part by the National Science Council under Grant No.NSC 99-2218-E-241-001.

a) E-mail: pcwang@cs.nchu.edu.tw (Corresponding author)

DOI: 10.1587/transinf.E96.D.102

the same time slot, a hidden terminal collision occurs. In other words, each node cannot transmit and receive packets simultaneously to avoid the direct collision. Each node also cannot receive more than one packet simultaneously to avoid the hidden terminal collision.

The BSP is defined as follows.

1. Input Data
   - $G : G = (V, E)$, where $V = v_1, v_2, \dots, v_n$ and $E = e_1, e_2, \dots, e_m$.
   - c_{ij}: indicates whether node j can receive packets from node i, i.e., there exists an edge with one end node i and the other end node j.
2. Decision Variables
   - x_{si}: the binary variable indicates whether node i transmits packets at the s_{th} time slot.
   - F: The time frame of a TDMA schedule.
   - $|F|$: The number of time slots in a time frame F.

A time frame F is a set of time slots in which at least one node i exists that satisfies $x_{si} = 1, 1 \leq i \leq n$, for each time slot s in F. The channel utilization for the whole network, U, is provided by

$$U = \frac{1}{|F|n} \sum_{s=1}^{|F|} \sum_{i=1}^{n} x_{si} \tag{1}$$

The BSP aims at minimizing the TDMA frame F and maximizing the channel utilization U, which is subjected to

$$\sum_{s=1}^{|F|} x_{si} \leq 1, \text{where } 1 \leq i \leq n \tag{2}$$

$$c_{ij} + x_{si} + x_{sj} \leq 2, \text{where } i \text{ and } j \text{ denote two different nodes in V (i.e.,} i \neq j), \forall s \in F \tag{3}$$

$$c_{ik}x_{si} + c_{kj}x_{sj} \leq 1, \text{where } i, j \text{ and } k \text{ denote three different nodes in V}, \forall s \in F \tag{4}$$

The first constraint in Eq. (2) means that each node should transmit at least once in each frame. Equation (3) and (4) guarantee a collision-free transmission. Equation (3) implies that every pair of two stations that are one hop apart must be scheduled to transmit in different time slots. Equation (4) implies that every pair of two stations that are two hops apart cannot transmit in the same time slot.

Let us consider an example of a network consisted of seven consecutive nodes in Fig. 1. If two adjacent nodes (e.g. nodes 2 and 3) transmit a packet to each other at the same time slot, then the packets from both nodes will collide. Yet, for a node which is sandwiched between two other nodes, the packets sent by the two outer nodes simultaneously will also collide with each other (e.g. the packets sent to node 5 from nodes 4 and 6).

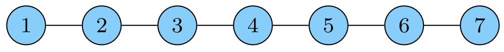


Fig. 1 Seven-node network.

The BSP aims at finding a TDMA time frame which satisfies that there is no direct and hidden terminal collisions. In addition, each node should be scheduled to transmit at least once in a time frame. To avoid both direct and hidden terminal collisions, an intuitive solution for the BSP is to assign one time slot to each node sequentially; however, such solution may not be suitable for a large-scale network due to the low performance of a long time frame. To improve the transmission efficiency, a broadcast scheduling should maximize the number of transmitting nodes at the same slot. For each node, we define the node utilization as the ratio of the number of transmission slots to the frame length. The overall channel utilization is defined as the average node utilization (i.e. average number of per-node transmission slots to the frame length). Intuitively, a time frame with a shorter frame length and higher channel utilization always has better transmission efficiency.

We can estimate the minimum required frame length according to the maximum degree of a node in a wireless ad-hoc network. Assume that the maximum degree is denoted as D for the nodes in V. The tight lower bound for a frame length is equal to $D + 1$ [4] since these nearby nodes must transmit in different time slots. Therefore, there are more than $2^{N(D+1)}$ schedule configurations, where N denotes the number of nodes in the network. An exhaustive search for the optimal schedules is prohibitive when D and N get larger. We need an efficient approach to finding a suboptimal solution for the BSP.

3. Related Work

The BSP has been proved to be a NP-complete combinatorial optimization problem [2]. Several heuristics and metaheuristic algorithms are proposed to solve the BSP in literatures [6], [7], [13], [14].

Cidon and Sidi [8] developed a new distributed dynamic channel assignment algorithm for a multihop packet radio network by ensuring conflict-free transmissions. The algorithm splits the shared channel into a control segment and a transmission segment and uses the control segment to avoid conflicts among nodes and increase the utilization of the transmission segment. Ephremides and Truong [9] presented a centralized algorithm that runs in polynomial time. They suggested a simple heuristic for achieving efficient schedules. Funabiki and Takefuji [10] used a parallel algorithm based on an artificial neural network model. It consists of a large number of simple processing elements which can be executed in parallel. In [6], the author presented a two-phase algorithm to find a collision-free time frame. In the first phase, they used the sequential vertex coloring algorithm, a well known method in graph theory, to determine a basic time frame. After completing the first phase, they use the frame length achieved to maximize the channel utilization in the second phase. The method can find near-optimal solutions in a short time period. Ahmad et al. [11] used a heuristic approach based on concepts of finite state machine by employing a tight lower bound derived from the maximal incompatibles and generating a search space from the set of maximal compatibles. The algorithm is very efficient and effective in conquering the intractable nature of BSP by exploring complex solution space within shorter CPU time. Ergen and Varaiya [7] presented two centralized heuristic algorithms. The first one is based on direct node-based scheduling, which is widely adapted in classical multi-hop scheduling algorithms. The second algorithm is level-based scheduling for many-to-one communication in sensor networks, which generates the levels in a routing tree before scheduling.

