INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2011; **24**:239–257
Published online 16 June 2010 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/dac.1152

Genetic algorithm-based scheduling in cognitive radio networks under interference temperature constraints

Didem Gözüpek and Fatih Alagöz*, †

Computer Engineering Department, Boğaziçi University, P.K. 2, TR-34342, Bebek, Istanbul, Turkey

SUMMARY

The proliferation of wireless technologies and services has intensified the demand for the radio spectrum. However, the currently existing fixed spectrum assignment policy leads to an inefficient and unevenly distributed spectrum utilization. Cognitive radio paradigm has been proposed to alleviate these drawbacks by employing dynamic spectrum access (DSA) methodology. Federal Communications Commission (FCC) has proposed the interference temperature model, which enables the unlicensed users to utilize the licensed frequencies simultaneously with the licensed users as long as they conform to the interference temperature constraints. Recently, throughput and delay optimal schedulers that meet the interference temperature constraints in cognitive radio networks have been formulated in the literature. However, these schedulers have high computational complexity. In this paper, we propose genetic algorithm (GA)-based suboptimal methods addressing these throughput and delay optimal scheduling problems. The simulation results corroborate that our GA-based approach yields very close performance to the optimal solutions and operates with much lower complexity. Copyright © 2010 John Wiley & Sons, Ltd.

Received 18 July 2009; Revised 15 February 2010; Accepted 16 April 2010

KEY WORDS: interference temperature; scheduling; cognitive radio networks; genetic algorithms

1. INTRODUCTION

The explosive growth in wireless technologies has escalated the demand for the radio spectrum. Per contra, recent studies exhibit that spectrum usage is concentrated on certain portions of the spectrum, while a significant amount of the spectrum remains unutilized [1]. This situation implies that there is a problem with the spectrum management and allocation methodology, rather than the true scarcity of the spectrum. Dynamic spectrum access (DSA) methods enable the devices to opportunistically access the licensed frequency bands, and thereby enhance the utilization of the existing wireless spectrum. Cognitive radios are smart devices that can sense and autonomously reason about their environment, and adapt their communication parameters in response to the changing conditions. They are the next evolution of adaptive/aware radios through the addition of a layer of intelligence providing the ability to realize the DSA concept [2].

The nodes in a cognitive radio network can be classified as primary user (PU) and secondary user (SU). A PU is a licensed user that has paid for the spectrum, and hence has exclusive rights to access it. A SU is an unlicensed user that can utilize the temporarily unused licensed spectrum

*Correspondence to: Fatih Alagöz, Computer Engineering Department, Boğaziçi University, P.K. 2, TR-34342, Bebek, Istanbul, Turkey.
†E-mail: fatih.alagoz@boun.edu.tr

Contract/grant sponsor: State Planning Organization of Turkey; contract/grant number: DPT-2007K 120610
Contract/grant sponsor: Scientific and Technological Research Council of Turkey (TUBITAK); contract/grant number: 109E256
Contract/grant sponsor: Boğaziçi University Research Fund (BAP); contract/grant number: 09A108P

bands opportunistically, as long as it vacates them as soon as a PU appears [3]. PUs are oblivious to the SUs and do not have cognitive capabilities. In the remainder of this paper, we use the terms *cognitive users* and *SUs* interchangeably.

In our previous work [4], we have formulated throughput and delay optimal schedulers for cognitive radio networks under interference temperature constraints. However, the formulated optimal schedulers have high computational complexity. To this end, suboptimal schedulers referred to as *Maximum Frequency Selection* (*MFS*) and *Probabilistic Frequency Selection* (*PFS*) have also been proposed in [4]. Although the computational complexity of the MFS and PFS schedulers is low, their throughput and delay performance are not satisfactory. Consequently, we have called for the design of better performing, yet computationally efficient suboptimal schedulers in [4]. In this paper, we address this open research issue identified by Gözüpek and Alagöz [4], and propose two GA-based suboptimal schedulers that yield very close performance to the optimal schedulers. To the best of our knowledge, the use of GAs for scheduling in cognitive radio networks has not previously been explored.

The remainder of the paper is structured as follows: Section 2 describes the related work, whereas Section 3 provides the relevant background information with respect to the throughput and delay optimal scheduling problems. Section 4 introduces our proposed GA-based suboptimal solutions to these problems, and Section 5 discusses the simulation results. Finally, Section 6 concludes the paper.

2. RELATED WORK

The usage of GAs has been proven to be quite successful for channel assignment schemes in cellular networks [5, 6]. In cognitive radio networks context, on the other hand, the studies about the usage of GAs mainly revolve around the configuration of various cognitive radio parameters such as pulse shape, symbol rate, and modulation. The authors in [7] present GA-based adaptive component of the cognitive radio engine developed at the Virginia Tech Center for Wireless Telecommunications (CWT). They first formulate a multi-objective optimization problem, and then evaluate the fitness function for this overall problem as the weighted sum of the fitness values of each objective. In line with this formulation, they define the chromosome structure as consisting of power, frequency, pulse shape, symbol rate, and modulation.

The cognitive radio software testbed discussed in [8] includes a cognitive radio engine based on GAs. The engine executes two separate GAs to select the channel and transmission parameters, each of which is a multi-objective problem similar to the one in [7].

The authors in [9] formulate the channel assignment problem specific to cognitive radio networks and propose an island GA, in which the population is divided into sub-populations called islands, and the chromosomes interact through migration to other islands. The channel assignment problem that they consider determines which channel (frequency) to assign for which communication link; hence, it is a static one-shot assignment procedure. In this paper, in contrast, we focus on the scheduling problem rather than the channel assignment problem. Therefore, unlike the work in [9], we also have a time aspect; i.e. the problems we focus on determine both the frequencies and the time slots to assign to the SUs.

The population adaptation technique introduced in [10] again considers cognitive radio engines, and devises a method to expedite the convergence of the GAs. Its method is based on having the cognitive engine to utilize the information about the wireless environment learned in the previous cognition cycles and seeding the initial generation of the GA with high scoring chromosomes from the previous run. Its results demonstrate that this approach performs quite well in slowly varying wireless environments but yields poor results when the conditions are changing fast.

The essential issues in scheduling, on the other hand, have been widely studied in the conventional networks [11–14]. Nonetheless, cognitive radio paradigm brings this topic into the focus of research again. Cognitive radio concept presents new challenges to the scheduling mechanisms because the

DOI: 10.1002/dac

altering channel availability owing to the coexistence of primary and SUs mandates the cognitive users to determine when and on which channel they should tune to in order to communicate with their neighbors.

Formulation of throughput maximization under signal-to-noise ratio (SNR) and interference conditions has previously been considered in the wireless networking literature [15, 16], where graph theoretical approaches are usually adopted. The scheduling problem considered in this paper can be distinguished from these works by its cognitive radio specific nature; i.e. it accommodates the usage of different frequencies, where not only the availability but also the maximum allowable transmission rates of the frequency bands are time-varying.

The authors in [17] focus on a radio resource allocation scheme for OFDMA-based cognitive radio networks. They propose an improved water-filling power allocation algorithm as well as an uplink subcarrier allocation algorithm. The authors estimate the water-filling level and also provide a lower bound on the stability of this estimation via an asymptotic performance analysis.

The adaptive downlink packet scheduling algorithm for cognitive radio networks proposed in [18] encompasses quality of service and spectrum variation awareness capability. The authors first calculate the priority values for the traffic queues in conformity with a priority function with channel adaptive coefficient. Subsequently, they decide on the best available channel among the channels that have free time slots and then allocate the time slots. Since they select the best available channel among the free channels, their scheme does not enable true coexistence of PUs and SUs.

The authors in [19] devise opportunistic scheduling policies that maximize the throughput of SUs subject to the maximum collision constraints with the PUs. They use the Lyapunov optimization technique in designing their algorithms. They assume that a collision occurs if an SU attempts to transmit in a channel that is already occupied by a PU; hence, their scheme also does not enable the true coexistence of PUs and SUs.

The authors in [20] formulate an integer linear programming (ILP) problem for the MAC-layer scheduling. Their scheme minimizes the schedule length in multi-hop cognitive radio networks. Moreover, they also propose a distributed heuristic to determine the channels and time slots for the cognitive nodes. However, they consider the interference to the PUs neither in their optimization formulation nor in their suboptimal heuristic. Therefore, the scheduling problem we focus on in this paper is distinct in principle from the work in [20].

