Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2010, Article ID 578370, 8 pages
doi:10.1155/2010/578370

Research Article

A New Iterated Local Search Algorithm for Solving Broadcast Scheduling Problems in Packet Radio Networks

Chih-Chiang Lin[1] and Pi-Chung Wang[1, 2]

[1] *Department of Computer Science and Engineering, National Chung Hsing University, Taichung 402, Taiwan*
[2] *Institute of Networking and Multimedia, National Chung Hsing University, Taichung 402, Taiwan*

Correspondence should be addressed to Pi-Chung Wang, pcwang.tw@gmail.com

Received 1 November 2009; Revised 7 April 2010; Accepted 25 April 2010

Academic Editor: Xinbing Wang

The broadcast scheduling problem (BSP) in packet radio networks is a well-known NP-complete combinatorial optimization problem. The broadcast scheduling avoids packet collisions by allowing only one node transmission in each collision domain of a time division multiple access (TDMA) network. It also improves the transmission utilization by assigning one transmission time slot to one or more nodes; thus, each node transmits at least once in each time frame. An optimum transmission schedule could minimize the length of a time frame while minimizing the number of idle nodes. In this paper, we propose a new iterated local search (ILS) algorithm that consists of two special perturbation and local search operators to solve the BSPs. Computational experiments are applied to benchmark data sets and randomly generated problem instances. The experimental results show that our ILS approach is effective in solving the problems with only a few runtimes, even for very large networks with 2,500 nodes.

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 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. By assigning more than one transmitting nodes in one time slot, the utilization of the shared channel could be promoted. The aim of the broadcast scheduling problem (BSP) is to schedule the node transmissions with a minimum frame length while maximizing utilization of the shared channel.

The BSP is a combinatorial optimization problem known to be NP-complete [2]. Several scheduling algorithms have been proposed on exact, heuristic, and metaheuristic techniques. Exact techniques for the BSP (e.g., *brand-and-bound*) can optimally solve smaller problem instances. However, for complex problem instances, the computation of exact techniques can be quite time and memory consuming. Many heuristics, such as mean field annealing [2], neural network [3], and graph theory [4], have been proposed to solve the BSP. Several heuristic approaches for this problem that only considers the criterion of minimizing time frames, do not include the criterion of maximizing channel utilization. Although, Yeo et al. [1] presented a two-phase algorithm which could minimize time frames and maximize channel utilization simultaneously, but its performance of solving large-scale BSP instances is unknown.

Currently, many applications of large-scale wireless sensor networks have been developed, for example, battlefield surveillance and environment monitoring [5, 6]. An efficient solution for large-scale BSP instances is thus getting important. Recently, metaheuristic algorithms, such as genetic algorithms [7], are widely recognized by their success in solving the large-scale BSP. A metaheuristic is a heuristic method for solving general computational problems by combining problem-specific optimization techniques to obtain a more efficient or more robust procedure. Although some of these metaheuristic methods show excellent results for the BSP, they require large computation efforts. Iterated local search (ILS) is a simple and powerful stochastic local search method which is robust, highly effective, and easy to implement [8]. It

has been successfully applied to solve the *traveling salesman problem* [9], the *quadratic assignment problem* [10] and the *permutation flowshop problem* [11].

In this paper, we propose an efficient ILS algorithm for the BSP to minimize the length of TDMA time frame while maximizing the number of transmitting nodes in a time frame. Our algorithm is effective and efficient regardless of the network sizes. Our experiments show that the new scheme can solve large problem instances with 2,500 nodes. The numerical results have also demonstrated that our scheme not only optimizes broadcast scheduling but also significantly reduces computation time.

The organization of this paper is as follows. The next section describes the broadcast scheduling problem in a TDMA packet radio network. In Section 3, we describe the ILS algorithm. Section 4 presents our proposed ILS algorithm for the BSP. Performance analysis is given in Section 5. Conclusions are presented in Section 6.

2. The Broadcast Scheduling Problem

The broadcast scheduling problem can be represented by a graph, $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ denotes the set of network nodes, and $E = \{e_1, e_2, \ldots, e_n\}$ denotes the set of transmission links. If there exists an edge between two nodes i and j in V, the two nodes i and j are one-hop apart and can receive the transmitted packet from each other. If they transmit packets at the same 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 the same time slot, a hidden terminal collision occurs.