The metaheuristic algorithms are apply both intensification and diversification operators to solve the BSP problem [17]. They can be classified into two categories: trajectory-based methods and population-based methods. The trajectory methods include iterated local search (ILS) and greedy randomized adaptive search procedure (GRASP). In [15], the authors presented an ILS algorithm for BSP transmission schedule that consists of two special perturbation and local search operators. The perturbation operator can improve the search diversification by generating new solutions and the local search operator further reduces the length of a time frame to improve the search intensification. This algorithm is simple and has excellent performance. Chakraborty et al. [14] presented a GRASP algorithm for optimum transmission schedule. The algorithm generates a pool of valid solutions without considering the optimization criterion of minimizing the TDMA frame length. The time complexity of the algorithm is $O(N^2)$.

Most population-based algorithms of BSP is based on the genetic algorithm. It provides a natural, intrinsic way for exploring the search space. Chakraborty showed that a standard genetic algorithm is only suitable for solving small problem instances and performs poorly for large networks [13]. This is because that classical crossover and mutation operations could create invalid members. These members flood the whole population and hinder the progress of searching for valid solutions. Therefore, the author defined specialized crossover and mutation operators, which can assure that the members of the population are always valid solutions. With the new operators, good solutions can obtained in a few generations.

4. The Proposed Greedy Genetic Algorithm

4.1 Framework

Genetic algorithms provide a powerful stochastic search in optimization problems. In a genetic algorithm, a chromosome is an encoding of the solution to an optimization problem. Its performance is evaluated by using a fitness function. The genetic algorithm uses a population of chromosomes for applying the genetic operators, such as *Crossover* and *Mutation*, to explore search space (or solution space). The genetic algorithm also uses evolution operators, such as *Selection*, to choose good solutions from the population of each generation. A genetic algorithm usually involves the following procedures. First, an initial population of the chromosomes is randomly generated. Second, the fitness values of the chromosomes are evaluated by using a fitness function. Third, the chromosomes are applied with the genetic and evolution operators to create new chromosomes, and the population of the next generation are selected. Finally, the second and third steps are repeated until one or more termination conditions are satisfied. We list the pseudocode of the genetic algorithm in Fig. 2, where P denotes the population of the chromosomes. There are $|P|$ chromosomes in a population. Each generation of the population is denoted as P_t, where t is the generation number. P' is a temporary population for selection.

```
PROCEDURE GeneticAlgorithm()
$t \leftarrow 0$
InitialPopulation($P_t$)
Evaluation($P_t$)
WHILE termination conditions not met DO
    $P' \leftarrow \emptyset$
    Crossover($P_t, P'$)
    Mutation($P_t, P'$)
    $P_{t+1} \leftarrow$ Selection($P_t, P'$)
    $t \leftarrow t + 1$
ENDWHILE
```

Fig. 2 Pseudocode of the genetic algorithm.

One advantage of a genetic algorithm is its capability of evaluating a solution without any a prior knowledge. However, the flexibility would also degrade the efficiency of producing good solutions. It is intuitive to use a greedy method for improving the performance of generating good solutions.

In the following sections, we describe the encoding scheme, initial solution generation and the genetic operators used in our genetic algorithm. First, we introduce the encoding scheme, which uses an integer string coding for the solutions of BSP. Next, the initial operator, which generates an initial chromosome by using random permutation and a simple *next-fit* algorithm, is presented. In the third part, we show the proposed crossover operator for guiding the search space and producing new solutions. Our crossover operator adopts a greedy method to select the time slots with more transmitting nodes. With this operator, good parts of a chromosome can be kept to generate better offsprings. For the genetic algorithm, the greedy method also achieves better exploitation. Finally, we design two different mutation operators in order to simultaneously consider two optimization objectives, the minimum TDMA time frame length and the maximum channel utilization.

4.2 Encoding

In the proposed genetic algorithm, the chromosome encoding is carried out by using an integer string coding, which is similar to those used in most combinatorial optimization problems [18]. We number each node and use a string array representation for TDMA time frame. Each string represents a time slot. For example, recall to the seven-nodes

```
PROCEDURE NewChromosome()
C ← $\phi$
i = 1
$\mathbb{V}$ ← RandomPermutation(V)
FOR j ← 1 TO N
    IF Collision(C[i], $\mathbb{V}$[j]) = true THEN
        i = i + 1
        AddNode(C[i], $\mathbb{V}$[j])
    ENDIF
NEXT j
RETURN C
```

Fig. 3 Pseudocode of the proposed procedure of generating a new chromosome.

network in Fig. 1. Assume that a chromosome C is listed as {1,4,7}{2,5}{3,6}. The chromosome implicates that there are three time slots. The first time slot includes three transmitting nodes, 1, 4 and 7. The second time slot includes two transmitting nodes, 2 and 5, and the last two nodes, 3 and 6, transmit in the third time slot. $|C|$, the frame length of chromosome C, is thus equal to *three*.