The interference temperature model [21] proposed by FCC prescribes true coexistence of licensed and unlicensed users. In this model, SUs are allowed to simultaneously operate on the same frequencies as the PUs as long as they can quantify and bound the additional interference exposed to the PUs. The interference temperature threshold for a particular frequency quantifies the maximum aggregate interference that can occur at a PU in that frequency. Our work in this paper is based on our previous work in [4], which focuses on scheduling in cognitive radio networks under interference temperature constraints. In [4], we had formulated the problems of throughput maximization and delay minimization as optimization problems. We had also proposed suboptimal schedulers with low complexity at the expense of poor throughput and delay performance. Hence, an open research issue about the design of better performing suboptimal schedulers with reasonable complexity has been pointed out in [4]. In this paper, we address this research issue and propose a GA-based suboptimal solution to these optimization problems. We demonstrate the efficacy of our approach through extensive simulations.

3. BACKGROUND INFORMATION ON THROUGHPUT AND DELAY OPTIMAL SCHEDULING PROBLEMS

A time-slotted IEEE 802.22 system [22], where the cognitive base station (CBS) controls and guides the SUs is considered. Figure 1 illustrates the pertinent network architecture. The scheduler is at the CBS and determines how many packets and with which frequency each SU will transmit in each time slot. The goal of the throughput optimal scheduler is to maximize

DOI: 10.1002/dac

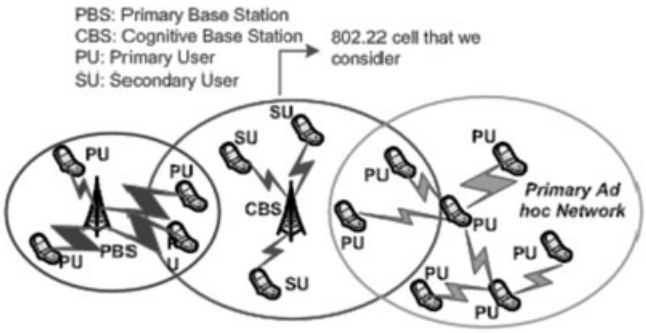

Figure 1. IEEE 802.22 cell.

the total throughput of the SUs in the cell, whereas the goal of the delay optimal scheduler is to minimize the total scheduling delay of the SUs in the cell. Both schedulers ensure that the interference temperature perceived by the PUs in the cell is within the interference temperature limits, reliable communication between the CBS and the SUs is achieved, and collisions among the SUs are avoided [4].

A two-step solution is proposed by Gözüpek and Alagöz [4] to both throughput and delay optimal schedulers. In the first step of both schedulers, each SU i calculates for every frequency f the value for U_{if}, which is the maximum number of packets that it can transmit for frequency f in a time slot. The calculation procedure for U_{if} values guarantees that the interference temperature perceived by the PUs is within the predetermined limits. The CBS then constitutes a matrix called $\mathbf{U}=[U_{if}]$. This matrix is utilized in the second step, which is a different binary integer program for both schedulers [4].

Throughput optimal scheduler: The CBS executes the following binary ILP in the second step of the throughput optimal scheduler:

$$\max\left(\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T}\frac{U_{if}X_{ift}}{T}\right) \tag{1}$$

$$\text{s.t.}\ \sum_{f=1}^{F}\sum_{t=1}^{T}X_{ift}\geqslant 1 \quad \forall i\in\{1,\ldots,N\}, \tag{2}$$

$$X_{ift}+X_{i'ft}\leqslant 1, \quad \forall i,\ i'\in\{1,\ldots,N\},\ i\neq i',\ \forall f,\ \forall t, \tag{3}$$

where N is the total number of SUs, F is the total number of frequencies, T is the total number of time slots in the schedule, and X_{ift} is a binary variable such that $X_{ift}=1$ if user i transmits with frequency f in time slot t and 0 otherwise. Here, (2) guarantees that at least one time slot is assigned to each SU, whereas (3) makes certain that at most one user can transmit in a particular time slot and frequency combination, and consequently preventing collisions among the cognitive nodes. Moreover, the schedule length T is the time period in which the spectral and networking environment changes slowly enough so that the U_{if} values are not affected. For example, the TV bands used by an IEEE 802.22 network constitute a slowly altering spectral environment, and hence enable T to be quite large [4].

Delay optimal scheduler: The CBS executes the following binary integer program in the second step of the delay optimal scheduler:

$$\min\left(\frac{\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T}tU_{if}X_{ift}}{\sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T}U_{if}X_{ift}}\right) \tag{4}$$

$$\text{s.t.}\ \sum_{i=1}^{N}\sum_{f=1}^{F}\sum_{t=1}^{T}U_{if}X_{ift}>0. \tag{5}$$

(2) and (3)

DOI: 10.1002/dac

The objective function in (4) minimizes the scheduling delay in terms of time slots experienced per packet. Furthermore, (5) obviates the case where the scheduler constantly chooses the frequencies with which the nodes can send at most zero packets in order to decrease the average scheduling delay. In other words, (5) avoids the situation of zero throughput [4].

The two optimization problems formulated as throughput and delay optimal schedulers are binary integer programming problems, which are known to have high computational complexity. In other words, the high complexity of both problems stems from the second step. Our GA-based algorithms take the U_{if} values resulting from the first step as input, and operate as a second step in the solution procedure.

4. PROPOSED GA-BASED SCHEDULERS

4.1. Overview of GAs

A GA is a biologically inspired heuristic search technique appropriate for problems with large search spaces. It operates by emulating natural processes such as reproduction, evolution, and survival. In nature, populations of individuals compete for resources, and the best suited for competition survive, whereas the weakest tends to die out. This phenomenon is referred to as '*survival of the fittest*' in evolution. These ideas were first incorporated into a problem solving algorithm by Holland [23] and they are now used in a multitude of problems.

A GA operates by evolving a *population* of solutions called *chromosomes*. A chromosome is a binary string that represents one sample in the solution space of the problem. The bits of the string are regarded as the *genes* of the chromosome. The *fitness value* assigned to a chromosome exhibits the extent to which the chromosome satisfies the problem requirements. A GA takes a group of chromosomes from a population using a genetic operation called *selection*, and mixes the genes of these chromosomes using *crossover* to produce offspring. These offspring solutions may be further randomly altered using a genetic operation called *mutation*.

The fact that GAs operate on a population of solutions rather than a single solution implies that the algorithm makes parallel searches in the search space. The selection operation, on the other hand, serves the purpose of eliminating the relatively bad solution candidates and focusing the search operation on a relatively good portion of the solution space. Crossover operation is based on the idea that if two solution candidates that are both good but for different reasons (for instance the first half of the bits in one candidate have desirable qualities, like not violating any problem constraints, and the second half of the bits in the other candidate have good features in a similar way), then we can obtain an even better solution if we take the good parts of both solutions and combine them. Mutation, on the other hand, is like introducing some noise with little magnitude into the system in order to take the search procedure out of a locally optimal region, and enable the search process to possibly delve into a better region of the search space.

Our motivation for utilizing GAs in designing suboptimal schedulers for the throughput and delay optimal scheduling problems is manifold. First, GAs are proper for problems with large search spaces. They are equipped with many tools to reduce computational complexity and produce a diverse set of solutions, since they can quickly center in a specific solution and diversify search to develop wide range of solutions to address unknown environments. Considering that the solution space in the throughput and delay optimal scheduling problems is enormous (even for 5 nodes, 3 frequencies, and 5 time slots, the size of the solution space for the throughput optimal scheduler is 2^{75}), GAs seem to be a suitable tool. Second, GAs can be implemented on semiconductor devices and enable rapid integration with wireless technologies and leverage economies of scale. Rapid prototyping is possible using digital signal processors (DSPs) or field programmable gate arrays (FPGAs). Third, as also pointed out by Gözüpek and Alagöz [4], the binary decision variables X_{ift} can be easily encoded to a binary string, and therefore, GAs can be conveniently implemented.

DOI: 10.1002/dac

4.2. GA-based suboptimal schedulers

Figure 2 illustrates the flowchart of our proposed GA-based algorithm for both throughput and delay optimal scheduling schemes.

