CHAPTER 3

MEMETIC ALGORITHM FOR BROADCAST SCHEDULING PROBLEM

3.1 INTRODUCTION

In literature, the broadcast scheduling problem was solved using many approaches having fitness functions as minimum number of time slots and maximum number of transmissions. Major drawbacks of these approaches are the large number of iterations and huge computation time. In the previous chapter, the advantage of immune genetic algorithm for wireless multi-hop network is compared with standard genetic algorithm. Even though IGA performs well compared to other algorithms the execution time is not reduced, which is an important factor to validate an algorithm. Hence, it is desirable to propose a new algorithm to reduce the number of iterations and the computation time. The objective of this chapter is to reduce the time slots and to maximize the total number of transmissions in an acceptable execution time.

The task of a broadcast scheduling algorithm is to produce and maintain a schedule of TDMA slots such that each station is periodically assigned a slot for transmission and all transmissions are received without collision. Most broadcast scheduling algorithms operate by producing a finite length TDMA schedule in which each station is assigned exactly one slot for transmission and then indefinitely repeating that schedule. In order to increase the channel utilization within the identified TDMA schedule the total number
of transmission has to be increased, i.e., the number of time slots has to be increased. The WMN scheduler determines a collision free schedule with minimum TDMA frame length and maximum slot utilization by the nodes in an acceptable running time. The scheduler assumes that each MS has network connectivity information within a two-hop radius.

Figure 3.1 represents a simple wireless multi-hop network. Each node represents a mobile station and a line connecting two nodes indicates that the two MSs are within the communication range. The neighbours of A are those MSs that can communicate directly with A (i.e., B and C). Node mobility in WMN causes frequent changes in the network topology. The main difficulty in designing WMN is that not all MSs can communicate directly with each other.

![Figure 3.1 A simple wireless multi-hop network](image)

From Figure 3.1 the values of \( N = \{A, B, C, D, E, F, G, H\} \) and \(|N| = 8\), where \(|N|\) represent the number of nodes in the given network, i.e., \( N = \{n_1, n_2, \ldots, n_8\} \) and \(|M|\) is the number of time slots. The connectivity matrix \([CM]\), hop matrix \([HM]\) and scheduler matrix \([SM]\) for the network given in Figure 3.1 is identified as follows:
In scheduler matrix, the row represents the number of time slots. It takes the value 0 or 1, where 1 represents the node allowed for transmitting in that time slot. In first time slot, nodes A and E are allowed to send their packets without interference.

Three evolutionary algorithms namely, genetic algorithm, immune genetic algorithm, and memetic algorithm are reported in this chapter to solve broadcast scheduling for TDMA in WMN. The aim is to minimize the TDMA cycle length and to maximize the node transmissions with reduced computation time. In comparison to GA and IGA, MA actively aim at improving the solutions and is explicitly concerned with exploiting all available knowledge about the problem. A WMN scheduler with the three evolutionary algorithms are carried out and a series of simulations is conducted to evaluate the performance of the proposed MA in terms of solution quality, running time and to verify its superiority over GA and IGA.
3.2 MEMETIC ALGORITHM

A hybrid population based approach known as memetic algorithm, introduced in late 1980s, combines the recognized strength of population methods with the intensification capability of a local search. It form a type of metaheuristic that balances exploration and exploitation to find high quality solutions to the given optimization problem. Exploration is attained by means of the population approach together with the use of recombination and mutation operators. Exploitation comes with the use of individual improvement procedures, usually efficient local search methods (Banos et al 2010, Krasnogor and Gustafson 2002, Moscato 1999, Lu and Hao 2010).

Memetic algorithms are evolutionary algorithms that apply a separate local search process to refine individuals, i.e., improve their fitness. A particular feature of MA is greatly responsible for its success, unlike traditional evolutionary computation methods, MAs are intrinsically concerned with exploiting all available knowledge about the problem under study. This is not an optional mechanism but a fundamental feature, which is something that was neglected in evolutionary algorithms for a long time. Despite the good results obtained by some MAs, the process of designing efficient MAs often depends on the problem specific details.

MAs are inspired by Richard Dawkin’s concept of a meme, which represents a unit of cultural evolution that can exhibit local refinement. The characterization of a meme suggests that in cultural evolution processes, information is not simply transmitting unaltered between individuals. Instead, it is processed and enhanced by the communicating parts. This property is accomplished in MA by incorporating heuristics, approximate algorithms,
local search techniques, specialized recombination operators, truncated exact methods, etc. Most MAs can be regarded as a search strategy in which a population of optimizing agents cooperate and compete with each other.