4.3 Initiation

Our initial operator generates a new chromosome by using random permutation and a simple *next-fit* approach. We first create a node list, $\mathbb{V}$, by using the random permutation of N nodes. Next, the first node of $\mathbb{V}$, denoted as $\mathbb{V}[1]$, is inserted into the first time slot, $C[1]$, of a new time frame, C. Since there is only one node in $C[1]$ initially, no collision occurs. Next, we insert the second node, $\mathbb{V}[2]$, into $C[1]$. Before the node insertion, we check whether node $\mathbb{V}[2]$ will cause a collision with the nodes in $C[1]$. If *no*, node $\mathbb{V}[2]$ is inserted into $C[1]$. Otherwise, $\mathbb{V}[2]$ is inserted into a new time slot, $C[2]$. After inserting $\mathbb{V}[2]$, we repeat the above procedure to insert the third node, $\mathbb{V}[3]$, and so on. A new chromosome is created until every node has been inserted into one of the time slots of C. The pseudocode of the proposed initial operator is listed in Fig. 3, where i indicates the number of the time slots in C. The function, *RandomPermutation*(V), generates a random permutation of the nodes in V. Another function, *AddNode*($C[i], \mathbb{V}[j]$), inserts node $\mathbb{V}[j]$ to time slot $C[i]$. The function, *Collision*($C[i], \mathbb{V}[j]$), is a boolean function for checking whether node $\mathbb{V}[j]$ collides with the nodes in $C[i]$.

Let us consider the example of seven consecutive nodes in Fig. 1 again. First, we create a new time frame C and randomly generate a node permutation, $\mathbb{V} = \{4,7,3,2,5,1,6\}$. Next, we insert node 4 into the first time slot, $C[1]$, of C. Node 7 is then inserted into $C[1]$ since the signals from node 4 and node 7 do not collide with each other. However, the third node, node 3, will cause a direct collision with node 4 and must be inserted into a new time slot, $C[2]$. Node 2 is also inserted into a new time slot since it also causes a direct collision with node 3. We repeat the operation for all nodes and get a new time frame, $C = \{4,7\}\{3\}\{2,5\}\{1,6\}$. The *InitialPopulation*() function in Fig. 2 will repeatedly call the *NewChromosome*() procedure $|P|$ times to generate the first generation of chromosomes (initial solutions).

```
PROCEDURE Crossover(P, P′)
FOR j ← 1 TO (|P| × CrossoverRatio)
    Parent1 ← RandomSelection(P)
    Parent2 ← RandomSelection(P)
    WHILE (Parent1= Parent2)
        Parent2← RandomSelection(P)
    C ← $\phi$
    $\mathbb{C}$ ← Parent1 ∪ Parent2
    WHILE $\mathbb{C}$ is not empty DO
        $S_{max}$ ← $\mathbb{C}$[1]
        FOR i ← 2 TO |$\mathbb{C}$|
            IF |$\mathbb{C}$[i]| > |$S_{max}$| THEN $S_{max}$ ← $\mathbb{C}$[i]
        NEXT i
        $\mathbb{C}$ ← $\mathbb{C}$ − $S_{max}$
        C ← C ∪ $S_{max}$
        FOR i ← 1 TO |$\mathbb{C}$|
            IF $\mathbb{C}$[i] ∩ $S_{max}$ ≠ $\phi$ THEN
                $\mathbb{C}$[i] ← $\mathbb{C}$[i] − ($\mathbb{C}$[i] ∩ $S_{max}$)
                IF $\mathbb{C}$[i] = $\phi$ THEN $\mathbb{C}$ ← $\mathbb{C}$ − $\mathbb{C}$[i]
            ENDIF
        NEXT i
    ENDWHILE
    INSERT $\mathbb{C}$ to P′
NEXT j
```

Fig. 4 Pseudocode of the proposed crossover operator.

4.4 Crossover Operator

The crossover operator attempts to generate a better solution from the existing solutions. In our genetic algorithm, we use a greedy method to serve the purpose by selecting the time slot with the most transmitting nodes. The selection can minimize the number of nodes that must transmit in the other time slots. Although the greedy method cannot guarantee to yield the shortest time frame, it shortens the calculation time and also produces a near-optimal time frame, which is shown in our experimental results.

The steps of the crossover operator are described as follows. First, we randomly select two chromosomes, denoted as *Parent1* and *Parent2*, from the current population P. We combine the chromosomes of *Parent1* and *Parent2* to a temporary chromosome, $\mathbb{C}$. Next, the time slot S_{max} with the most transmission nodes in $\mathbb{C}$ is chosen and moved from $\mathbb{C}$ to a new chromosome, C. In the fourth step, we remove the nodes in S_{max} from $\mathbb{C}$. If a time slot in $\mathbb{C}$ becomes *empty*, the time slot is removed. The above steps are repeated until $\mathbb{C}$ is empty. The pseudocode of our crossover procedure is listed in Fig. 4. The function, *RandomSelection*(P), randomly selects two chromosomes from the population P. The number of iterations of executing the crossover operator is determined by *crossover ratio*, which indicates the ratio of new chromosomes generated by the crossover operator. The new chromosomes are stored in P'.

For the example of seven consecutive nodes in Fig. 1, assume that we randomly select two parent chromosomes, *Parent1* $= \{1,5\}\{2,5\}\{3,6\}\{4,7\}$ and *Parent2* $= \{1,4,7\}\{3,7\}\{2,6\}\{5\}$. We merge *Parent1* and *Parent2* to produce a new chromosome $\mathbb{C} = \{1,5\}\{2,5\}\{3,6\}\{4,7\}\{1,4,7\}\{3,7\}$

```
PROCEDURE LengthMutation($P, P'$)
$C \leftarrow RandomSelection(P)$
$S \leftarrow RandomRemoveTimeSlot(C)$
FOR $i \leftarrow 1$ TO $|C|$
    $C[i] \leftarrow C[i] - (C[i] \cap S)$
NEXT $i$
FOR $j \leftarrow 1$ TO $|S|$
    FOR $i \leftarrow 1$ TO $|C|$
        IF $Collision(C[i], S[j]) = false$ THEN
            $AddNode(C[i], S[j])$
            $RemoveNode(S, S[j])$
        ENDIF
    NEXT $i$
NEXT $j$
IF $S \neq \phi$ THEN $C \leftarrow C \cup S$
ENDIF
INSERT $\mathbb{C}$ to $P'$
```

Fig. 5 Pseudocode of our mutation operator for shortening time frame C.