4.2.1. Chromosome representation (*encoding*). We use binary encoded chromosomes, which contain X_{ift} values. Thus, the chromosome size is equal to $N \times F \times T$. The order used in decoding the possible X_{ift} values has an impact on the performance of the algorithm, since this order affects the gene pattern that can survive in the subsequent generations. In our analysis, we evaluate the impact of the following two encoding methods. In *Encoding Type-1*, the chromosome structure is $[X_{111}, X_{211}, X_{311}, X_{112}, X_{212}, \ldots, X_{123}, X_{223}, X_{323}]$, whereas in *Encoding Type-2*,

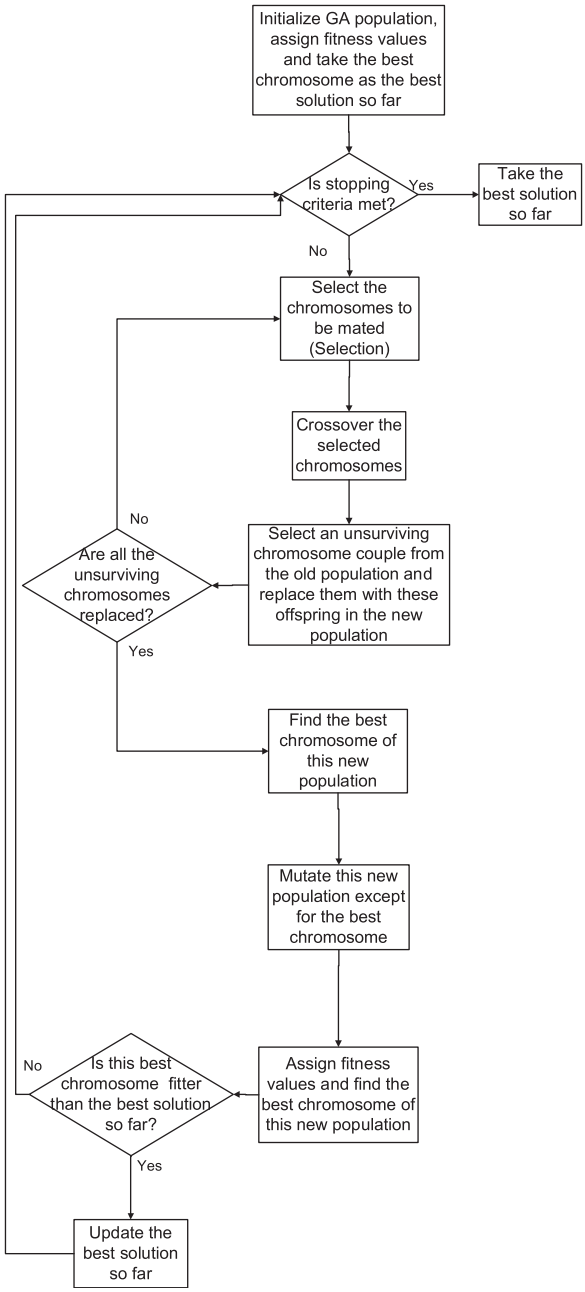

Figure 2. Block diagram for our proposed GA-based algorithm.

DOI: 10.1002/dac

the chromosome structure is $[X_{111}, X_{112}, X_{113}, X_{121}, X_{122}, \ldots, X_{322}, X_{323}]$ for $N=3$, $F=2$, and $T=3$.

4.2.2. Initial population creation. The most common way to generate the initial population is to employ uniformly random generation for each bit of the chromosomes. We call this approach *Rand* in this paper. Another alternative, which we refer to as *RandComp*, is to randomly generate half of the chromosomes, and then take the complement of the first half to generate the second half [24]. This approach ensures diversity by requiring every bit to assume both a one and a zero within the population. In both of our suboptimal schedulers, we assess the impact of both approaches on the performance of the algorithm.

4.2.3. Fitness function evaluation. The extent to which a chromosome satisfies the problem requirements depends on two factors. The first one is how much it maximizes or minimizes the objective function, and the second one is how many of the problem constraints it violates. We represent the former by a *primary fitness* value and the latter by a *secondary fitness* value. If any constraint is violated, the primary fitness F_{p} equals zero and the secondary fitness F_{s} is nonzero. Likewise, if no constraint is violated, F_{s} equals zero and F_{p} is nonzero. More formally, we formulate the fitness values for the throughput maximizing scheduler as follows:

$$F_{\mathrm{p}} = \begin{cases} 0 & \text{if } V_1 + V_2 > 0, \\ \displaystyle\sum_{i=1}^{N} \sum_{f=1}^{F} \sum_{t=1}^{T} \frac{U_{if} X_{ift}}{T} & \text{otherwise (if } V_1 + V_2 = 0), \end{cases} \tag{6}$$

$$F_{\mathrm{s}} = \begin{cases} 0 & \text{if } V_1 + V_2 = 0, \\ \dfrac{1}{V_1 + V_2} & \text{otherwise (if } V_1 + V_2 > 0), \end{cases} \tag{7}$$

where V_1 is the number of violations of constraint (2), and V_2 is the number of violations of constraint (3). If $V_1 + V_2 > 0$, then it means that some constraint is violated. If $V_1 + V_2 = 0$, on the other hand, it means that none of the constraints are violated.

Specifically, we can express V_1 and V_2 as follows:

$$V_1 = \sum_{i=1}^{N} V_1^i \quad \text{where } V_1^i = \begin{cases} 1 & \text{if } \displaystyle\sum_{f=1}^{F} \sum_{t=1}^{T} X_{ift} < 1 \quad \text{for } i \in \{1, \ldots, N\}, \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

$$V_2 = \sum_{i=1}^{N} \sum_{i'=i+1}^{N} \sum_{f=1}^{F} \sum_{t=1}^{T} V_2^{ii'ft}, \tag{9}$$

$$V_2^{ii'ft} = \begin{cases} 1 & \text{if } X_{ift} + X_{i'ft} > 1, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Besides, we define the fitness functions for the delay minimizing scheduler as follows:

$$F_{\mathrm{p}} = \begin{cases} 0 & \text{if } V_1 + V_2 + V_3 > 0, \\ \dfrac{\sum_{i=1}^{N} \sum_{f=1}^{F} \sum_{t=1}^{T} U_{if} X_{ift}}{\sum_{i=1}^{N} \sum_{f=1}^{F} \sum_{t=1}^{T} t U_{if} X_{ift}} & \text{otherwise (if } V_1 + V_2 + V_3 = 0), \end{cases} \tag{11}$$

$$F_{\mathrm{s}} = \begin{cases} 0 & \text{if } V_1 + V_2 + V_3 = 0, \\ \dfrac{1}{V_1 + V_2 + V_3} & \text{otherwise (if } V_1 + V_2 + V_3 > 0), \end{cases} \tag{12}$$

DOI: 10.1002/dac

$$V_3 = \begin{cases} 0 & \text{if } \sum_{i=1}^{N} \sum_{f=1}^{F} \sum_{t=1}^{T} U_{if} X_{ift} > 0, \\ 1 & \text{otherwise,} \end{cases} \tag{13}$$

where V_3 is the number of violations of constraint (5), and V_1 and V_2 are the same as in the throughput maximizing scheduler.

We demonstrate in Table I the method we employ for the overall fitness comparison of two chromosomes α and β having primary fitness F_{p}^{α}, F_{p}^{β} and secondary fitness F_{s}^{α}, F_{s}^{β}. The primary fitness values have high priority in the comparison procedure; that is to say, the chromosome that has higher primary fitness value than the other chromosome is declared to have the higher fitness value. The reason for this is that a chromosome that does not violate any constraints and hence has positive primary fitness value is fitter than the one that violates some constraint and therefore having a primary fitness value of zero. Likewise, when comparing two chromosomes that both have a positive primary fitness value (i.e. none of them violate any constraint), the one that better satisfies the objective function has the higher fitness value. If both chromosomes have zero primary fitness value; i.e. both chromosomes violate some problem constraint, then the chromosome that violates less number of problem constraints and hence has higher secondary fitness value is declared to have the higher fitness value.

One of the possible approaches employed when handling constraint-based problems using GAs is to penalize a constraint violating chromosome by setting its fitness value to zero, and hence basically nullifying its chance to survive to the next generation [7, 25]. Nevertheless, in our scheme, since the probability that constraint (3) is violated in a randomly generated population is quite high (will be discussed in detail in Section 5), nullifying the fitness values of the constraint violating chromosomes would not be sensible. Some gene pattern might violate a constraint, but another part of the chromosome might be fit in terms of the primary fitness value. Preventing this chromosome from surviving in the subsequent generations would decrease the diversity. Diversity helps to prevent the algorithm from getting stuck in a local optimum. Therefore, we adopt the mechanisms outlined in (6)–(13) for the fitness function evaluation.