The basic structure of MA is composed of four main methods:

1. Initialization method – generates a set of diverse initial solutions.
2. Reproduction method – creates a set of children from the agents of the main population or parents.
3. Combination method – generates populations to the next iteration with the best solutions of the parents and children.
4. Local optimization method – improves the current agents.

3.3 EVOLUTIONARY ALGORITHMS

Evolutionary algorithms can be applied to any problem that can be formulated as a function optimization task. It requires a set of solutions as input, a fitness function to evaluate solutions and variation operators to generate new solutions from old solutions. It is always reasonable to incorporate domain specific knowledge into an algorithm when working with particular real world problems. Specialized algorithms can outperform unspecialized algorithms on a restricted domain of interest.

Evolutionary algorithms offer a framework such that it is easy to incorporate such knowledge. Specific variation operators may be known to be useful when applied to particular representations. They can be directly applied as mutation or recombination operations. Incorporating such knowledge
focuses the evolutionary search, yielding a more efficient exploration of the state space of possible solutions. The selection, crossover, mutation, fitness function and termination condition discussed below is common for all three algorithms.

3.3.1 General Representations for GA, IGA and MA

3.3.1.1 Representation of chromosome

The TDMA scheduler matrix is a $M \times N$ matrix, where $M$ is the number of time slots and $N$ is the total number of nodes in the network. The scheduler matrix is represented as a bit string chromosome containing 0s and 1s. Each row and column of the scheduler matrix represents a time slot and node transmission. The value 1 in the position $(i, j)$ in the matrix indicates that $j^{th}$ node is allowed for transmission in the $i^{th}$ time slot.

3.3.1.2 Initial chromosomes

The initial TDMA frames are constructed using the elite population method of Chakraborty (2004). GA manipulates a set of chromosomes to search for an optimal solution. The making of initial TDMA frames based on elite population method is explained using the eight-node example of Figure 3.1. For $N = 8$, let us take a permutation sequence E, A, F, C, G, D, B, H to create an initial TDMA frame. The idea is to allocate time slots for different nodes in this permutation sequence. Here, the first node to assign is E so time slot 1 is allotted to node E, i.e., node E transmits at time slot 1. After consulting the hop matrix, it is known that nodes C, D, F, G and H cannot transmit simultaneously with node E. To indicate this * symbol is inserted into column C, D, F, G and H in the first time slot. It specifies that
when node E is transmitting, the nodes C, D, F, G and H are not allowed to transmit since they might lead to either primary or secondary interference. Next, the time slot allocation is for node A, it is evident that nodes A or B can still transmit in time slot 1 without making interference to node E, so the node A is also allocated in the time slot 1. When node A is allowed for transmission, the nodes B, C and D are not allowed. Since the nodes C and D are already marked, the * symbol is inserted to node B in the time slot 1. Time slot 1 is now full, i.e., in the first time slot the nodes A and E are allowed for transmission. Similarly, the nodes in the permutation sequence are allocated in the TDMA frame. Depending on the permutation sequence, the length of the TDMA frames will be different.

3.3.1.3 Evaluation of chromosome

The fitness function evaluates the quality (fitness) of candidate solutions. The fitness function for the scheduling problem is based on the variable channel utilization and tight lower bound. The termination point determines whether the best feasible solution is identified in that generation or not. The best feasible solution is the one, which satisfies all the criteria. When the generation of evolution reaches this termination point, the algorithm stops and outputs the optimal solution for the given network.

3.3.1.4 Selection operator for GA, IGA and MA

Choosing parameters in genetic algorithm might result in very different results. A good pattern of parameters might cause the algorithm to converge towards best results in a short time while a worse pattern might cause the algorithm to run for a long time before finding a good solution or even it might never be able to find a good solution.
The algorithms perform \( k \)-tournament selection for parent selection, choose the winner among \( k \) individuals that are drawn randomly from the population. The number \( k \) controls selection pressure, a higher \( k \) gives higher selection pressure.

Another significant parameter to enhance the evolutionary algorithm is the survivor selection pressure, which is the process of selecting the best individuals for the next generation. If it is too low, then the rate of convergence towards the optimum solution is too low. If the selection pressure is too high, the algorithm will likely to be stuck in a local optimum due to the loss of diversity in the population. Hence, the survivor selection method controls the selection pressure, which in turn determines how fast the algorithms converge. The survivor selection mechanism should be chosen such that it converges to the global optimum solution by without being caught into local optimum and should encompass the knowledge of the existing data.