$\{2,6\}\{5\}$. Next, we can find the time slot with the most transmission nodes, $S_{max} = \{1, 4, 7\}$, in $\mathbb{C}$. S_{max} is then removed from $\mathbb{C}$ and inserted into a new time frame, C. In addition, we also remove the transmitting nodes in S_{max} from the other time slots $\mathbb{C}[i]$, $1 \leq i \leq |\mathbb{C}|$ of $\mathbb{C}$. If $\mathbb{C}[i]$ becomes empty, $\mathbb{C}[i]$ is removed from $\mathbb{C}$. We execute the above procedure until $\mathbb{C}$ is empty to get a new chromosome, $C = \{1,4,7\}\{2,5\}\{3,6\}$.

4.5 Mutation Operators

A mutation operator can generate a new chromosome from a chromosome of the current population. The number of chromosomes generated by mutation is controlled by *mutation ratio*. We design two greedy mutation operators, where each operator is designed for one objective of BSP, the minimum TDMA time frame length or the maximum channel utilization. When a chromosome is selected for mutation, a probability is used to decide which mutation operator is used.

We start by describing the mutation operator for shortening a time frame. First, we randomly select a chromosome C in the current population P. We further select a time slot S from C. The time slot S is removed from C, and all nodes in S are also removed from the other time slots of C. Next, we attempt to reinsert each node of S into a time slot of C. The nodes which have been successfully inserted are then removed from S. If there is at least one node of S which cannot be inserted, then S is appended to C again. The pseudocode of our mutation operator of shortening a time frame is listed in Fig. 5, where $S[i]$ denotes i_{th} node of S. The function, *RandomRemoveTimeSlot*(C), randomly removes a time slot from chromosome C. Another function, *RemoveNode*($S, V[i]$), removes node $V[i]$ from S. If $V[i]$ does not exist in S, then the *RemoveNode* function does nothing.

We use the example in Fig. 1 to explain further. First, we randomly select a chromosome $C = \{2,5\}\{4,7\}\{3,6\}\ \{1,5\}$ from the current population. We further select a time slot

```
PROCEDURE UtilizationMutation($P, P'$)
$C \leftarrow$ RandomSelection($P$)
$S \leftarrow$ RandomRemoveTimeSlot($C$)
FOR $i \leftarrow 1$ TO $|V|$
    IF $V[i] \notin S$ THEN
        IF Collision($S, V[i]$) = $false$ THEN
            AddNode($S, V[i]$)
        ENDIF
    ENDIF
NEXT $i$
FOR $i \leftarrow 1$ TO $|C|$
    IF $S \neq \phi$ THEN
        IF $S = S \cup C[i]$ THEN
            $C[i] \leftarrow S$
            $S \leftarrow \phi$
        ENDIF
    ENDIF
NEXT $i$
IF $S \neq \phi$ THEN
    $C \leftarrow C \cup S$
ENDIF
INSERT $\mathbb{C}$ to $P'$
```

Fig. 6 Pseudocode of our mutation operator for improving channel utilization.

$S = \{2, 5\}$ and remove nodes 2 and 5 from the other time slots of chromosome C. After updating the time slots, chromosome C consists of three time slots: $C = \{4,7\}\{3,6\}\{1\}$. Next, we attempt to insert nodes 2 and 5 into C. Only node 5 can be successfully inserted into the third time slot of C without causing any collision. Since node 2 is still left in S, S is reinserted into the chromosome C. Finally, we have a new chromosome $C = \{4,7\}\{3,6\}\{1,5\}\{2\}$.

Next, we introduce the mutation operator for increasing channel utilization. In the first step, we randomly select a chromosome C from the current population P and randomly select a time slot S from C. We attempt to insert any nodes without causing any collision into S. Next, we check C whether it has any redundant time slots whose transmitting nodes also belong to S. If *yes*, then the time slot is replaced by S. As a result, the time frame length is also reduced. The pseudocode of our second mutation operator is listed in Fig. 6.

Let us apply the proposed mutation operator to the seven consecutive nodes again. First, we randomly select a chromosome, $C = \{2\}\{1,4,7\}\{3,6\}\{5\}$, and one of its time slot, $S = \{2\}$. We add a non-collided node 5 into time slot S and generate a new chromosome, $C = \{2,5\}\{1,4,7\}\{3,6\}\{5\}$. In the new chromosome, we notice that the first and the fourth time slots can be merged since the fourth time slot only contains node 5. Finally, we generate another new solution, $C = \{2,5\}\{1,4,7\}\{3,6\}$.

Since there are two mutation operators, we use two probabilities, *LengthMutation* and *UtilizationMutation*, to control the number of chromosomes mutated by the mutation operators. The new chromosomes are also stored in P' for further selection.