4.2.4. Selection. After the population is sorted by comparison of the fitness values according to Table I, the top $N_{\mathrm{pop}} \times R_{\mathrm{select}}$ number of chromosomes are included in the mating pool, where N_{pop} is the population size and R_{select} is the selection rate, which was chosen to be 0.5 in our work. The mother and father chromosomes are selected from this pool and mated through crossover mechanisms (explained in detail in Section 4.2.5). Each crossover generates two offsprings, which replace two chromosomes from the bottom of the population that is not in the mating pool. These two at a time replacement process continue until all the chromosomes that are not in the mating pool are replaced. This way, the chromosomes that have high fitness, i.e. the ones in the mating

Table I. Pseudocode for fitness comparison.

```
procedure CompareFitness (F_p^α, F_p^β, F_s^α, F_s^β)
      // Return the chromosome having higher fitness value, return 0 if equal
 if F_p^α > F_p^β then
       Return α
 else if F_p^α < F_p^β then
       Return β
 else if F_s^α > F_s^β then
       Return α
 else if F_s^α < F_s^β then
       Return β
 else
       Return 0
end if
```


DOI: 10.1002/dac

pool, survive in the subsequent generations. In contrast, the chromosomes that have low fitness, i.e. the ones that are not in the mating pool, do not survive and are replaced by the offspring of the mating pool, which potentially has higher fitness.

We evaluate the performance of two different selection schemes. The first one is *Roulette Wheel Selection*, also referred to as *Proportional Selection* or *Weighted Random Pairing with Cost Weighting* [24], whereas the second one is *Tournament Selection*.

Roulette wheel selection is a way of choosing mother and father chromosomes from the population in a way that is proportional to their fitness. We have made pertinent modifications to the general Roulette wheel selection framework to accommodate our fitness function evaluation and comparison techniques. In our version of Roulette wheel selection scheme, we choose the mother chromosome according to the algorithm outlined in Table II, where N_p is the number of chromosomes in the mating pool that have nonzero primary fitness value (that do not violate any problem constraint), and N_s denotes the number of chromosomes in the mating pool that have nonzero secondary fitness value (that violate some problem constraint). Furthermore, F_p^{α} and $F_p^{\alpha'}$ denote the primary fitness values of the chromosomes α and α', respectively. Likewise, F_s^{α} and $F_s^{\alpha'}$ denote the secondary fitness values of the chromosomes α and α', respectively. In order to create more diversity, we do not allow the mating of a chromosome with itself. Hence, we select the father chromosome in the same way as the mother, but from a population that excludes the mother.

Tournament selection approach picks a small subset of chromosomes (two or three in general, three in our case) from the mating pool in a uniformly random manner. The chromosome with the highest fitness in this subset becomes a parent, where the fitness values of the chromosomes are compared with each other according to our proposed algorithm outlined in Table I. The tournament repeats for every parent needed. It is computationally simpler than Roulette wheel selection because the population does not need to be sorted [24].

4.2.5. Crossover. Crossover is a genetic operator analogous to reproduction and biological crossover. It is used to vary the programming of chromosomes from one generation to the next. In this paper, we evaluate the performance of three crossover types, namely, single-point crossover, two-point crossover, and uniform crossover.

In single-point crossover, a single crossover point on both parents' strings is selected. All data beyond that point in either organism string are swapped between the two parent organisms. The resulting organisms are the offspring. Two-point crossover requires that two points are selected on the parent organism strings. Everything between the two points is swapped between the parent organisms, rendering two child organisms. In the uniform crossover scheme, individual bits in the string are compared between two parents. The bits are swapped with a fixed probability [24], which we selected as 0.5.

4.2.6. Mutation. A bit within a chromosome is inverted with probability equal to the mutation rate, denoted as μ_m. In our scheme, we mutate every chromosome of the population except for the best

Table II. Pseudocode for Roulette wheel selection.

procedure *RouletteWheel* (*population*)
if $N_p > 0$ **then**
// Select among the chromosomes having $F_p > 0$
Return α with probability $F_p^{\alpha} / \sum_{\alpha'=1}^{N_p} F_p^{\alpha'}$
else
// Select among the chromosomes having $F_p = 0$
Return α with probability $F_s^{\alpha} / \sum_{\alpha'=1}^{N_s} F_s^{\alpha'}$
end if

DOI: 10.1002/dac

chromosome. The exclusion of the best chromosome in mutations is a common practice in GAs, since the best chromosomes are designated as *elite* solutions destined to propagate unchanged [24].

4.2.7. Stopping criteria. We stop the execution of the GA when any of the two following conditions are met: Either the same best solution has been found for N_{best} number of iterations or the maximum number of iterations, $N_{\max}$, has been reached.

5. NUMERICAL EVALUATION

We simulated the first stages of all the schedulers and acquired the U_{if} values in OPNET Modeler 14.0 [26]. In the second stages, we implemented the GA-based suboptimal schedulers in OPNET, whereas we solved the optimization problems in CPLEX [27]. Additive white Gaussian noise (AWGN) channels are considered. The bandwidth, noise variance, path loss, and PU activity models are as in [4]. In all the simulations, each SU has three primary neighbors in its interference range. There are three frequencies with interference temperature thresholds of 1000, 2000, and 3000 K. For the GA-based schedulers, the selection rate $R_{\text{select}} = 0.5$ and $N_{\max} = 5000$. The average values in all of the results were obtained using 10 different seeds and 10 000 schedules for each seed.

The methodology we employ for both throughput maximizing and delay minimizing GA-based suboptimal schedulers is first to evaluate the impact of numerous methods outlined in Table III in a relatively small cognitive radio network consisting of $N = 5$ cognitive nodes. This first evaluation equips us with the knowledge about which set of methods outlined in Table III suits best for our problem. We then evaluate the performance of the schedulers using these determined methods for varying number of cognitive nodes; i.e. $N = 5, 10, \ldots, 30$. This sequential experimental design method of employing a series of smaller experiments each with a specific objective is a common method in experimental design [28] because the experimenter can quickly learn crucial information from a small group of runs that can be used to plan the next experiment. Employing a very large experiment directly in the first steps is usually considered to be a waste of time [28]. In line with this design approach, we initially make a series of experiments using $N = 5$ SUs and observe the impact of different method combinations, and then evaluate the scalability of the system with $N = 5, 10, \ldots, 30$ SUs using the methods that were found to yield better results in the initial experiments.

First, we consider five nodes with $N_{\text{pop}} = 100$, $N_{\text{best}} = 50$, $\mu_{\text{m}} = 0.01$, and single-point crossover. We evaluate the performance of parameter sets defined in Table III. For the throughput maximizing GA-based suboptimal schedulers, the achieved average network throughput and the average number of iterations are stated in Table IV.

First of all, when we compare the results of GA-based solutions (Cases 1–4), we see that all of our GA-based solutions yield almost twice better results than the MFS and PFS schedulers proposed in [4], at the same time being very close to the throughput optimal scheduler's performance.

When we compare the results of Cases 1 and 4, where only the initial population creation method is different, we observe that Case 1 has a slightly larger throughput at the expense of a slightly higher number of iterations, leading to the conclusion that either scheme might be preferred. Since the resulting throughput of both schemes are very close to the optimal one, we prefer to select Case 4 as our candidate parameter set.

When we examine the results of Case 2 and Case 4, where only the encoding type is different, we observe that Case 4 achieves higher throughput as well as a significantly reduced average

Table III. Test case parameters.

Case	Initial population creation	Encoding	Selection
1	RandComp	Type-1	Roulette wheel
2	Rand	Type-2	Roulette wheel
3	Rand	Type-1	Tournament
4	Rand	Type-1	Roulette wheel

DOI: 10.1002/dac

Table IV. Results for throughput maximizing GA scheduler. $N=5$, $N_{pop}=100$, $N_{best}=50$, $\mu_m=0.01$, and single-point crossover.