The BSP aims at finding a TDMA time frame, where each node cannot transmit and receive packets or receive more than one packet in the same time slot, to avoid direct and hidden terminal collision. 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 assigning one time slot to each node; however, such solution may not be suitable for a large-scale network. To improve the efficiency, a broadcast scheduling should maximize the number of transmitted 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). Therefore, 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 by defining the maximum degree of a vertex in V and denoting this as D. The tight lower bound for a frame length is equal to $D + 1$ [1]. Therefore, there are at least $2^{N(D+1)}$ 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. Thus, the BSP is an NP-complete combinatorial optimization problem, and there is no algorithm to find the optimal solution in a polynomial time. We need a heuristic algorithm to find a suboptimal solution.

```
PROCEDURE IteratedLocalSearch()
s0 ← GenerateInitialSolution()
s ← LocalSearch(s0)
WHILE termination conditions not met DO
   s′ ← Perturbation(s)
   s″ ← LocalSearch(s′)
   s ← AcceptanceCriterion(s, s″)
END WHILE
```

Algorithm 1: Pseudocode of an iterated local search algorithm.

3. The Framework of an Iterated Local Search Algorithm

ILS is an effective metaheuristic to solve the combinatorial optimization problem. It is also simple and easy to implement [8]. It consists of four basic operators. The first operator "*GenerateInitialSolution*" generates an initial solution s_0. The second operator "*Perturbation*" reconstructs the current solution s and results in some intermediate solution s'. The third operator "*LocalSearch*" improves s' to another local optimum solution s''. The last operator "*AcceptanceCriterion*" decides which solution is applied for the next perturbation step. The termination conditions could be a maximum number of iterations, maximum runtime, or the maximum number of iterations without improvement. Algorithm 1 shows the pseudocode of an ILS algorithm.

Search intensification and diversification are two key strategies that determine the behavior of a metaheuristic [12]. Whereas search intensification applies a local search operator until a local optimum is reached, search diversification utilizes the perturbation operator to leave the current local optimum and explore different solution spaces. The acceptance criterion operator is between intensification and diversification of the search. When the best found solutions are always applied to the perturbation for promoting search intensification, accepting new solutions with a predefined probability usually improves search diversification. Both search intensification and diversification are considered in the operators of our ILS algorithm.

4. Our Iterated Local Search Operators

In this section, we introduce the operators of our iterated local search approach based on the framework of the ILS algorithm. Our initial operator generates an initial solution by using random permutation of n nodes and the *next-fit* algorithm. We first create a random permutation π. Next, we add a node π_i to the first time slot S of an empty broadcast scheduling time frame F, where π_i is the ith nodes in the permuted node list. If the node insertion for time slot S does not cause any collisions between node transmissions, then we store the current result in the temporary schedule S' and

```
Variables and functions
F: The TDMA time frame
S and S′: The Time slot
n: The number of nodes
π: Fill permutation of nodes
AddNode(i): Add node i to S
Check(S): Boolean function for checking that the time slot S
is not interference, if no interference then retrun true,
otherwise false.
AddTimeSlot(S): Add time slot S to TDMA time frame F
PROCEDURE GenerateInitialSolution()
F ← φ, S ← φ, S′ ← φ
π ← RandomGeneratePermutation()
FOR i ← 1 TO n
  S ← AddNode(π_i)
  IF Check(S) is not interference THEN
    S′ ←S
  ELSE
    F ← AddTimeSlot(S′)
    S← φ
    S ← AddNode(π_i)
  END IF
NEXT i
F ← AddTimeSlot(S)
RETURN F
```

Algorithm 2: Pseudocode of the proposed operator of initial solution generation.

insert next node π_{i+1} to S. Otherwise, we add a new time slot S to the frame and repeat the above steps. This procedure is repeated until all nodes are inserted into the broadcast schedule F. The pseudocode of the proposed initial operator is listed as follows.

The perturbation operator attempts to produce new solutions by using a three-phase operation, which includes reducing, increasing or maintaining a time slot. The reducing phase is applied to F by randomly removing a time slot S. The reason for removing the time slot S is to reduce the frame length of F. As a result, there might be remaining nodes that have not been scheduled after moving the time slot S. These unscheduled nodes are then inserted into a temporary time slot S'. Next, the increasing phase attempts to increase channel utilization by inserting the nodes of the time slot S into a randomly selected time slot without causing any collisions. If a successfully inserted node also exists in the time slot S', then it is removed from S'. Finally, the remaining phase considers the unscheduled nodes in the time slot S'. If S' is not empty, then S' is appended to the time frame F. The perturbation procedure is listed in Algorithm 3.