4.6 Selection

The selection operator is based on a tournament procedure, which compares each chromosome in P_t with a randomly selected chromosome in P'. The one with a shorter time frame is stored in P_{t+1}. If both chromosomes have the same time frame length, then the one with higher utilization is selected. The time frame length has higher priority because the end-to-end transmission latency can benefit from a shorter time frame.

4.7 Termination Condition

The termination conditions can be a maximum number of generations, maximum runtime, or the maximum number of generations without improvements. In our algorithm, the termination condition is the maximum number of generations.

5. Performance Analysis

In this section, we present the experimental results of our greedy genetic algorithm for BSP instances. Since there is no standard benchmark problem set for BSP, we perform the experiments by using five problem instances from [6] and [13], where these data sets contain 15, 30, 40, and 100 nodes with different connectivity degrees. Also, we randomly generate several square networks with $5 \times 5, 10 \times 10, 20 \times 20, 30 \times 30, 40 \times 40$ and 50×50 nodes and different degrees. These problem instances are used to measure the performance of our algorithm for different network sizes. The performance metrics include TDMA time frame length, channel utilization and runtime. We also show the number of iterations to generate the best results. For each problem instance, we repeat our genetic algorithm 30 times for statistical evaluation. We compare our algorithm with several algorithms based on different optimization techniques, including heuristic [6] and metaheuristic [13], [15], where [13] is a population-based metaheuristic and [15] is a trajectory-based metaheuristic.

In our genetic algorithm, we use the following parameters throughout our experiments:

- Population Size = 10
- Crossover Ratio = 0.5
- Mutation Ratio = 0.5
- *LengthMutation* Probability = 30%
- *UtilizationMutation* Probability = 70%
- Maximum number of iterations = 10000

Both crossover and mutation ratios determines the number of new chromosomes generated by the corresponding genetic operators. Since there are two mutation operators, we use *LengthMutation* and *UtilizationMutation* probabilities to control their usage. In the following experiments, we adjust these ratios and probabilities to show the effect of different genetic operators.

In general, a small population may degrade the quality of solutions while a large population would lead to long computation time. In our genetic algorithm, we limit the population size to only ten chromosomes for shortening the computation time. We further employ the greedy method in our genetic operators to achieve comparable results as with large population. We use the 100-node network instance in [13] to show the affect of population size. We vary the number of chromosomes from 10 to 100 to evaluate the optimization results. As shown in Fig. 7, a larger population size does not shorten the frame length. While channel utilization can be slightly improved with 40 or more chromosomes, the computation time is proportional to the number of chromosomes. Therefore, we believe that a population with ten chromosomes can leverage the tradeoff between the optimization results and computation time.

We extend our experiments to the problem instances in the previous work [6], [13]. Table 1 lists the data instances along with their properties. For each data instance, the best and the worst optimization results are listed. We also show means and standard deviations for frame length and utilization. As shown in Table 1, our genetic algorithm can generate a TDMA frame whose length is less than or equal to ten time slots for all instances. Channel utilization varies from 0.108 to 0.2 for different instances. For the smallest problem instance, our algorithm can generate the best result within 380 iterations. The number of iterations increases to several thousand for large instances. The runtime for these instances is less than 5.1 seconds. For all data instances, our

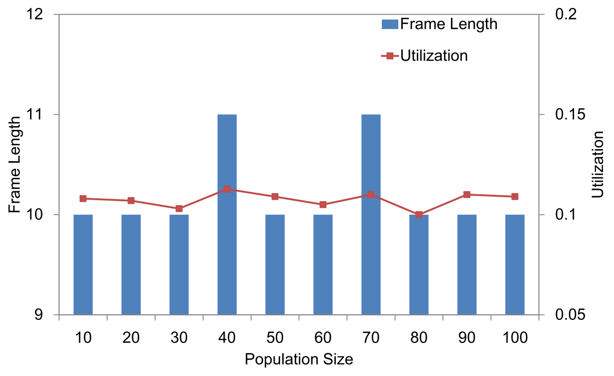
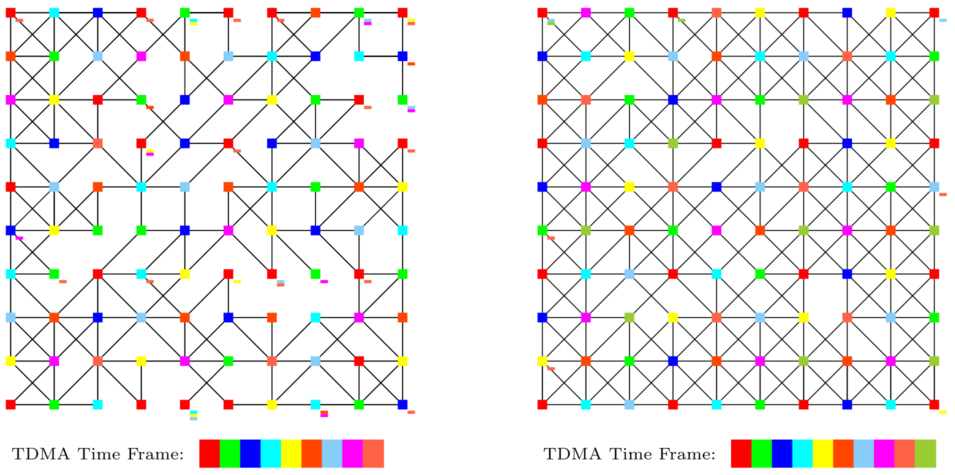


Fig. 7 A performance comparison of our genetic algorithm with different population size.