Case	Average network throughput	Average number of iterations
1	26.58	155.27
2	26.23	171.24
3	26.43	132.77
4	26.57	154.24
Throughput optimal	27.41	—
MFS	14.33	—
PFS	13.74	—

number of iterations. In order to investigate the reason for the superior performance of *Encoding Type-1* over *Encoding Type-2*, note that *Encoding Type-1* has the ability to recognize a gene pattern that does not violate constraint (3) because the genes that correspond to the same time slot and frequency pair are next to each other in this representation. For instance, if a chromosome starts with the bit string [010100...], then $X_{111}=0$, $X_{211}=1$, $X_{311}=0$, and $X_{112}=1$, $X_{212}=0$, $X_{312}=0$ for $N=5$, $F=3$, and $T=5$. This gene pattern does not violate constraint (3) and hence has higher fitness value. In consequence, *Encoding Type-1* enables this kind of a gene pattern to prevail and produce better individuals in the subsequent generations through crossover. Similarly, *Encoding Type-2* is capable of preserving gene patterns that do not violate constraint (2).

Let P_{nv-1}, where nv stands for '*not violate*', denote the probability that constraint (3) is not violated in a randomly generated chromosome for N nodes, F frequencies, and T time slots. If we consider the entire binary string of the chromosome as the concatenation of $F\times T$ string groups, each with N strings encoded according to *Encoding Type-1*, and denote the probability that constraint (3) is not violated in a group by P_{gnv-1}, where gnv stands for '*group not violate*', then $P_{nv-1}=(P_{gnv-1})^{F\times T}$. Besides, probability that constraint (3) is not violated in a group equals the probability that bit '1' occurs at most once in a randomly generated string of length N, which in turn equals $0.5^N+(N\times 0.5^N)=(N+1)/2^N$. Thus, $P_{nv-1}=((N+1)/2^N)^{F\times T}$.

Likewise, let P_{nv-2} denote the probability that constraint (2) is not violated in a randomly generated chromosome of length $N\times F\times T$. If we consider the whole binary string as the concatenation of N string groups each with $F\times T$ strings encoded according to *Encoding Type-2* and denote the probability that constraint (2) is not violated in a group by P_{gnv-2}, then $P_{nv-2}=(P_{gnv-2})^N$. Note that P_{gnv-2} equals the probability that bit '1' occurs at least once in a randomly generated string of length $F\times T$. Therefore, $P_{gnv-2}=1-(0.5)^{F\times T}$ and $P_{nv-2}=(1-(0.5)^{F\times T})^N$. Even for $N=5$, $F=3$, and $T=5$, $P_{nv-1}=1.24\times 10^{-11}$, whereas $P_{nv-2}=0.99$. Therefore, we can conclude that the probability that constraint (3) is violated in a randomly generated chromosome is much higher than the probability that constraint (2) is violated. Whenever (3) is violated, it will drive the primary fitness value of the chromosome to zero; therefore, being able to recognize and track a nonviolating pattern will enable this gene pattern with good traits to prevail in the next generations and thus conduce better fitness values as well as faster convergence. For this reason, *Encoding Type-1* yields superior performance compared with *Encoding Type-2*.

Finally, when we compare the performance of Case 3 and Case 4 in Table IV, where only the selection type differs, we observe that the average number of iterations of tournament selection is less than one of Roulette wheel selection at the expense of a little decrease in average network throughput. Since the difference in throughput is slight, we conclude that faster convergence is more important and hence Case 3 is superior to Case 4. Therefore, in the following simulations, we evaluate Case 3 with different crossover types.

Table V presents the average network throughput and average number of iteration values for Case 3 with single-point, two-point, and uniform crossover, where $N=5$, $N_{pop}=100$, $N_{best}=50$, and $\mu_m=0.01$. We observe that uniform crossover outperforms the other two schemes both in terms of average network throughput and average number of iterations. Therefore, in the following, we use Case 3 with uniform crossover.

DOI: 10.1002/dac

Figure 3 presents the average throughput for the throughput optimal, MFS, and PFS schedulers as well as the throughput maximizing GA-based scheduling scheme with Case 3, uniform crossover, $N=5$, $N_{\text{best}}=50$, $\mu_{\text{m}}=0.01$, and varying N_{pop}. Figure 4, on the other hand, presents the average number of iterations for the throughput maximizing GA-based scheduling scheme with the same parameters. Increasing the population size decreases the average number of iterations

Table V. Crossover-type comparison for throughput maximizing GA scheduler. Case 3, $N=5$, $N_{\text{pop}}=100$, $N_{\text{best}}=50$, $\mu_{\text{m}}=0.01$.

Crossover type	Average throughput	Average number of iterations
Single point	26.43	132.77
Two point	26.44	130.88
Uniform	26.48	127.20
Throughput optimal	27.41	—
MFS	14.33	—
PFS	13.74	—

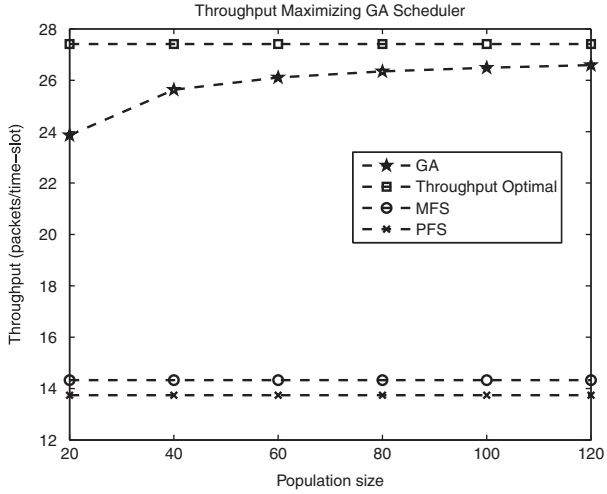

Figure 3. Average network throughput for the GA-based scheduling scheme with Case 3, $N=5$, $N_{\text{best}}=50$, $\mu_{\text{m}}=0.01$, uniform crossover, and varying population size.

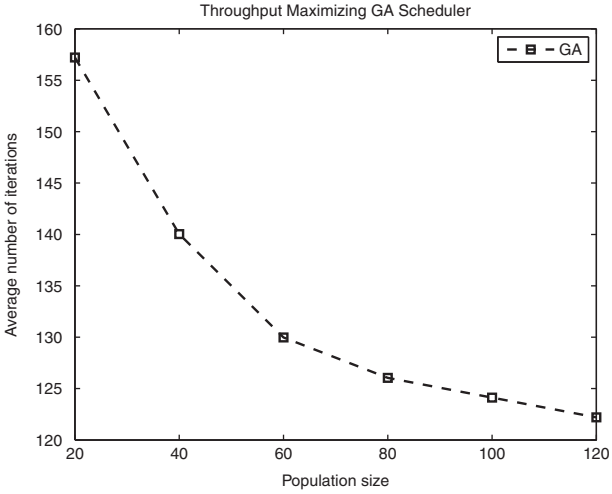

Figure 4. Average number of iterations for throughput maximizing GA-based scheduling scheme with Case 3, $N=5$, $N_{\text{best}}=50$, $\mu_{\text{m}}=0.01$, uniform crossover, and varying population size.

DOI: 10.1002/dac

and increases the throughput; however, the computational cost of a single iteration increases since more chromosomes have to be processed at each iteration. Moreover, the performance improvement decreases as the population size increases. The results indicate that setting $N_{\text{pop}} = 100$ is a reasonable decision in terms of both performance criteria. Furthermore, we can also see that our GA-based approach outperforms the other suboptimal schedulers; i.e. MFS and PFS, with several orders of magnitude, while at the same time yielding a very close performance to the throughput optimal scheduler.

Figure 5 shows the average network throughput for the throughput optimal, MFS, and PFS schedulers as well as the throughput maximizing GA-based scheduling scheme with Case 3, uniform crossover, $N = 5$, $N_{\text{pop}} = 100$, $\mu_{\text{m}} = 0.01$, and varying N_{best}. Figure 6, on the other hand, displays the average number of iterations for the throughput maximizing GA-based scheduling scheme with the same parameters. As N_{best} increases, the rate of increase in throughput decreases after some point, whereas the average number of iterations increases linearly with increasing N_{best}. The results point out that setting $N_{\text{best}} = 50$ is feasible when we take both performance criteria into account.