The local search operator uses a straightforward approach to further reduce the time frame length by merging the redundant transmitting nodes from different time slots. In the operation of the proposed perturbation operator, the increasing phase improves channel utilization by increasing the number of transmitting nodes, thereby resulting in some redundant time slots in F. Therefore, we compare any two time slots in F. If the transmitting nodes in one time slot is

```
Variables and functions
V: The set of all nodes, where |V| denotes the
number of all nodes.
S: The nodes in time slot S, where |S| denotes the
number of nodes in slot S.
CollectNodes(F): Collect nodes in the TDMA
time frame F.
GetNode(S, i): Get i_th node in time slot S.
RemoveNode(i): Remove i_th node in time slot S.
AddRemainNode(S): Add nodes in the time slot S to
TDMA time frame F.
PROCEDURE Perturbation(F)
//reducing phase:
S ← RandomRemoveOneOfTimeSlots(F)
N ← CollectNodes(F)
S′ ← φ
FOR i ← 1 TO |S|
  n ← GetNode(S, i)
  IF n ∉ N THEN
    S′ ← AddNode(n)
  END IF
NEXT i
//increasing phase:
FOR i ← 1 TO |V|
  n ← GetNode(V, i)
  IF n ∉ S′′ THEN
    S′′ ← AddNode(n)
    IF Check(S′′) is interference THEN
      S′′ ← RemoveNode(n)
    ELSE
      IF n ∈ S′ THEN
        S′ ← RemoveNode(n)
      END IF
    END IF
  END IF
NEXT i
//remaining phase:
IF S′ does not equal φ THEN
  F ← AddRemainNode(S′)
END IF
```

Algorithm 3: Pseudocode of our perturbation operator.

equivalent to or a subset of those in another time slot, then the time slot with more transmitting nodes is reserved and the other time slot is removed. In the local search operator, we attempt to minimize TDMA time frame length first. In other words, the local search operator always accepts a new solution with a shorter time frame. In the case that a new time frame has the same length as previous time frame, the new time frame is accepted only when the channel utilization is improved. The procedure of our local search is shown in Algorithm 4.

In our algorithm, we use the termination condition that limits the maximum number of iterations. Our perturbation operator can improve the search diversification by generating new solutions. Our local search operator further reduces the length of a time frame to improve the search intensification. Moreover, the acceptance criterion of new solutions in the

Variables and functions
F^*: The improvement TDMA time frame.
$|F|$: The number of time slots in TDMA time frame F.
GetTimeSlot(i): Get i_{th} time slot in TDMA time frame F.
ReomveTimeSlot(i): Remove i_{th} time slot in TDMA time frame F.
ReplaceTimeSlot(i, S): Replace i_{th} time slot by time slot S.
Utilization(F): Calculation of channel utilization in TDMA time frame F.

```
PROCEDURE LocalSearch(F)
F* ←F
has Improvement ← false
FOR i ← 1 TO |F|
  S_i ← GetTimeSlot(F, i)
  FOR j ← i + 1 TO |F|
    S_j ← GetTimeSlot(F, j)
    S' ← S_i ∪ S_j
    IF Check(S') is not interference THEN
      F' ←F
      F' → ReomveTimeSlot(j)
      F' → ReplaceTimeSlot(i, S')
      IF |F*| > |F'| THEN
        F* ← F'
        has Improvement ← true
      END IF
      IF Utilization(F*) < Utilization(F') THEN
        F* ← F'
        has Improvement ← true
      END IF
    END IF
  NEXT j
NEXT i
IF has Improvement equal true THEN
  RETURN F*
END IF
RETURN F
```

Algorithm 4: Pseudocode of our local search operator.

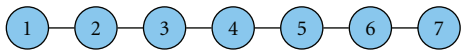


Figure 1: A seven-node network.

local search operator can achieve better search diversification. As a result, our ILS algorithm considers the search intensification and diversification simultaneously.