In GA, IGA and MA, a small percentage of best fitness individuals are retained in the next generation. It increases the performance of algorithm, by preventing the loss of best found solution. From each generation 10\% of the best solution is retained in the next iteration.

### 3.3.1.5 Crossover and mutation

The selected chromosomes for reproduction are gathered in the mating pool. After initializing the population, the selection operator picks two chromosomes from the population to serve as parent. The crossover operator then exchanges the information between these two parents to produce their offspring. A predetermined crossover rate defines the probability of performing crossover.
The single-point crossover operator is done on the rows of the population. Once a crossover point is identified, a random row from the first parent PR1 is crossed over with a random row from the second parent PR2. The resultant chromosome CH1 is replaced with PR1 and CH2 is replaced with PR2. After replacing, if the solution violates the constraints then it is penalized. Mutation is performed with a probability, called mutation rate, to alter slightly some genes in the offspring. The mutation operator behaves in a different manner depending on the fitness of the selected gene. The mutation operator changes one bit in the selected chromosome depending on the individual fitness.

3.3.1.6 Elitism

The idea of elitism is to retain some of the best individuals in each generation. In this algorithm, a small percentage of best fitness is retained in the next generation. It increases the performance of algorithm, because it prevents losing the best found solution. From each generation 10% of the best solution is retained in the next iteration.

3.3.1.7 Termination Condition

The purpose of the termination point is to determine if the algorithm has reached the terminal condition. The terminal condition determines whether the best feasible solution is identified in that generation or not. The terminal condition is the channel utilization variable and the tight lower bound. When the generation of evolution reaches this termination condition, the algorithm stops and outputs the optimal solution for the given network. While working out traditional genetic operations, it is possible that the offspring created violates the constraints. The following steps are taken to
penalize those infeasible solutions. (i) After the completion of genetic operation, each population generated is validated to identify whether it has primary and secondary conflict. (ii) If a population did not fulfill the condition, it is dropped in that generation. (iii) Otherwise, it will be considered for next generation based on their fitness function.

    Genetic algorithms solve many search and optimization problems, effectively. However, they may drop into local optimal solutions or they may find the optimal solution by low convergence speed and GA blindly wanders over the search space. To overcome these problems, immune concept is used to enhance the GA. IGA gets the knowledge from hop matrix during vaccination process. IGA increases the number of transmission in a reduced time slot but not in a good computation time, and MA reduces the processing time. Memetic algorithm is a blooming dialect of evolutionary algorithm (EA). In addition to Darwinism, MA adopts the Lamarckian theory that the offspring can inherit the knowledge or characteristics that their parents acquire during their life time. MA implements this idea by integrating a local enhancement, such as local search and repair operator, into the canonical EA and making the enhancement inheritable. This integration significantly improves the exploitation ability of EA. In genetic algorithm, the mutation creates new genes for the population and the crossover operator orients seeking the best solution from the genes in the population. In memetic algorithm, this orientation is achieved by local search. Local search reduces the search space and reaches high quality solution faster. MA actively aims at improving the solution and is explicitly concerned with exploiting all available knowledge about the problem.
3.3.2 Genetic Algorithm

After initializing the population, the selection operator picks two chromosomes from the population to serve as a parent. The crossover operator then exchanges the information between these two parents to produce their offspring. A predetermined crossover rate defines the probability of performing crossover. Mutation is performed with a probability, called mutation rate, to alter slightly some genes in the offspring. Algorithm 1 presents the framework of genetic algorithm. The generated populations are evaluated with the fitness conditions. If the optimal solution is identified in the generation, then the algorithm is terminated with the solution. Otherwise, elitism method is done on the populations and proceeds to the next generation. At the end of iteration, the populations produced in the generation are taken for duplicate row elimination, i.e., time slot which is repeated is removed from the population in order to produce optimized TDMA frame.

Algorithm 1. Genetic algorithm

initialize population GAPop;
evaluate GAPop;
while (not terminated)
{
    GAPs = Select (GAPop);
    GAPc = Crossover (GAPs);
    GAPm = Mutate (GAPc);
    GAP' = evaluate GAPm;
    GAPop = Survival (GAPop, GAP');
}

3.3.3 Immune Genetic Algorithm

The disadvantage of genetic algorithm is that it has trouble in finding the exact global optimum because of the random and unsupervised
searching during the entire process. Therefore, there is no guarantee to find the best solution. Another drawback of GA is that in their computational time they are slower since GA lacks the capability of making use of some basic and obvious characteristic or knowledge on the pending problem. Based on the considerations above, IGA is proposed in this chapter. Algorithm 2 shows the structure of immune genetic algorithm. The solution after the reproduction stage is taken for immune operations. IGA is an intelligent optimization algorithm, which mainly constructs an immune operator accomplished by two steps: immune selection and vaccination. The knowledge added IGA algorithm is performed in the following way.