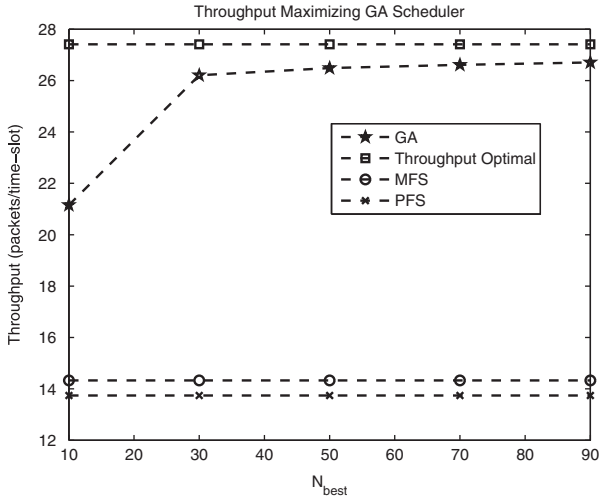

Figure 5. Average network throughput for the GA-based scheduling scheme with Case 3, $N = 5$, $N_{\text{pop}} = 100$, $\mu_{\text{m}} = 0.01$, uniform crossover, and varying N_{best}.

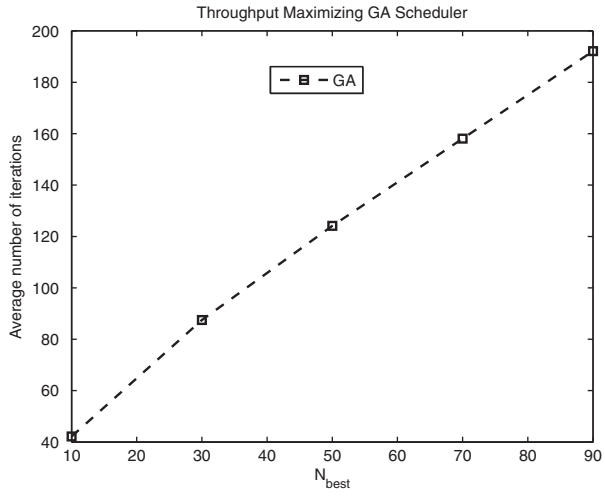

Figure 6. Average number of iterations for the GA-based throughput maximizing scheduling scheme with Case 3, $N = 5$, $N_{\text{pop}} = 100$, $\mu_{\text{m}} = 0.01$, uniform crossover, and varying N_{best}.

DOI: 10.1002/dac

Figure 7 exhibits the average network throughput for the throughput optimal, MFS, and PFS schedulers as well as the throughput maximizing GA-based scheduling scheme with Case 3, uniform crossover, $N=5$, $N_{pop}=100$, $N_{best}=50$, and varying μ_m. Figure 8 shows the average number of iterations for the throughput maximizing GA-based scheduling scheme with the same parameters. Both average throughput and average number of iterations initially increase as μ_m increases; however, they decrease after some point. Setting μ_m to a too high value can result in the introduction of unnecessarily large noise to the current solution; hence, the solution space can even get further away from the good solution area while trying to get out of the local optimum, and thereby yielding even worse performance than the MFS and PFS schedulers. The results indicate that $\mu_m=0.01$ yields throughput very close to the optimal value with a reasonable number of iterations.

Besides, the parameters N_{pop}, N_{best}, and μ_m influence the performance depending on the number of cognitive nodes N. In Table VI, we have outlined the values for these parameters that we empirically found to yield near optimal results for varying values of N. We have used Case 3 and uniform crossover in the simulations of the throughput maximizing GA-based scheduling scheme in Table VI. We can see that our GA-based scheduling scheme yields much better performance

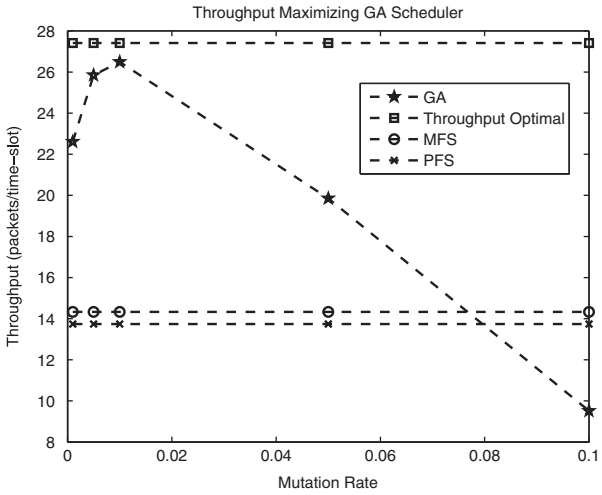

Figure 7. Average network throughput for the GA-based scheduling scheme with $N=5$, $N_{pop}=100$, $N_{best}=50$, and varying μ_m.

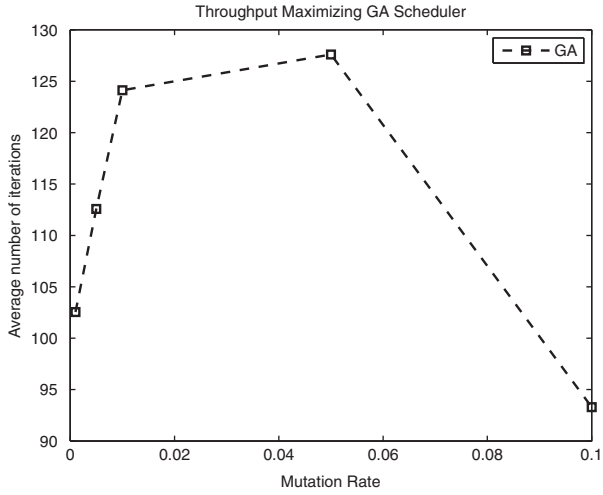

Figure 8. Average number of iterations for the GA-based scheduling scheme with $N=5$, $N_{pop}=100$, $N_{best}=50$, and varying μ_m.

DOI: 10.1002/dac

than the MFS and PFS schedulers, and very close performance to the optimal scheduler for all $N = 5, 10, \ldots, 30$. Note that the parameters in this table are set so that the suboptimal scheduler gives very close results to the optimal one. If less number of iterations are desired at the expense of reduced throughput, then the N_{pop}, N_{best}, and μ_{m} parameters can be adjusted depending on the number of SUs N, with Table VI serving as a baseline.

We have also evaluated the performance of the GA-based suboptimal scheduler that minimizes the scheduling delay. Initially, we evaluated the performance of the parameter sets defined in Table III. The resulting average scheduling delay and the average number of iteration values of the delay minimizing GA scheduler for $N = 5$, $N_{\text{pop}} = 100$, $N_{\text{best}} = 75$, $\mu_{\text{m}} = 0.01$, and single-point crossover are shown in Table VII. The results indicate that the GA-based schedulers yield much better scheduling delay performance than the MFS and the PFS schedulers, while at the same time providing close to optimal performance. Furthermore, the results also indicate that Case 3 conduces the least scheduling delay and the least number of iterations. Hence, we employ Case 3 in the subsequent simulations.

Table VIII presents the average scheduling delay and the average number of iteration values for $N = 5$, $N_{\text{pop}} = 100$, $N_{\text{best}} = 75$, $\mu_{\text{m}} = 0.01$ with Case 3 and single-point, two-point, as well as uniform crossovers. The results show that the GA-based schedulers again have far better performance than the MFS and the PFS schedulers, at the same time having very close to optimal performance. The results also reveal that uniform crossover outperforms the other two schemes both in terms of the average scheduling delay and the average number of iterations. Therefore, in the sequel, we use Case 3 with uniform crossover.

Table VI. Parameter settings for throughput maximizing GA scheduler with varying number of cognitive nodes.

N	5	10	15	20	25	30
N_{pop}	100	150	200	300	400	500
N_{best}	50	75	200	300	400	500
μ_{m}	0.01	0.01	0.01	0.001	0.001	0.001
Average throughput	26.48	26.87	25.61	25.89	26.12	25.99
Average number of iterations	127.20	330.99	885.22	1954.77	3076.82	5084.88
Throughput optimal	27.41	27.55	26.29	26.46	26.81	26.68
MFS	14.33	15.16	16.06	16.81	16.76	16.51
PFS	13.74	14.50	15.97	16.01	16.46	16.47

Table VII. Results for delay minimizing GA scheduler for $N = 5$, $N_{\text{pop}} = 100$, $N_{\text{best}} = 75$, $\mu_{\text{m}} = 0.01$.

