Throughput Satisfaction-Based Scheduling for Cognitive Radio Networks

Didem Gözüpek, Başak Eraslan, and Fatih Alagöz

***Abstract*—Opportunistic scheduling algorithms in cognitive radio networks (CRNs) allocate resources by exploiting the variations in channel conditions and spectral activities of primary users. However, most of these scheduling algorithms ignore the per-user throughput requirements. In this paper, we formulate a scheduling problem called maximizing the number of satisfied users (MNSU), which maximizes the number of secondary users that are satisfied in terms of throughput in a centralized CRN. We show that MNSU is NP-hard in the strong sense and cannot be approximated within any constant factor better than 2 unless $P = NP$. We also prove that MNSU is at least as hard as the max–min fair scheduling problem, which has previously been proven to be a computationally very difficult problem in the literature. We then propose two heuristic algorithms: 1) best first resource assignment and 2) resource assignment with partial backtracking. We demonstrate that our proposed algorithms yield high performance while still achieving low computational complexity.**

***Index Terms*—Cognitive radio networks (CRN), dynamic spectrum access (DSA), resource allocation, scheduling.**

I. Introduction

THE dynamic spectrum access (DSA) concept aims to enhance spectrum utilization by having intelligent devices called cognitive radios (CR). These devices can opportunistically utilize portions of the spectrum that are temporarily unused by their licensed owners. Licensed owners are called primary users (PUs), whereas CR devices are called secondary users (SUs) [1].

The opportunistic scheduling paradigm in wireless networks relies on the idea of exploiting the inherently time-varying fluctuations in the wireless environment to achieve a certain objective, such as maximizing the total throughput. The opportunistic nature of the DSA concept lends itself conveniently to the opportunistic scheduling paradigm since it enables the SUs to utilize from the time-varying spectral activities of the PUs. Existing opportunistic scheduling works in CR networks (CRNs) mostly focus on maximizing throughput [2], [3] or providing fairness among the users [4], [5]. However, in a practical scenario, each SU has a minimum throughput requirement below which the user is not satisfied. The value of this minimum throughput requirement may depend on the application executed by the SU. For instance, delay sensitive real-time applications necessitate a higher minimum data rate. In other words, providing a data rate lower than the minimum required value is useless for this SU. Resources can instead be assigned to another SU so that its minimum data rate requirement can be met. Likewise, allocating more than the required data rate to an SU does not bring additional benefit since these excess resources can be assigned to some other SU. In practice, the primary goal of a cognitive base station (CBS) operator is to maximize the number of satisfied (happy) SUs. To the best of our knowledge, this problem has not been previously addressed in the CRN literature.

We outline the abbreviations used throughout this paper in Table I in alphabetical order. The remainder of this paper is organized as follows. Section II describes the related work, whereas Section III presents maximizing the number of satisfied users (MNSU) problem formulation. We prove in Section IV that MNSU is NP-hard in the strong sense and cannot be approximated within any constant factor better than 2 unless $P = NP$. This proof implies that it is highly unlikely to find a polynomial time algorithm that guarantees that the throughput requirements of at least half of the SUs are met. Inapproximability proofs are important since they demonstrate the impossibility of finding an algorithm with a certain provable worst-case performance guarantee for all instances of that problem. Finding an inapproximability result for a problem before proceeding to design a heuristic algorithm is important because it not only justifies the usage of a heuristic algorithm but also gives information about the computational difficulty inherent in the combinatorial nature of the problem. Moreover, we also prove that MNSU is at least as hard as the max–min fair scheduling (MMFS) problem, which has previously been proven to be a computationally very difficult problem in the literature. We discuss our proposed algorithms in Section V and present the simulation results in Section VI. Finally, we conclude this paper in Section VII.