3.3.3.1 Immune selection

The newly created population after reproduction, which satisfies the primary and secondary constraints, is selected for duplicate row elimination. The resulting populations are arranged according to the channel utilization variable and stored in the vaccine pool.

3.3.3.2 Vaccination

Vaccination is used to improve the fitness by modifying the genes of an individual population with the prior knowledge to gain higher fitness with greater probability. A chromosome from vaccine pool is taken for vaccination. The IGA identifies the node that transmits first in the population. During the same time slot, some other node, which does not create interference with the transmitting node, can be allowed to transmit. To perform this, a node is selected randomly and checked with the hop matrix whether it creates an interference with the currently transmitting node. If not, the node value is mutated to one, allowing the selected node to transmit in the
same time slot. The genes of the selected chromosome are modified based on the knowledge obtained from the hop matrix of the given network. Hence, the vaccination process increases the number of transmissions.

Algorithm 2. Immune genetic algorithm

initialize population IGAPop;
evaluate IGAPop;
while (not terminated)
{
    IGAPs = Select (IGAPop);
    IGAPc = Crossover (IGAPs);
    IGAPm = Mutate (IGAPc);
    Immunization (IGAPm)
    {
        IGAPsel = ImmuneSelection (IGAPm);
        IGAPv = Vaccination (IGAPsel);
    }
    IGAP' = evaluate IGAPv;
    IGAPop = Survival (IGAPop, IGAP');
}

3.3.4 Memetic Algorithm

Memetic algorithms are extensions of evolutionary algorithms that apply local search processes in the agents and improve their fitness (Banos et al 2010, Krasnogor and Gustafson 2002, Moscato 1999, Lu and Hao 2010). Compared with other approaches, memetic algorithms are superior, because of wide applicability. The construction of memetic algorithm is given in Algorithm 3.

The initial population is constructed using the elite population method and the parent selection for reproduction is done using $k$–tournament selection. On the selected chromosomes, a single point crossover operator is performed and the mutation operator is carried out based
on the given mutation probability. After crossover and mutation, optimizer and improver are applied on the chromosomes in MA.

3.3.4.1 Optimizer

The optimizer phase of MA reduces the number of time slots by determining the channel utilization for each node. $\rho_x$ is the performance of node $x$ in the current population, i.e., the total number of transmissions carried out by the node $x$ in the given time slot is identified using Equation (2.3). The optimizer phase obtains each node transmissions. Then, it identifies whether the same node is transmitting in some other time slot $j$. In the $j^{\text{th}}$ time slot, if the nodes that are transmitted contain $\rho_x > 1$, then the row is removed from that population. In the algorithm reported by Chakraborty (2004), the rows that are subsets of a row generated after crossover are eliminated. In this chapter, duplicate row elimination in GA and IGA performs reduction of time slots, if a time slot is repeated in that population, while in MA the optimizer performs reduction based on the value of $\rho_x$. This phase generates population with minimum number of time slots with the constraint that every node has transmitted at least once in that TDMA frame.

3.3.4.2 Improver

Improver is a greedy way of improving solution and it reduces the solution diversity. The populations from optimizer are taken to the improver phase where the total number of transmissions is increased in the reduced time slots. Obtaining knowledge from hop matrix, improver phase increases the number of transmissions after reduction of time slots. Since memetic algorithm operations are carried out in each iteration, the optimal solution is
identified in less number of generations. Hence, running time of the algorithm is also reduced compared to other recently proposed competitive algorithms.

**Algorithm 3. Memetic algorithm**

initialize population MPop;
evaluate MPop;
while (not terminated)
{
    MPs = Select (MPop);
    MPc = Crossover (MPs);
    MPm = Mutate (MPc);
    MemeticAlg (MPm)
    {
        MPop = Optimizer (MPm);
        MPI = Improver (MPop);
    }
    MP’ = evaluate MPI;
    MPop = Survival (MPop, MP’);
}

**3.4 SIMULATION RESULTS**

A series of simulations was carried out to evaluate the performance of the proposed MA to solve the broadcast scheduling problem, in comparison with mean field annealing (Wang and Ansari 1997), GA (Chakraborty 2004) and competent permutation genetic algorithm (Wu et al 2005). The following sections discuss the simulation results regarding the number of nodes \( N \), the number of time slots \( M \) and the degree of networks. The fitness function factors depend on channel utilization variable and tight lower bound derived in the previous chapter.