Table 1 Experimental results with benchmark data sets.

Set No.	No. of Nodes	Avg. Degree	Max. Degree	Frame Length Min.	Frame Length Max.	Frame Length Avg.	Frame Length Std.	Channel Utilization Min.	Channel Utilization Max.	Channel Utilization Avg.	Channel Utilization Std.	Required no. of Iterations	Runtime (sec.)
1	15	3.8	7	8	8	8.0	0.00	0.167	0.167	0.167	0.000	380	0.062
2	30	4.6	8	10	10	10.0	0.00	0.100	0.116	0.116	0.004	2,172	0.687
3	40	3.3	7	8	8	8.0	0.00	0.193	0.200	0.199	0.003	3,850	1.000
4	100	4.0	8	9	9	9.0	0.00	0.135	0.148	0.147	0.003	8,129	5.078
5	100	6.0	8	10	11	10.4	0.48	0.100	0.111	0.108	0.002	4,968	3.562



(a) Data Set No.4 (b) Data Set No.5

Fig. 8 The broadcast scheduling for the last two instances in Table 1 with our genetic algorithm.

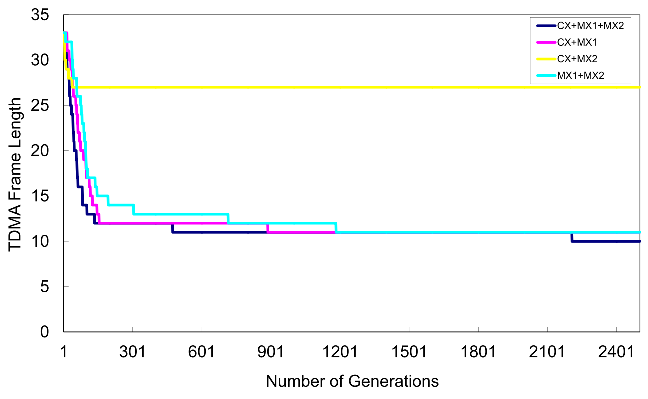


Fig. 9 Performance comparison of different genetic operators for the TDMA frame length.

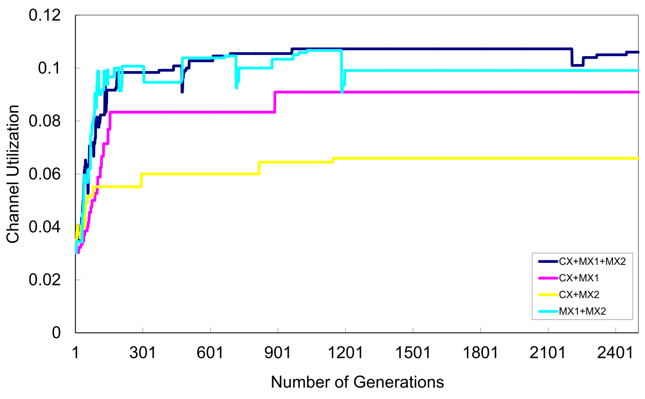


Fig. 10 Performance comparison of different genetic operators for the channel utilization.

Table 2 Experimental results with difference genetic operator combinations.

Genetic Operator Combination	Frame Length				Channel Utilization			
	Min.	Max.	Avg.	Std.	Min.	Max.	Avg.	Std.
CX+MX1+MX2	10	11	10.4	0.48	0.100	0.108	0.107	0.002
CX+MX1	10	11	10.7	0.47	0.090	0.100	0.094	0.004
CX+MX2	27	28	27.6	0.47	0.056	0.067	0.061	0.004
MX1+MX2	11	12	11.1	0.30	0.095	0.107	0.104	0.004

algorithm generates nearly consistent optimization results. For example, our algorithm produces the same frame length except for the fifth data instance. Also, the difference between the best and the worst utilization is less than 0.016. The results demonstrates the stability of our algorithm.

In Fig. 8 (a) and 8 (b), we show the results of broadcast scheduling for the fourth and fifth data sets. Each time slot is represented by one color. Nodes with multiple color labels transmit packets in the time slots corresponding to the color labels.

Next, we use different combinations of genetic operators to compare the performance of the TDMA frame length and the channel utilization. The first combination is (CX+MX1+MX2), which include all operators. The second and third combinations are (CX+MX1) and (CX+MX2), which combines the *Crossover* operator with *LengthMutation* and *UtilizationMutation*, respectively. In both combinations, one mutation operator is disabled by setting its probability to zero. The last combination, (MX1+MX2), only uses the mutation operators by setting the crossover ratio to zero.

In both Fig. 9 and 10, We can observe that the (CX+MX1+MX2) combination could achieve the best performance. The combination of (CX+MX2) is obviously worse than the other combinations with MX1 because of the long time frame. Although the *UtilizationMutation* operator could also reduce the time frame length by detecting the redundant time slots, the redundant time slots are rarely generated for large problem instances. Therefore, the *Utilization-Mutation* operator has relatively limited effect in shortening time frame length. As a result, the (CX+MX2) combination yields the worst channel utilization because of the long time frame. Using the *LengthMutation* operator can effectively obtain a shorter TDMA frame length and a higher channel utilization. This seems unreasonable for Fig. 10 since the *LengthMutation* operator aims at improving the time frame length. The reasoning is that the channel utilization is also affected by the time frame length. Therefore, the channel utilization can still be improved in the combinations without the *UtilizationMutation* operator. In Fig. 10, we can still observe the effect of the *UtilizationMutation* operator on improving the channel utilization. For instance, the combination of (MX1+MX2) has better channel utilization than another combination of (CX+MX1). The comparison between (MX1+MX2) and (CX+MX1) also shows that *Utilization-Mutation* can improve diversification since the former can still produce new results in later iterations. We further list the statistical results of 30 optimization trials for all combinations in Table 2. The results also suggest that the three heuristic genetic operators have different functions and must be used to get the best results.