We use an example in Figure 1 to illustrate the operations of our algorithm. First, we generate the initial solution by using the proposed operator. First, we randomly generate a node permutation, $\pi = \{3, 6, 2, 4, 7, 1, 5\}$. Next, we create the first time slot, S_1, and insert node 3 into S_1. The node 6 is then inserted into S_1 since the signals from node 3 and node 6 do not collide with each other. However, the third node, node 2, may cause signal collision with node 3. Therefore, node 2 is inserted into a new time slot, S_2. Node 4 is also inserted into a new time slot since it may cause signal collision with node 2. We repeat the operation until all nodes have been inserted. Finally, we get an initial solution, $F = \{3, 6\}\{2\}\{4, 7\}\{1, 5\}$.

After generating the initial solution, we use the perturbation operator to generate new solutions. In the first step, we randomly remove the fourth time slot, $S' = \{1, 5\}$, from F. Since nodes in S' do not appear in the rest time slots in F, both nodes 1 and 5 are kept in S'. The second step of perturbation operator randomly select the third time slot $S'' = \{4, 7\}$ for inserting nodes from S'. Since node 1 can be inserted into S'' without causing collision, it is removed from S'. There is only one remaining node in S', node 5. S' is then inserted into F and the new solution is $F = \{3, 6\}\{2\}\{1, 4, 7\}\{5\}$.

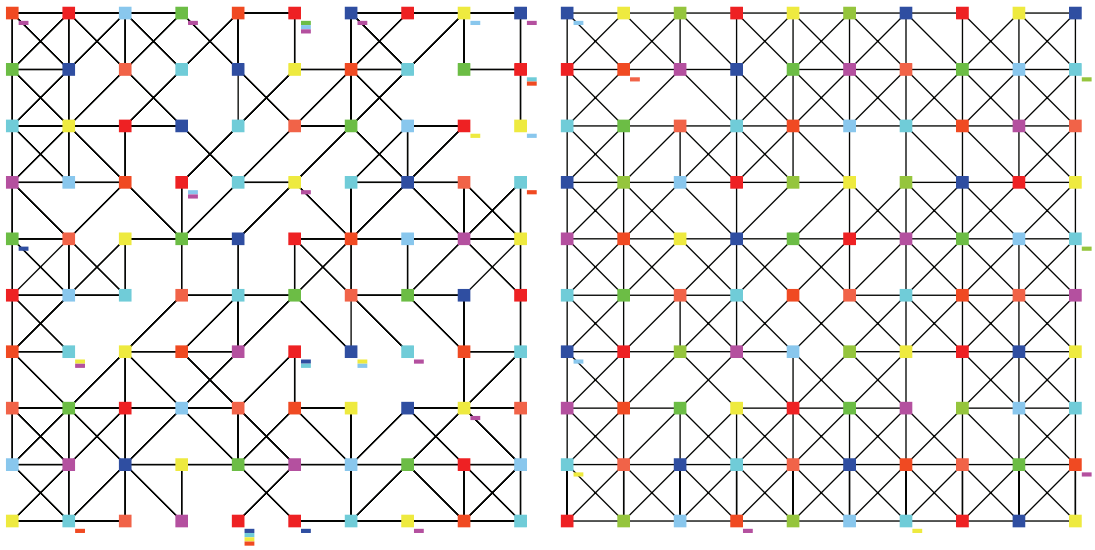
In the third operator of local search, we determine which two time slots can be merged. In the previous solution, we notice that the second and the fourth time slots can be merged without causing collision. Therefore, we can derive another new solution, $F = \{3, 6\}\{2, 5\}\{1, 4, 7\}$. The above three operators are repeated by a predefined number of iterations.

5. Performance Analysis

In this section, we present the experimental results of applying the proposed ILS algorithm to the BSP. The experiments were performed by using five benchmark data sets and randomly generated problem instances. The benchmark data sets come from [1, 7], where these data sets contain 15, 30, 40, and 100 nodes with different connectivity degrees. We show two of these data sets with 100 nodes in Figure 2. We also use the method in [7] to randomly generate problem instances whose network sizes vary among 25, 100, 400, 900, 1,600, and 2,500. In addition, the average connectivity degree of nodes for each network changes among 4, 5, and 6. For each problem instance, we repeat our algorithm with 15,000 iterations. The performance metrics include TDMA frame length, channel utilization and runtime. We also show the number of iterations to generate the best results. The experiments were done on a modern INTEL processor with a single core (1.83 GHz). The source code for our ILS algorithm was written in Microsoft Visual C + +.