If \( \Lambda = 1 \), then the solution is optimal. The terminal conditions for the three algorithms discussed in this study are \( \Lambda = 1 \) or the maximum number of generations, which is taken as 500 in all the experiments.
Various randomly generated networks with different degree and nodes test the three algorithms where each represents a multi-hop topology. For a particular setting of parameters, the algorithm was carried out for 150 times. The average value of the results is given in the following simulation results. The simulation results are based on the following parameters: population size 50, maximum number of generations 500, crossover rate 0.30, mutation probability 0.001 and on these three measures,

1. Tight lower bound $\Delta$ value is one.
2. Channel Utilization variable finds the improvement in the number of transmission.
3. Running time is measured on a simulation platform that uses Matlab code on Windows XP/Intel Core 2 Duo T6600 2.2 GHz machine.

### 3.4.1 Simulation Results of GA

The purpose of the first simulation was to investigate the performance of GA for different networks shown in Table 3.1. The number of nodes taken for simulation ranges from five to hundred. Smaller node networks are executed with more number of transmissions in an acceptable generation. However, a 100 node network with 200 edges with degree of nine identifies the optimal solution TDMA frame after 489 generations. The average number of generations for a 100-node network is 422. This has to be reduced in order to reduce the execution time.
Table 3.1 Simulation results of GA

<table>
<thead>
<tr>
<th>No. of nodes</th>
<th>No. of links</th>
<th>Avg. degree</th>
<th>Avg. ND</th>
<th>Max. ND</th>
<th>Minimum TDMA frame length</th>
<th>Avg. $\sigma$</th>
<th>Avg. No. of generations</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>25</td>
<td>3.3</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>0.219</td>
<td>30.6</td>
<td>0.80 sec.</td>
</tr>
<tr>
<td>30</td>
<td>49</td>
<td>3.3</td>
<td>4.8</td>
<td>8</td>
<td>9</td>
<td>0.156</td>
<td>27.2</td>
<td>01.10 min.</td>
</tr>
<tr>
<td>80</td>
<td>156</td>
<td>3.9</td>
<td>5.8</td>
<td>8</td>
<td>9</td>
<td>0.154</td>
<td>238</td>
<td>16.08 min.</td>
</tr>
<tr>
<td>100</td>
<td>200</td>
<td>4</td>
<td>7.5</td>
<td>9</td>
<td>10</td>
<td>0.104</td>
<td>422</td>
<td>32.00 min</td>
</tr>
</tbody>
</table>

Table 3.1 Simulation results of GA
3.4.2 Simulation Results of IGA

Table 3.2 represents the output produced by immune genetic algorithm for varying number of nodes and edges. Compared to genetic algorithm, knowledge added IGA could improve the searching ability and adaptability, greatly increase the converging speed (Liu et al 2006, Jiao and Wang 2000, Wang 2009 a). During vaccination process, the selected antigen is improved with more number of transmissions so that the channel utilization is increased.

Comparing the simulation results of IGA in Table 3.2 with GA in Table 3.1, the number of generations is reduced and the average number of transmission of each network is improved. For network with 80 nodes and 100 nodes, the solution is identified with acceptable generation. For a 100-node network with average degree of four and five, the optimal solution is identified in 16 min and 25 min. However, first two measures are satisfied by IGA, but third one, i.e., running time for a large network is not reduced.

3.4.3 Simulation Results of MA

The methods discussed so far mainly focused on the convergence of the algorithms in terms of tight lower bound and increase in number of transmissions. Therefore, the question arises: What is the relation of these methods compared to each other in terms of time? This has been set as the main question of MA. The simulation result of MA aims at finding the efficiency, i.e., the speed of convergence.
Table 3.2 Simulation results of IGA