We compare our genetic algorithm with the previous work in Table 3, where the first two data sets come from [6] and the last three come from [13]. For the previous algorithm in [6], its runtime for the first two instances is not available. In terms of the time frame length and channel utilization, the numerical results show that our algorithm has superior performance as compared to the previous algorithms. For the second and fifth problem instances, our algorithm outperforms the previous algorithm by reducing one time slot of the TDMA frame. For the first data set, channel utilization is increased by 0.017. Although the channel utilization is declined by about 0.003 for the third instances with our scheme, the average channel utilization is increased by more than 0.004.

Table 4 presents the experimental results for the randomly generated problem instances with various numbers of nodes from 25 to 2,500. For each network size, we generate three network topologies with different node degrees. The statistical results demonstrate that our genetic algorithm can provide consistent results regardless the size of problem instances. In the 30 optimization trials, the difference between the best and worst time frame lengths is always less than two. The maximal standard deviation for channel utilization is 0.15. Since channel utilization also relates to time frame length, a short time frame usually provides a relatively high channel utilization. We also observed that the data instances with lower degrees usually have higher standard deviations for channel utilization. Since more nodes can transmit simultaneously in a low-degree network, the variation of channel utilization is also increased. The runtime results also show that our genetic algorithm can effectively solve these problem instances with variable sizes. For example, our algorithm takes less than 2 minutes to produce the best result for the problem instance with 900 nodes. Even for larger problem instances, our algorithm still yields solutions with good efficiency. For the largest problem instances with 2,500 nodes, our algorithm could generate a TDMA frame whose length is equal to 13 time slots and the runtime is less than 25 minutes. The performance of both our genetic algorithm and ILS is similar for most results; however, our algorithm outperforms ILS for the large problem instances (1600 nodes and 2500 nodes) with higher degrees. Our algo-

Table 3 Performance comparison between our algorithm and the previous algorithms.

No. of	Max.	Our Genetic Algorithm		Previous Algorithms	
Nodes	Degree	Frame Length	Channel Utilization	Frame Length	Channel Utilization
15	7	8	0.167	8	0.150
30	8	10	0.116	11	0.112
40	7	8	0.200	8	0.203
100	8	9	0.148	9	0.148
100	8	10	0.108	11	0.104

Table 4 Experimental results with random data sets.

				Our Genetic Algorithm									ILS [15]		
Set	No. of	Avg.	Max.	Frame Length				Channel Utilization				Runtime	Frame	Channel	Runtime
No.	Nodes	Degree	Degree	Min.	Max.	Avg.	Std.	Min.	Max.	Avg.	Std.	(sec.)	Length	Utilization	(sec.)
1	25	4	7	8	8	8.0	0.00	0.135	0.140	0.139	0.002	0.08	8	0.140	0.01
2	25	5	8	9	9	9.0	0.00	0.124	0.124	0.124	0.000	0.09	9	0.124	0.03
3	100	4	8	9	9	9.0	0.00	0.118	0.164	0.153	0.012	1.80	9	0.154	0.61
4	100	5	8	10	10	10.0	0.00	0.105	0.132	0.126	0.007	3.24	10	0.128	1.81
5	100	6	8	10	11	10.5	0.50	0.100	0.110	0.109	0.003	4.39	10	0.107	5.43
6	400	4	8	10	10	10.0	0.00	0.106	0.146	0.136	0.012	4.39	10	0.145	8.37
7	400	5	8	11	12	11.0	0.17	0.100	0.118	0.114	0.007	16.84	11	0.117	36.17
8	400	6	8	12	13	12.1	0.29	0.086	0.106	0.100	0.004	13.25	12	0.101	15.79
9	900	4	8	10	11	10.1	0.24	0.101	0.145	0.131	0.012	91.18	10	0.145	70.76
10	900	5	8	11	12	11.6	0.49	0.091	0.112	0.108	0.010	174.10	11	0.116	70.06
11	900	6	8	12	13	12.6	0.50	0.084	0.093	0.086	0.006	162.11	13	0.098	53.71
12	1600	4	8	10	11	10.7	0.47	0.100	0.125	0.111	0.015	327.34	10	0.143	401.66
13	1600	5	8	11	12	11.9	0.33	0.091	0.116	0.111	0.007	178.44	12	0.113	394.44
14	1600	6	8	12	13	12.8	0.43	0.083	0.100	0.097	0.005	198.14	13	0.098	968.87
15	2500	4	8	10	11	10.9	0.33	0.100	0.122	0.109	0.012	960.52	10	0.145	2838.29
16	2500	5	8	12	12	12.0	0.00	0.098	0.118	0.114	0.004	953.95	12	0.115	1656.78
17	2500	6	8	13	13	13.0	0.17	0.087	0.100	0.098	0.003	460.16	13	0.095	3344.53

rithm can achieve better channel utilization with only 20% to 75% execution time of ILS. Since ILS is a trajectory-based method, it usually performs well for the small or low-network-degree problem instances. However, for the complex problem instances, it may not be capable of producing feasible solutions in a reasonable runtime due to its relatively weak diversification. The population-based genetic algorithm can avoid this problem of weak diversification. Therefore, our algorithm achieves better performance than ILS for the complex problem instances and shows superior scalability.