We conduct our experiments from the five benchmark data sets. Table 1 lists the results for the benchmark data sets along with their properties. As shown in Table 1, the proposed ILS algorithm could derive a TDMA frame whose length is less than or equal to ten time slots. The channel utilization varies from 0.109 to 0.2 for different data sets. Our ILS algorithm derives the best results with several thousand iterations, except for the data set with 30 nodes, where only 107 iterations are required. The runtime for these data sets is less than 2.7 seconds, which is almost directly proportional to the number of nodes. In Figure 2, we show the results of broadcast scheduling for the fourth and fifth data sets. Each time slot is indicated with a color. Nodes with multiple color labels transmit packets in the time slots corresponding to the color labels.

We also compare our ILS algorithm with the previous work in Table 2, where the first two data sets are from [1] and the last three are from [7]. For the previous algorithm in [1], its runtime for the first two data sets is not available. In terms



(a) Data set no. 4 (b) Data set no. 5

Figure 2: The broadcast scheduling of two data sets in Table 1 with our ILS algorithm.

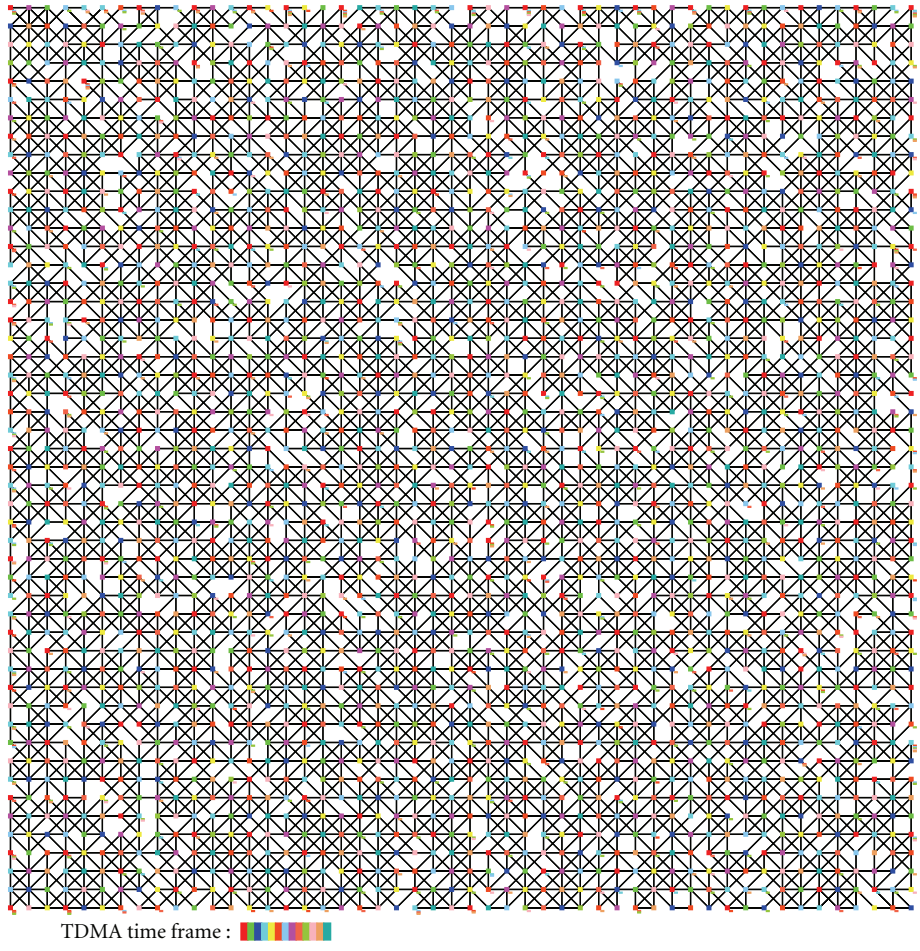


Figure 3: The broadcast scheduling for the seventeenth instance in Table 3 with our ILS algorithm.

Table 1: Experimental Results with Benchmark Data Sets.