II. Related Work

Maximizing user satisfaction was addressed by works about scheduling in various areas of wireless networks. For instance, the authors in [6] proposed a resource allocation algorithm to maximize user satisfaction in orthogonal frequency-division multiple-access (OFDMA) systems. Their algorithm assigns subcarriers to the users in an OFDMA system while maximizing the number of satisfied users, where a user is satisfied if the average data rate it experiences is greater than or equal to its

Manuscript received December 16, 2011; revised April 26, 2012 and June 10, 2012; accepted July 2, 2012. Date of publication July 25, 2012; date of current version November 6, 2012. This work was supported in part by the State Planning Organization of Turkey (DPT) under Grant DPT-2007K 120610 and in part by the Scientific and Technological Research Council of Turkey under Grant 109E256. The review of this paper was coordinated by Prof. N. Kato.

The authors are with the Department of Computer Engineering, Bogazici University, Istanbul 34342, Turkey (e-mail: didem.gozupek@boun.edu.tr; basak.elyildirim@boun.edu.tr; fatih.alagoz@boun.edu.tr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2012.2210257

TABLE I
TABLE FOR ABBREVIATIONS (ALPHABETICAL ORDER)

Abbreviation	Explanation
BFRA	Best First Resource Assignment
BILP	Binary Integer Linear Program
CBS	Cognitive Base Station
CR	Cognitive Radio
CRN	Cognitive Radio Network
DSA	Dynamic Spectrum Access
ERQ	Excess Resource Queue
FTRP	Frequency and Time slot Resource Pair
ILP	Integer Linear Program
MMFS	Max-Min Fair Scheduling
MNSU	Maximizing the Number of Satisfied Users
PN	Packet Need
PU	Primary User
RAPB	Resource Assignment With Partial Backtracking
RQ	Resource Queue
SU	Secondary User

average data rate requirement. Their two-step algorithm provides a suboptimal solution to their formulated problem. In [7], users' satisfaction in OFDMA is addressed, albeit with a different approach. Instead of formulating a different optimization problem, they proposed a dynamically configurable framework that combines maximum rate, max–min fairness, and proportional fairness policies to maximize the number of satisfied users. On the other hand, in [8], opportunistic scheduling policies with utilitarian fairness for orthogonal frequency-division-multiplexing (OFDM) systems, where utilitarian fairness means that the scheduling scheme aims to ensure that all users get at least a certain fraction of the overall system performance, is derived. Moreover, the scheduling approach proposed in [9] aimed to maximize the number of satisfied users in a multirate wireless system. Since their formulated problem is NP-complete, they presented an approximation algorithm based on two-phase combinatorial reverse auction. Our proposed scheduler in this paper has resemblance also with OFDM systems. In fact, OFDM is seen as a promising candidate technology for future CRNs [10]. In this respect, our work is different from [7] and [8] since we propose an optimization problem to maximize the number of satisfied users, whereas they mainly focus on adaptive resource allocation considering fairness constraints. In addition, our work is also different from the work in [6] since we focus on a time-slotted frame-based scheduling scheme while, at the same time, considering the number of antennas of the users, whereas the work in [6] lacks all of these features.

Users' satisfaction was also addressed in the realm of code-division multiple-access (CDMA) networks. The authors in [11] formulated a resource allocation framework for CDMA cellular networks supporting multimedia services. A user's utility characterizes its degree of satisfaction (happiness) with the received service. By using network utility maximization theory, they proposed a method to dynamically adapt the users' utility functions. Additionally, the authors in [12] aimed at maximizing users' utilities in CDMA networks by considering the probabilistic short-term throughput requirements of real-time services and long-term throughput requirements of nonreal-time services. They demonstrated that their joint scheduling approach yields substantial performance improvements. Moreover, the work in [13] concentrated on the downlink of a CDMA cellular network and proposed a scheduling algorithm to maximize the number of satisfied users, where a user is satisfied if its packet delay is below a certain threshold.