From our experimental results, we found that our crossover operator and two mutation operators can find a new solution within a few minutes for most problem instances. Our results also show that the proposed greedy genetic algorithm has the capability of producing new solutions even after several thousands generations. Since there is a higher chance for our genetic operators to attain an optimum solution, our algorithm only needs a small population with *ten* chromosomes to significantly reduce the computation cost. As a results, our greedy genetic algorithm is efficient and effective in solving the BSP, even for large problem instances with several thousands nodes.

6. Conclusions

In this paper, we propose a greedy genetic algorithm for the TDMA broadcast scheduling problem. In our genetic algorithm, we use a greedy crossover operator to guide the search space. We also design two greedy mutation operators to change the TDMA time slot and increase the channel utilization. With these genetic operators, we can leverage the balance between exploration and exploitation for searching solution space. Therefore, even with only a few chromosomes in each population, the proposed greedy genetic algorithm can still yield good solutions. The experimental results show that our algorithm is both efficient and effective in solving the BSP. As compared to previous algorithms, our algorithm can generate better TDMA broadcast schedule while keeping the computation time low.

References

[1] T.N.V. Cionca and V. Dadarlat, “TDMA protocol requirements for wireless sensor networks,” Second International Conference on Sensor Technologies and Applications, pp.30–35, Aug. 2008.

[2] S. Ramanathan, “A unified framework and algorithms for (T/F/C)DMA channel assignment in wireless networks,” Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies, pp.900–907, April 1997.

[3] S. Menon, “A sequential approach for optimal broadcast scheduling in packet radio networks,” IEEE Trans. Commun., vol.57, no.3, pp.764–770, March 2009.

[4] G. Wang and N. Ansari, “Optimal broadcast scheduling in packet radio networks using mean field annealing,” IEEE J. Sel. Areas Commun., vol.15, no.2, pp.250–260, Feb. 1997.

[5] A. Sen and E. Malesinska, “Approximation algorithms for radio network scheduling,” 35th Allerton Conference on Communication, Control and Computing, pp.573–582, Sept. 1997.

[6] S.K.J. Yeo and H. Lee, “An efficient broadcast scheduling algorithm for TDMA ad-hoc networks,” Computers & Operations Research, vol.29, no.13, pp.1793–1806, Nov. 2002.

[7] S.C. Ergen and P. Varaiya, “Tdma scheduling algorithms for wireless sensor networks,” Wirel. Netw., vol.16, pp.985–997, May 2010.

[8] I. Cidon and M. Sidi, “Distributed assignment algorithms for multihop packet radio networks,” IEEE Trans. Comput., vol.39, no.10, pp.1353–1361, Oct. 1989.

[9] A. Ephremides and T.V. Truong, “Scheduling broadcasts in multihop radio networks,” vol.38, no.4, pp.456–460, April 1990.

[10] N. Funabiki and Y. Takefuji, “A parallel algorithm for broadcast scheduling problems in packet radio networks,” IEEE Trans. Commun., vol.41, no.6, pp.828–831, 1993.

[11] I. Ahmad, B. Al-Kazemi, and A.S. Das, “An efficient algorithm to find broadcast schedule in ad hoc TDMA networks,” J. Comp. Sys., Netw., and Comm., vol.2008, pp.12:1–12:10, Jan. 2008.

[12] A. Capone and M. Trubian, “Channel assignment problem in cellular systems: a new model and a tabu search algorithm,” IEEE Trans. Veh. Technol., vol.48, no.4, pp.1252–1260, Aug. 1999.

[13] G. Chakraborty, “Genetic algorithm to solve optimum TDMA transmission schedule in broadcast packet radio networks,” IEEE Trans. Commun., vol.52, no.5, pp.765–777, May 2004.

[14] N.S. Goutam Chakraborty, Debasish Chakraborty, “A heuristic algorithm for optimum transmission schedule in broadcast packet radio networks,” Comput. Commun., vol.38, no.10, pp.74–85, 2005.

[15] C.C. Lin and P.C. Wang, “A new iterated local search algorithm for solving broadcast scheduling problems in packet radio networks,” EURASIP J. Wireless Communications and Networking, 2010.

[16] B. Liu and D. Towsley, “A study of the coverage of large-scale sensor networks,” First IEEE International Conference on Mobile Ad-hoc and Sensor Systems, pp.475–483, Oct. 2004.

[17] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual comparison,” ACM Comput. Surv., vol.35, no.3, pp.268–308, 2003.

[18] M. Gen and R. Cheng, Genetic Algorithms and Engineering Design, John Wiley & Sons, 1997.

Chih-Chang Lin is currently a Ph.D. candidate in the Department of Computer Science and Engineering at National Chung Hsing University. He is also working for Rising Star Technology. His research interests include the application of optimization techniques to network problems.

Pi-Chung Wang received the M.S. and Ph.D. degrees in Computer Science and Information Engineering from the National Chiao Tung University in 1997 and 2001, respectively. From 2002 to 2006, he was with Telecommunication Laboratories of Chunghwa Telecom, working on network planning in broadband access networks and PSTN migration. During these four years, he also worked on IP lookup and classification algorithms. Since February 2006, he has been an assistant professor of Computer Science at National Chung Hsing University. Wang's research interests include IP lookup and classification algorithms, scheduling algorithms, congestion control, network processors, algorithms and applications related computational geometry.