<table>
<thead>
<tr>
<th>No. of nodes</th>
<th>No. of links</th>
<th>Avg. degree</th>
<th>Avg. ND</th>
<th>Max. ND</th>
<th>Minimum TDMA frame length</th>
<th>Avg. $\sigma$</th>
<th>Avg. No. of generations</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>25</td>
<td>3.3</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>0.289</td>
<td>18.6</td>
<td>0.5 sec.</td>
</tr>
<tr>
<td>30</td>
<td>60</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>9</td>
<td>0.199</td>
<td>21.0</td>
<td>12 sec.</td>
</tr>
<tr>
<td>40</td>
<td>80</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>0.187</td>
<td>35.2</td>
<td>3.8 sec.</td>
</tr>
<tr>
<td>50</td>
<td>100</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>0.180</td>
<td>49.7</td>
<td>10.23 sec.</td>
</tr>
<tr>
<td>70</td>
<td>140</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0.172</td>
<td>64.9</td>
<td>2.49 min.</td>
</tr>
<tr>
<td>80</td>
<td>160</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0.167</td>
<td>89.0</td>
<td>4.0 min.</td>
</tr>
<tr>
<td>100</td>
<td>200</td>
<td>4</td>
<td>7</td>
<td>9</td>
<td>10</td>
<td>0.141</td>
<td>92.3</td>
<td>12.61 min.</td>
</tr>
<tr>
<td>100</td>
<td>250</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>0.117</td>
<td>98.0</td>
<td>27.37 min.</td>
</tr>
</tbody>
</table>
It is clear from Tables 3.1 and 3.2 that IGA has improved the channel utilization in reduced number of generations and also in less computation time compared to GA. The average channel utilization, average number of generations and the computation time of various networks using MA for network with different degree is analyzed in Table 3.3.

The time taken for 100-node network in MA is 2.0 min is more efficient than the time taken by IGA for the same network, i.e., 12.61 min. For network with more than 100 nodes, computation time is not much efficient in IGA and other recently proposed efficient methods when compared with MA. This is the main advantage of MA.

Total number of transmissions generated by MA for different node networks with different time slots is compared with hopfield neural network with genetic algorithm (HNN-GA) given by Salcedo-Sanz et al (2003), tabu search with greedy method (TS-GR) reported by Peng et al (2004) and integer programming broadcast scheduling problem (IPBSP) given by Menon (2009) as shown in Table 3.4. Transmission comparison graph of MA with these algorithms is also shown in Figure 3.2. The starting point of vertical line in the graph represents lowest transmission value for given network, the ending point of the line represents highest transmission value and a small horizontal line represents number of transmissions generated by each algorithm.

The table and figure proves that MA produces higher number of transmissions for varying networks compared to existing algorithms and produces with a difference of 6 to 11 in total number of transmissions. The computation time and number of generations to identify optimal solutions are reduced whereas channel utilization is increased in MA compared to GA by Chakraborty (2004). The comparison of average time taken by MA and GA (Chakraborty 2004) is given in Figure 3.3.
Table 3.3 Simulation results of MA

<table>
<thead>
<tr>
<th>No. of nodes</th>
<th>No. of links</th>
<th>Avg. degree</th>
<th>Avg. ND</th>
<th>Max. ND</th>
<th>Minimum TDMA frame length</th>
<th>Avg. ( \sigma )</th>
<th>Avg. No. of generations</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>85</td>
<td>3.4</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>0.203</td>
<td>5.12</td>
<td>7 sec.</td>
</tr>
<tr>
<td>80</td>
<td>152</td>
<td>3.8</td>
<td>7</td>
<td>9</td>
<td>10</td>
<td>0.175</td>
<td>7.48</td>
<td>11 sec.</td>
</tr>
<tr>
<td>100</td>
<td>150</td>
<td>3</td>
<td>6.9</td>
<td>9</td>
<td>10</td>
<td>0.170</td>
<td>9.03</td>
<td>1.7 min.</td>
</tr>
<tr>
<td>100</td>
<td>200</td>
<td>4</td>
<td>7.5</td>
<td>10</td>
<td>11</td>
<td>0.162</td>
<td>16.58</td>
<td>2.0 min.</td>
</tr>
<tr>
<td>100</td>
<td>250</td>
<td>5</td>
<td>7</td>
<td>10</td>
<td>11</td>
<td>0.147</td>
<td>17.0</td>
<td>2.0 min</td>
</tr>
<tr>
<td>100</td>
<td>300</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0.121</td>
<td>19.76</td>
<td>2.3 min</td>
</tr>
<tr>
<td>200</td>
<td>400</td>
<td>4</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>0.150</td>
<td>29.76</td>
<td>12.2 min.</td>
</tr>
<tr>
<td>200</td>
<td>500</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>0.139</td>
<td>38.02</td>
<td>20.8 min</td>
</tr>
<tr>
<td>300</td>
<td>600</td>
<td>4</td>
<td>7</td>
<td>10</td>
<td>11</td>
<td>0.176</td>
<td>55.54</td>
<td>30.5 min</td>
</tr>
<tr>
<td>400</td>
<td>800</td>
<td>4</td>
<td>12</td>
<td>16</td>
<td>17</td>
<td>0.159</td>
<td>60.58</td>
<td>65.3 min.</td>
</tr>
<tr>
<td>500</td>
<td>1000</td>
<td>4</td>
<td>9</td>
<td>14</td>
<td>15</td>
<td>0.128</td>
<td>89.04</td>
<td>72.11 min.</td>
</tr>
</tbody>
</table>
Table 3.4 Comparison of MA with other recent algorithms in terms of number of transmissions