Case	Average scheduling delay	Average number of iterations
1	0.039	94.43
2	0.038	92.55
3	0.023	87.24
4	0.040	92.29
Delay optimal	0.00095	—
MFS	0.60	—
PFS	0.55	—

Table VIII. Crossover-type comparison for delay minimizing GA scheduler for $N = 5$, $N_{\text{pop}} = 100$, $N_{\text{best}} = 75$, $\mu_{\text{m}} = 0.01$ with Case 3.

Crossover type	Average delay	Average number of iterations
Single point	0.023	87.24
Two point	0.022	86.68
Uniform	0.0099	82.38
Delay optimal	0.00095	—
MFS	0.60	—
PFS	0.55	—

DOI: 10.1002/dac

Table IX presents the average scheduling delay for the delay optimal, MFS, and PFS schedulers as well as the delay minimizing GA-based scheduling scheme with Case 3, uniform crossover, $N=5$, $N_{best}=75$, $\mu_m=0.01$, and varying N_{pop}. The table also shows the average number of iterations for the GA-based schemes. The results indicate that the GA-based scheduling scheme results in much better performance than the MFS and the PFS schedulers, even with a low population size. Moreover, the scheduling delay and the average number of iterations decrease as N_{pop} increases; however, the computational cost of a single generation also increases. Furthermore, the rate of decrease in the average number of iterations decreases as N_{pop} increases, whereas the average scheduling delay values diminish almost linearly. The results point out that setting $N_{pop}=100$ is a reasonable decision in terms of both performance criteria.

Table X shows the average scheduling delay for the delay optimal, MFS, and PFS schedulers as well as the delay minimizing GA-based scheduling scheme with Case 3, uniform crossover, $N=5$, $N_{pop}=100$, $\mu_m=0.01$, and varying N_{best}. The table also shows the average number of iterations for the GA-based schemes. The results reveal that the GA-based scheduling scheme yields far better scheduling delay performance than the MFS and the PFS schedulers even with small values of N_{best}. We can also see that the scheduling delay decreases as N_{best} increases. We can also see in Table X that the average number of iterations increases as N_{best} increases. As in the throughput maximizing GA scheduler, the increase in the average number of iterations is linear as N_{best} increases. The results point out that setting $N_{best}=75$ is feasible when we take both performance criteria into account.

Table XI shows the average scheduling delay for the delay optimal, MFS, and PFS schedulers as well as the delay minimizing GA-based scheduling scheme with Case 3, uniform crossover, $N=5$, $N_{pop}=100$, $N_{best}=75$ and varying μ_m. Again, the results show that the scheduling delay performance of the GA-based scheme is much better than the ones of MFS and PFS, while at the same time being close to the optimal delay performance. Table XI also shows the average number of iterations for the GA-based scheme with the same parameters. We can see that both the average scheduling delay and the average number of iterations initially decrease as μ_m increases; nevertheless, they

Table IX. Average scheduling delay and average number of iterations for the GA-based scheme with Case 3, uniform crossover, $N=5$, $N_{best}=75$, $\mu_m=0.01$, and varying population size.

Scheduler type	Average delay	Average number of iterations
Delay minimizing GA, $N_{pop}=20$	0.073	242.29
Delay minimizing GA, $N_{pop}=40$	0.056	93.38
Delay minimizing GA, $N_{pop}=60$	0.034	88.03
Delay minimizing GA, $N_{pop}=80$	0.018	85.37
Delay minimizing GA, $N_{pop}=100$	0.009	82.38
Delay minimizing GA, $N_{pop}=120$	0.0009	81.51
Delay optimal	0.00095	—
MFS	0.6	—
PFS	0.55	—

Table X. Average scheduling delay and average number of iterations for the GA-based scheduling scheme with Case 3, uniform crossover, $N=5$, $N_{pop}=100$, $\mu_m=0.01$, and varying N_{best}.

Scheduler type	Average delay	Average number of iterations
Delay minimizing GA, $N_{best}=15$	0.0108	23.89
Delay minimizing GA, $N_{best}=30$	0.0107	39.00
Delay minimizing GA, $N_{best}=45$	0.0106	53.93
Delay minimizing GA, $N_{best}=60$	0.0104	68.98
Delay minimizing GA, $N_{best}=75$	0.0099	84.12
Delay minimizing GA, $N_{best}=90$	0.0096	99.2
Delay optimal	0.00095	—
MFS	0.6	—
PFS	0.55	—

DOI: 10.1002/dac

Table XI. Average scheduling delay and average number of iterations for the GA-based scheduling scheme with Case 3, uniform crossover, $N=5$, $N_{pop}=100$, $N_{best}=75$, and varying μ_m.

Scheduler type	Average delay	Average number of iterations
Delay minimizing GA, $\mu_m=0.001$	0.019	89.82
Delay minimizing GA, $\mu_m=0.005$	0.015	84.81
Delay minimizing GA, $\mu_m=0.01$	0.009	84.12
Delay minimizing GA, $\mu_m=0.05$	0.012	87.74
Delay minimizing GA, $\mu_m=0.1$	0.02	94.87
Delay minimizing GA, $\mu_m=0.2$	0.0345	98.24
Delay optimal	0.00095	—
MFS	0.6	—
PFS	0.55	—

Table XII. Parameter settings for delay minimizing GA scheduler with varying number of cognitive nodes.

N	5	10	15	20	25	30
N_{pop}	100	150	200	250	300	350
N_{best}	75	100	150	200	250	300
μ_m	0.01	0.01	0.01	0.001	0.001	0.001
Average delay	0.0099	0.212	0.538	1.11	1.60	2.11
Average number of iterations	84.12	132.27	316.97	613.34	987.14	1112.45
Delay optimal	0.00095	0.203	0.509	1.01	1.51	2.01
MFS	0.60	1.27	1.96	2.66	3.41	4.09
PFS	0.55	1.23	1.87	2.55	3.21	3.94

both increase after some point. The results indicate that $\mu_m=0.01$ yields reasonable performance in terms of both criteria when we compare it with the other mutation rates.

Besides, as in the throughput maximizing GA scheduler, the parameters N_{pop}, N_{best}, and μ_m influence the performance depending on the number of cognitive nodes N. In Table XII, we have outlined the values for these parameters that we empirically found to yield satisfactory results with reasonable number of iterations for varying values of N. We have used Case 3 and uniform crossover in the simulations in Table XII.

All in all, both of our proposed GA-based schedulers achieve very close performance to their optimal scheduler counterparts while at the same time operating with much lower complexities. However, the average number of iterations in the simulation results reveal that our GA-based schedulers are computationally more costly than the MFS and the PFS schedulers in [4]. Nevertheless, when they are compared with respect to the throughput and delay performance, we can see that our GA-based schedulers are approximately twice better than the MFS and the PFS schedulers. Moreover, our GA-based schedulers are computationally more efficient than the classical branch and bound algorithms that are used to solve binary integer programming problems [29, 30]. Therefore, our GA-based schedulers present a very reasonable tradeoff between computational complexity and performance, hence addressing the open research issue identified by Gözüpek and Alagöz [4]. Hence, we can conclude that our GA-based schedulers are more suitable for slowly varying spectral environments, whereas MFS and PFS schedulers are more suitable for very swiftly changing spectral environments. Considering that IEEE 802.22 networks [22] operate on the TV broadcast bands that are slowly varying, we can confidently conclude that our GA-based schedulers can operate in realistic network settings, and provide useful solutions to the open research problem pinpointed by Gözüpek and Alagöz [4].

6. CONCLUSION

Recently, throughput and delay optimal schedulers for cognitive radio networks under interference temperature constraints have been proposed in the literature [4]. The common features of these schedulers is that they both ensure that the interference temperature constraints of the PUs are

DOI: 10.1002/dac

not violated, no collisions occur among the SUs, and reliable communication of the SUs with the cognitive base station is achieved. In this paper, we propose genetic algorithm (GA)-based suboptimal schedulers for the throughput and delay optimal scheduling problems in [4]. Our proposed GA-based schedulers alleviate the computational complexity drawback of the optimal schedulers in [4]. Specifically, we formulate GA-based algorithms and chromosome encoding methods as well as fitness function evaluation and comparison techniques for both schedulers. We compare the performance of different selections, crossovers, encodings, and initial population creation techniques, as well as different values of population size (N_{pop}), maximum number of consequent generations for finding the same best solution in order for convergence to occur (N_{best}), and mutation rate (μ_{m}). We also evaluate the scalability of our solution by using feasible parameter values for different number of cognitive nodes (N).