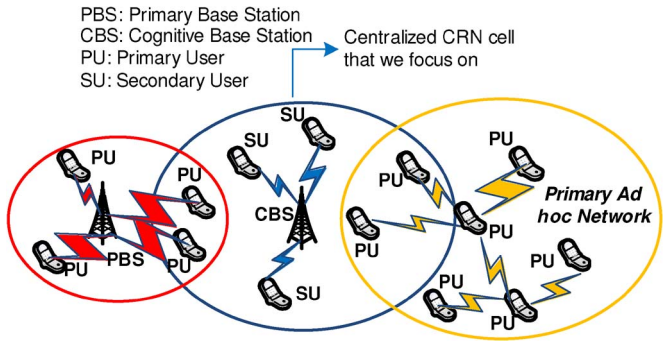

Fig. 1. Considered centralized CRN architecture [22].

Maximizing user satisfaction in terms of throughput received little attention in CRNs. In [14], a scheduling mechanism that maximizes the sum of utility functions of the source flow rates of origin–destination pairs in an ad hoc CRN is provided. Unlike our work, they concentrated on the scheduling of flows rather than packets. Moreover, their focus was on ad hoc CRNs, whereas we concentrate on centralized CRNs. Furthermore, our scheduling mechanism not only determines data rates but also frequencies and time slots. Moreover, the authors in [15] focused on a cooperation-based spectrum leasing scenario where they maximized the total expected utility while satisfying an average performance constraint for each primary node in the network. Unlike our work, their focus was on cooperation, and time slot was the only resource that needed to be shared between SUs and PUs. Furthermore, PUs and SUs in their scenario communicate with a central entity, whereas in our work, only SUs communicate with a central entity (CBS). Maximizing user satisfaction received some interest in CRNs within the power allocation context [16], [17]. On the other hand, opportunistic scheduling works in CRNs mainly aimed to maximize spectrum utilization [18], total throughput, or achieve fairness [19]–[21]. To the best of our knowledge, existing opportunistic scheduling algorithms in CRNs do not aim to maximize the number of SUs whose minimum throughput requirements have been met.

III. PROBLEM FORMULATION

In our previous work [22], we formulated a scheduling problem that maximizes the total average throughput of all SUs in a CRN cell while at the same time ensuring that reliable communication of the SUs with the centralized CBS is maintained, no collisions occur among the SUs, and the PUs are not disturbed. The centralized time-slotted CRN cell that the scheduling framework in [22] focuses on is illustrated in Fig. 1. In [22], the data transmission of SUs to the CBS is focused on, where SUs possibly have multiple antennas for data transmission. The first step of the solution in [22] consists of determining the values for U_{if}, which stands for the maximum number of packets that can be sent by SU i using frequency f in any time slot during the entire scheduling period consisting of T time slots. These values are determined such that the

transmission power of the SUs does not violate the maximum tolerable interference power limits of the PUs, and reliable communication of the SUs with the CBS is maintained. The following formula derived in [22] accomplishes these goals: $U_{if} = \lfloor \ln(1 + P_{IF_{\max}} \times (|G_{i0}|/|G_{if}| \times \sigma)^2) \rfloor$, where $P_{IF_{\max}}$ stands for the maximum tolerable interference power of the Pus, and $|G_{i0}|$ is the channel gain between SU i and the CBS. Moreover, $|G_{if}|$ is the maximum channel gain among all channel gains between SU i and the PUs that are actively using frequency f in the scheduling period, and σ^2 is the noise variance. Moreover, it is assumed for simplicity and yet without loss of generality (w.l.o.g.) that $S = B \times T_s$, where S is the packet size, B is the bandwidth, and T_s is the time slot length. Interference and spectrum availabilities are considered in the derivation of the formula for U_{if}. For instance, if an SU i is very close to a PU that uses frequency f so that the SU i cannot use frequency f at all, then the derivation of the formula for U_{if} ensures that $U_{if} = 0$. In other words, U_{if} values quantify the availability of frequency f for SU i, i.e., the higher the U_{if} value is, the higher the data rate SU i can have if it uses frequency f. U_{if} values are input variables for the optimization problems in this paper.