<table>
<thead>
<tr>
<th>No. of nodes</th>
<th>Time slots</th>
<th>HNN-GA</th>
<th>TS-GR</th>
<th>IPBSP</th>
<th>MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>8</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>28</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>35</td>
<td>37</td>
<td>37</td>
<td>48</td>
</tr>
<tr>
<td>30</td>
<td>11</td>
<td>40</td>
<td>43</td>
<td>-</td>
<td>51</td>
</tr>
<tr>
<td>30</td>
<td>12</td>
<td>47</td>
<td>48</td>
<td>-</td>
<td>54</td>
</tr>
<tr>
<td>40</td>
<td>8</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>76</td>
</tr>
<tr>
<td>40</td>
<td>9</td>
<td>77</td>
<td>77</td>
<td>-</td>
<td>84</td>
</tr>
</tbody>
</table>
Figure 3.2  Comparison of number of transmissions in existing algorithms and MA

Figure 3.3  Comparison of computation time taken by MA and GA
Figure 3.4  Comparison of average time delay for 30-node and 40-node networks

Figure 3.5  Comparison of channel utilization for 30-node and 40-node networks
Two benchmark problems discussed by Wang and Ansari (1997) are solved using MA and the results are compared with other algorithms such as G-HNCNN, G-NCNN, GACFS, FSMA, CPGA and MFA as shown in Table 3.5. 30 nodes with 70 edges are analyzed in instance #1 and 40 nodes with 66 edges are analyzed in instance #2 by considering the number of time slots, channel utilization and time delay. The time delay is calculated using

$$\text{Time delay} = \frac{M}{|N|} \sum_{i=1}^{N} \left( \frac{1}{\sum_{j=1}^{M}\mid S_{ij} \mid} \right)$$  \hspace{1cm} (3.1)$$

As seen in Table 3.5, MA increases the channel utilization with the smallest time delay. This indicates that MA performs efficiently when compared to the other recently proposed algorithms. Figures 3.4 and 3.5 compares the time delay and channel utilization of various algorithms for 30-node and 40-node networks with MA. Table 3.6 compares the computation time of MA with algorithms HNN-GA, IPBSP and illustrates that computation time is greatly reduced.

Figure 3.6 compares channel utilization generated by GA, IGA and MA with total number of nodes. Figure 3.7 compares number of generations taken by GA, IGA and MA with number of nodes of various networks. These results illustrate that MA performs efficiently in terms of tight lower bound and increases the channel utilization in an acceptable computation time.

The average time delay taken by HNN-GA, G-NCNN and MA for various networks ranging from 15 to 250 is compared in Figure 3.8. If there is more number of transmissions, then there is a decrease in the time delay. Figure 3.8 also indicates the total number of transmission produced by MA is high when compared to other two algorithms.
Table 3.5  Comparison of MA with other competitive algorithms in terms of time slot, channel utilization and time delay

<table>
<thead>
<tr>
<th>Instance</th>
<th>Parameter</th>
<th>MA</th>
<th>G-HNCNN</th>
<th>G-NCNN</th>
<th>GACFS</th>
<th>FSMA</th>
<th>CPGA</th>
<th>MFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td></td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>M</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\sigma$</td>
<td>0.24</td>
<td>0.1233</td>
<td>----</td>
<td>0.093</td>
<td>0.1167</td>
<td>0.1233</td>
<td>0.1056</td>
</tr>
<tr>
<td></td>
<td>Time delay</td>
<td>6.1529</td>
<td>8.83</td>
<td>9.0</td>
<td>----</td>
<td>9.2</td>
<td>----</td>
<td>10.5</td>
</tr>
<tr>
<td>#2</td>
<td></td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>M</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\sigma$</td>
<td>0.2844</td>
<td>0.2125</td>
<td>----</td>
<td>0.203</td>
<td>0.200</td>
<td>0.200</td>
<td>0.197</td>
</tr>
<tr>
<td></td>
<td>Time delay</td>
<td>5.0433</td>
<td>5.7056</td>
<td>5.8</td>
<td>----</td>
<td>6</td>
<td>----</td>
<td>6.9</td>
</tr>
</tbody>
</table>
Table 3.6 Comparison of MA with other competitive algorithms in terms of computation time