The simulation results show that our GA-based throughput maximizing and delay minimizing schedulers work best with uniformly random generation of each bit of the chromosomes during initial population creation, tournament selection, and uniform crossover. Furthermore, our proposed GA-based suboptimal schedulers yield close to optimal performance with a reasonable number of iterations, while at the same time resulting in much better performance than MFS and PFS, which are the suboptimal schedulers proposed by Gözüpek and Alagöz [4], to solve the same problems.

Considering that our GA-based schedulers in this paper work best in slowly varying spectral environments, the design of schedulers whose performance is as good as the GA-based schedulers while at the same time being suitable for swiftly changing spectral environments is an open research issue. We are currently investigating graph theory-based solution techniques to address this issue.

REFERENCES

1. No F. 03-222 notice of proposed rule making and order, 2003.
2. Mitola J. Cognitive radio: an integrated agent architecture for software defined radio. *Doctor of Technology*, Royal Institute of Technology (KTH), Stockholm, Sweden 2000.
3. Akyildiz I, Lee W, Vuran M, Mohanty S. Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *Computer Networks* 2006; **50**(13):2127–2159.
4. Gözüpek D, Alagöz F. Throughput and delay optimal scheduling in cognitive radio networks under interference temperature constraints. *Journal of Communications and Networks* (*JCN*) 2009; **11**(2):147–155.
5. Habib I, Sherif M, Naghshineh M, Kermani P. An adaptive quality of service channel borrowing algorithm for cellular networks. *Wiley's International Journal of Communication Systems* (*IJCS*) 2003; **16**(8):759–777.
6. Sandalidis H, Stavroulakis P, Rodriguez-Tellez J. Comparison of two novel heuristic dynamic channel allocation techniques in cellular systems. *Wiley's International Journal of Communication Systems* (*IJCS*) 1998; **11**(6): 379–386.
7. Rondeau T, Le B, Rieser C, Bostian C. Cognitive radios with genetic algorithms: intelligent control of software defined radios. *Proceedings of SDR Forum Technical Conference*, Orlando, FL, U.S.A., 2006.
8. Kim J, Sohn S, Han N, Zheng G, Kim Y, Lee J. Cognitive radio software testbed using dual optimization in genetic algorithm. *Proceedings of International Conference on Cognitive Radio Oriented Wireless Networks and Communications* (*CROWNCOM*), 2008; 1–6.
9. Friend D, Elnainay M, Shi Y, Mackenzie A. Architecture and performance of an island genetic algorithm-based cognitive networks. *Proceedings of IEEE Consumer Communications and Networking Conference* (*CCNC'08*), Las Vegas, NV, U.S.A., 2008; 993–997.
10. Newman T, Rajbanshi R, Wyglinski A, Evans J, Minden G. Population adaptation for genetic algorithm-based cognitive radios. *Mobile Networks and Applications* 2008; **13**(5):442–451.
11. Knopp R, Humblet P. Information capacity and power control in single-cell multiuser communications. *IEEE International Conference on Communications* (*ICC'95*), Seattle, WA, U.S.A., vol. 1, 1995.
12. Viswanath P, Tse D, Laroia R. Opportunistic beamforming using dumb antennas. *IEEE Transactions on Information Theory* 2002; **48**(6):1277–1294.
13. Andrews M, Kumaran K, Ramanan K, Stolyar A, Whiting P, Vijayakumar R. Providing quality of service over a shared wireless link. *Communications Magazine*, *IEEE* 2001; **39**(2):150–154.
14. Zhou C, Wunder G. A novel low delay scheduling algorithm for OFDM broadcast channel. *IEEE Global Telecommunications Conference* (*GLOBECOM'07*), Washington, DC, U.S.A., 2007; 3709–3713.
15. Chaporkar P, Kar K, Luo X, Sarkar S. Throughput and fairness guarantees through maximal scheduling in wireless networks. *IEEE Transactions on Information Theory* 2008; **54**(2):572–594.
16. Jung K, Shah D. Low delay scheduling in wireless network. *IEEE International Symposium on Information Theory* (*ISIT'07*), Nice, France, 2007; 1396–1400.
17. Wang W, Wang W, Lu Q, Peng T. An uplink resource allocation scheme for OFDMA-based cognitive radio networks. *Wiley's International Journal of Communication Systems* (*IJCS*) 2009; **22**(5):603–623.

DOI: 10.1002/dac

18. Li J, Xu B, Xu Z, Li S, Liu Y. Adaptive packet scheduling algorithm for cognitive radio system. *International Conference on Communication Technology* (*ICCT'06*), Guilin, China, 2006; 1–5.
19. Urgaonkar R, Neely M. Opportunistic scheduling with reliability guarantees in cognitive radio networks. *Proceedings of IEEE INFOCOM*, Phoenix, AZ, U.S.A., 2008; 1301–1309.
20. Thoppian M, Venkatesan S, Prakash R, Chandrasekaran R. Mac-layer scheduling in cognitive radio based multi-hop wireless networks. *Proceedings of the International Symposium on World of Wireless*, *Mobile and Multimedia Networks*. IEEE Computer Society: Washington, DC, U.S.A., 2006; 191–202.
21. No F. 03-289 notice of inquiry and proposed rulemaking, 2003.
22. Web page: http://www.ieee802.org/22/.
23. Holland J. *Adaptation in Natural and Artificial Systems*. University of Michigan Press: Ann Arbor, MI, 1975.
24. Haupt R, Haupt S. *Practical Genetic Algorithms* (2nd edn). Wiley: New York, 2004.
25. Fonseca C, Fleming P. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation. *IEEE Transactions on Systems*, *Man and Cybernetics*, *Part A* 1998; **28**(1):26–37.
26. Inc O. Opnet modeler. Web page: http://www.opnet.com.
27. Cplex. Web page: http://www.ilog.com/products/cplex.
28. Montgomery D. *Design and Analysis of Experiments*. Wiley: New York, 2006.
29. Land A, Doig A. An automatic method for solving discrete programming problems. *Econometrica* 1960; **28**(3):497–520.
30. Sherali H, Myers D. The design of branch and bound algorithms for a class of nonlinear integer programs. *Annals of Operations Research* 1986; **5**(1–4):463–484.

AUTHORS' BIOGRAPHIES

Didem Gözüpek received her BS degree (high honors) in Telecommunications Engineering from Sabanci University, Istanbul, Turkey, in 2004 and the MS degree in Electrical Engineering from the New Jersey Institute of Technology (NJIT), U.S.A. in 2005. During her MS studies, she was a research assistant in the Broadband, Mobile and Wireless Networking Laboratory, NJIT. From 2005 to 2008 she worked as an R&D engineer for Argela Technologies, Istanbul. Currently, she is a researcher at Telematics & Informatics Research Center (TAM) and a PhD candidate in Computer Engineering, Bogazici University, Istanbul. She has been selected as a finalist for the Google Anita Borg Memorial Scholarship in the EMENA (Europe, Middle East, and North Africa) region in 2009. Her main research interests are in the field of wireless networks, in particular, scheduling, media access control, radio resource management, cognitive radio networks, and wireless *ad hoc* network applications. She is a member of the IEEE and IEEE Communications Society.

Fatih Alagöz is an Associate Professor in the Department of Computer Engineering, Bogazici University, Turkey. From 2001 to 2003, he was with the Department of Electrical Engineering, United Arab Emirates University, UAE. In 1993, he was a research engineer in a missile manufacturing company, Muhimmatsan AS, Turkey. He received the BSc degree in Electrical Engineering from the Middle East Technical University, Turkey, in 1992, and MSc and DSc degrees in Electrical Engineering from the George Washington University, U.S.A., in 1995 and 2000, respectively. His current research interests are in the areas of satellite networks, wireless networks, sensor networks, UWB communications, and cognitive radio networks. He has contributed/managed to 10 research projects for various agencies/organizations including the US Army of Intelligence Center, Naval Research Laboratory, UAE Research Fund, Turkish Scientific Research Council, State Planning Organization of Turkey, BAP, etc. He has edited five books and published more than 100 scholarly papers in selected journals and conferences. Dr Alagöz is the Satellite Systems Advisor to the Kandilli Earthquake Research Institute, Istanbul, Turkey. He has served on several major conference technical committees, and organized and chaired many technical sessions in many international conferences. He is a member of the IEEE Satellite and Space Communications Technical Committee. He has numerous professional awards.

DOI: 10.1002/dac