Data Set No.	No. of Nodes	Avg. Degree	Max. Degree	TDMA Frame Length	Channel Utilization	Required no. of Iterations	Runtime (sec.)
1	15	3.8	7	8	0.166	1,452	0.26
2	30	4.6	8	10	0.110	107	0.06
3	40	3.3	7	8	0.200	4,783	1.26
4	100	4	8	9	0.147	3,723	2.69
5	100	6	8	10	0.109	2,434	2.56

Table 2: Performance Comparison between Our Algorithm and the Previous Algorithms.

Number of Nodes	Max.Degree	Our Genetic Algorithm			Previous Algorithms [1, 7]		
		TDMA Frame Length	Channel Utilization	Runtime (sec.)	TDMA Frame Length	Channel Utilization	Runtime (sec.)
15	7	8	0.166	0.26	8	0.150	—
30	8	10	0.110	0.06	11	0.112	—
40	7	8	0.200	1.26	8	0.203	9.5
100	8	9	0.147	2.69	9	0.148	270
100	8	10	0.109	2.56	11	0.104	270

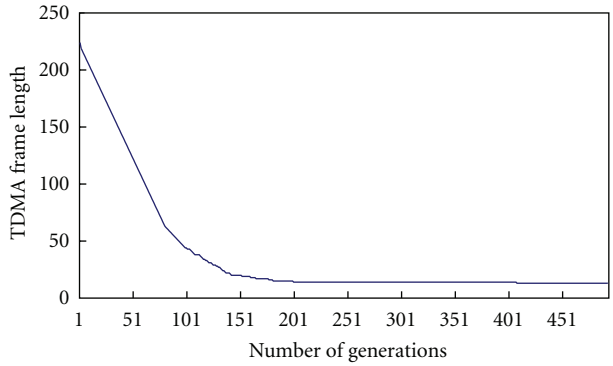


Figure 4: The TDMA frame length for the seventeenth instance in Table 3.

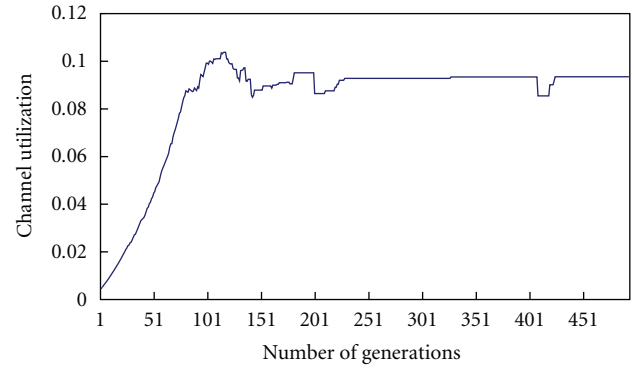


Figure 5: Channel utilization for the seventeenth instance in Table 3.

of the time frame length, the numerical results show that our algorithm has comparable performance with the previous algorithm for the first, third and fourth data sets. For the rest two data sets, our algorithm outperforms the previous algorithm by reducing one time slot of the TDMA frame. While considering the performance of channel utilization, our algorithm has slightly worse performance for the second to fourth data sets, declining by about 2%. However, for the first data set, channel utilization is improved by more than 10%. For the fifth data set, channel utilization is also improved by more than 4%. Table 2 also shows that our algorithm requires shorter computation time than the previous algorithm in [7]. For the previous algorithm, the computation cost would severely increase as the number of nodes increases from 40 to 100, whereas our algorithm only incurs a moderate increase of runtime. For the data sets with 100 nodes, our algorithm requires only about one percent of the computation time of the previous algorithm. Even when we consider the difference between CPU clock rates (a 600 MHz CPU is used for the experiments in [7]), our algorithm is still thirtyfold faster than the previous algorithm. Therefore, our algorithm has the advantage of low computation cost.

Table 3 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 numerical results show that our ILS algorithm could effectively solve these problem instances with variable sizes. For example, our algorithm takes less than 37 seconds to derive the result for the problem instance with 400 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

Table 3: Simulation Results with Randomly Generated Problem Instances.