<table>
<thead>
<tr>
<th>No. of nodes</th>
<th>Average degree</th>
<th>HNN-GA (sec)</th>
<th>IPBSP (sec)</th>
<th>MA (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3</td>
<td>5.1</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>12.97</td>
<td>0.06</td>
<td>0.01</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>85.4</td>
<td>0.07</td>
<td>0.02</td>
</tr>
<tr>
<td>40</td>
<td>3</td>
<td>165.12</td>
<td>0.3</td>
<td>0.14</td>
</tr>
<tr>
<td>50</td>
<td>3</td>
<td>194.4</td>
<td>0.64</td>
<td>0.27</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>5.24</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>20.04</td>
<td>0.33</td>
<td>0.01</td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>152.06</td>
<td>1.45</td>
<td>2.04</td>
</tr>
<tr>
<td>40</td>
<td>4</td>
<td>280.37</td>
<td>4.74</td>
<td>3.41</td>
</tr>
<tr>
<td>50</td>
<td>4</td>
<td>320.06</td>
<td>14.2</td>
<td>7.00</td>
</tr>
</tbody>
</table>
Figure 3.6 Comparison of channel utilization in GA, IGA and MA

Figure 3.7 Comparison of generations in GA, IGA and MA
Figure 3.8  Comparison of average time delay in HNN-GA, G-NCNN and MA

Table 3.7 Comparison of computation time in SLBIP and MA

<table>
<thead>
<tr>
<th>No. of nodes</th>
<th>No. of links</th>
<th>SLBIP (sec)</th>
<th>MA (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>24</td>
<td>0.031</td>
<td>0.011</td>
</tr>
<tr>
<td>12</td>
<td>36</td>
<td>1.015</td>
<td>0.073</td>
</tr>
<tr>
<td>12</td>
<td>48</td>
<td>3.119</td>
<td>1.492</td>
</tr>
<tr>
<td>15</td>
<td>56</td>
<td>16.558</td>
<td>8.961</td>
</tr>
<tr>
<td>20</td>
<td>68</td>
<td>6.289</td>
<td>2.133</td>
</tr>
<tr>
<td>35</td>
<td>100</td>
<td>1.613</td>
<td>0.918</td>
</tr>
</tbody>
</table>
Figure 3.9 Network with 50 nodes, 85 edges and average degree of 3.4

Figure 3.10 Optimal solution found by MA for the network given in Figure 3.9
A sample 50-node network with 85 edges with average degree of 3.4 and its corresponding optimal TDMA frame identified using MA is shown in Figures 3.9 and 3.10. Since maximum degree of the network is six, MA produces an optimal schedule in seven time slots with 61 transmissions. Channel utilization for this network is 0.174 and average time delay is 6.27.

Table 3.7 compares the computation time of MA for varying number of nodes and links with SLBIP (Oki and Iwaki 2010). All these results show that MA performs better compared not only with GA and IGA but also with the recently proposed algorithms as discussed earlier.

3.5 CONCLUSION

The basic genetic algorithm, knowledge added immune genetic algorithm and a domain specific memetic algorithm are discussed to solve wireless multi-hop network broadcast scheduling. Compared to GA and IGA, MA actively aims at improving solutions. While GA blindly wanders over the search space, MA exploits all available knowledge about the problem. While IGA gets knowledge from hop matrix during vaccination process. IGA also increases the number of transmissions in a reduced time slot but not in a good computation time, but MA overcomes it. The previous algorithms, had the main drawback of computation time for a large network, which is greatly reduced in MA.

The simulation results confirm that MA significantly outperforms several heuristic and evolutionary algorithms by solving well-known benchmark problems in terms of solution quality, which also demonstrates the effectiveness of MA in efficient use of channel bandwidth. The results
validate the advantages of MA in terms of channel utilization, number of
generations and running time. MA achieves the tight lower bound in shorter
running time compared with other algorithms. The outcome validates the
effectiveness and efficiency of MA for the broadcast scheduling problem. A
new metaheuristic algorithm that reduces the execution time further compared
to the MA attained is examined in the next chapter.