Data Set No.	No. of Nodes	Avg. Degree	Max. Degree	TDMA Frame Length	Channel Utilization	Required no. of Iterations	Runtime (sec.)
1	25	4	7	8	0.140	21	0.01
2	25	5	8	9	0.124	82	0.03
3	100	4	8	9	0.154	853	0.61
4	100	5	8	10	0.128	2,052	1.81
5	100	6	8	10	0.107	6,918	5.43
6	400	4	8	10	0.145	1,230	8.37
7	400	5	8	11	0.117	6,270	36.17
8	400	6	8	12	0.101	2,078	15.79
9	900	4	8	10	0.145	1,953	70.76
10	900	5	8	11	0.116	1,276	70.06
11	900	6	8	13	0.098	416	53.71
12	1,600	4	8	10	0.143	2,701	401.66
13	1,600	5	8	12	0.113	1,525	394.44
14	1,600	6	8	13	0.098	9,157	968.87
15	2,500	4	8	10	0.145	10,907	2,838.29
16	2,500	5	8	12	0.115	3,592	1,656.78
17	2,500	6	8	13	0.095	10,313	3,344.53

a TDMA frame whose length is less than or equal to 12 time slots while the runtime is less than an hour. For the problem instance no. 17, we show the result of broadcast scheduling in Figure 3 with thirteen different colors. We also show the optimization progress for the TDMA frame length and channel utilization in Figures 4 and 5, respectively. Figure 4 shows that our ILS algorithm could significantly reduce the length of the generated time frame in the first two hundred iterations. In the subsequent iterations, the ILS algorithm can further decrease the frame length and then improve channel utilization. This phenomenon can be observed in the 200 $\sim$ 250 and 400 $\sim$ 450 iterations in Figure 5. Our ILS algorithm optimizes the frame length first. Therefore, when the frame length is decreased by one, the following iterations then gradually increase channel utilization.

From our experimental results, we found that the perturbation operator can find a new solution from a few iterations and the local search operator can directly remove redundant time slots. Therefore, it seems reasonable to conclude that there is a higher chance for our perturbation and local search operators to attain an optimum solution. As demonstrated in our experimental results, our ILS algorithm is efficient and effective in solving the BSP.

6. Conclusions

In this paper, we propose a new iterated local search algorithm for the broadcast scheduling problem in a TDMA packet radio network. Our algorithm solves the combinatorial optimization problem by combining two strategies, namely, search diversification and intensification. A perturbation operator utilizes the first strategy and a local search operator uses the second. We also use an acceptance criterion for new solutions to improve search diversification. The experimental results show that our algorithm is both efficient and effective in solving the BSP. As compared to the previous algorithm, our algorithm can derive better TDMA time frame while keeping the computation time low.

Acknowledgment

This work is supported in part by the National Science Council under Grant no. NSC 98-2218-E-241-004.

References

[1] J. Yeo, H. Lee, and S. Kim, "An efficient broadcast scheduling algorithm for TDMA ad-hoc networks," *Computers and Operations Research*, vol. 29, no. 13, pp. 1793–1806, 2002.

[2] G. Wang and N. Ansari, "Optimal broadcast scheduling in packet radio networks using mean field annealing," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 2, pp. 250–260, 1997.

[3] L. Wang and H. Shi, "A gradual noisy chaotic neural network for solving the broadcast scheduling problem in packet radio networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 989–1000, 2006.

[4] S. C. Ergen and P. Varaiya, "TDMA scheduling algorithms for wireless sensor networks," *Wireless Networks*, vol. 16, no. 4, pp. 985–997, 2010.

[5] B. Liu and D. Towsley, "A study of the coverage of large-scale sensor networks," in *Proceedings of the 1st IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, pp. 475–483, October 2004.

[6] S.-K. Chen and P.-C. Wang, "An anycast-based emergency service for healthcare wireless sensor networks," *IEICE Transactions on Communications*, vol. E93-B, no. 4, pp. 858–861, 2010.

[7] G. Chakraborty, "Genetic algorithm to solve optimum TDMA transmission schedule in broadcast packet radio networks," *IEEE Transactions on Communications*, vol. 52, no. 5, pp. 765–777, 2004.

[8] H. Ramalhinho-Lourenco, O. C. Martin, and T. Stutzle, *Iterated Local Search*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[9] K. Katayama and H. Narihisa, "New iterated local search algorithm using genetic crossover for the traveling salesman problem," in *Proceedings of the 14th ACM Symposium on Applied Computing (SAC '99)*, pp. 302–306, March 1999.

[10] T. Stützle, "Iterated local search for the quadratic assignment problem," *European Journal of Operational Research*, vol. 174, no. 3, pp. 1519–1539, 2006.

[11] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.

[12] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.