In [22], it is assumed w.l.o.g. that $P_{IF_{\max}}$ is the same for all PUs and frequencies. This assumption can easily be relaxed so that the maximum tolerable interference power depends on the frequency and the PU. The analysis in [22] can be modified as follows.

We initially find the maximum permissible transmission power for each SU i and frequency f in time slot t, which is represented here by P_{xmt}^{ift}. We use free space path loss and fading in modeling the channels between SUs and PUs, as well as the ones between CBS and SUs. We derive the following expression for P_{xmt}^{ift}:

$$P_{xmt}^{ift} = \min_{j \in \Phi_{CBS}^{ft}} \frac{P_{IF_{\max}}^{fj}}{\left(\frac{\lambda_f}{4\pi d_{ijt}} \times |h_{ijt}|\right)^2} \tag{1}$$

where $P_{IF_{\max}}^{fj}$ represents the maximum tolerable interference power of PU j for frequency f, d_{ijt} is equal to the distance between SU i and PU j in time slot t, λ_f is the wavelength of frequency f, and Φ_{CBS}^{ft} symbolizes the set of PUs that are in the coverage area of the CBS and that are carrying out their communication using frequency f in time slot t. Moreover, h_{ijt} denotes the fading coefficient of the channel between SU i and PU j in time slot t. In particular, $(\lambda_f/4\pi d_{ijt})^2$ refers to the path loss of the channel between SU i and PU j for frequency f in time slot t as a result of the free space path loss formula. PU j that has the minimum value of $(P_{IF_{\max}}^{fj}/(\lambda_f/4\pi d_{ijt}) \times |h_{ijt}|)^2)$ is the one that is most severely affected by the data transmission of SU i using frequency f in time slot t. Intuitively, this corresponds to a PU that has low interference tolerance and good channel conditions (such as being close in proximity to the SU). If this PU j is guaranteed not to be affected by this SU transmission, other PUs will also not be disturbed. Therefore, the maximum transmission power of SU i using frequency f in time slot t is set to $\min_{j \in \Phi_{CBS}^{ft}} (P_{IF_{\max}}^{fj}/(\lambda_f/4\pi d_{ijt}) \times |h_{ijt}|)^2)$.

As in [22], let us assume for simplicity, and yet w.l.o.g., that $S = B \times T_s$. Recall that P_{xmt}^{ift} denotes the maximum permissible transmission power for SU i and frequency f in time slot t. Therefore, the following hold:

$$G_{ift}^{CBS} \triangleq \frac{\lambda_f}{4\pi d_{it}^{CBS} \times h_{it}^{CBS}} \tag{2}$$

$$P_{r_{CBS}}^{ift} = P_{xmt}^{ift} \times \left|G_{ift}^{CBS}\right|^2 \tag{3}$$

$$U_{ift} = \left\lfloor \ln\left(1 + \left(P_{xmt}^{ift} \times \frac{\left|G_{ift}^{CBS}\right|^2}{\zeta}\right)\right)\right\rfloor \tag{4}$$

where d_{it}^{CBS} is the distance between SU i and the CBS in time slot t, h_{it}^{CBS} denotes the fading coefficient of the channel between SU i and the CBS in time slot t, and U_{ift} is the maximum number of packets that can be transmitted by SU i using frequency f in time slot t. Equation (3) relates the power that would be received by the CBS due to the transmission of SU i using frequency f in time slot t ($P_{r_{CBS}}^{ift}$) if SU i uses its maximum permissible transmission power for frequency f and time slot t (P_{xmt}^{ift}). Finally, (4) is due to Shannon's capacity function for Gaussian channels, where $P_{r_{CBS}}^{ift}$ is replaced by (3), and ζ represents the sum of noise power and the interference power from the PUs to the SUs. The floor operation $\lfloor \cdot \rfloor$ in (4) is necessary since U_{ift} can naturally only take integer values.

The work in [22] assumes that the network conditions, i.e., PU and SU locations, PU spectrum occupancies, and all the channel fading coefficients, are small enough not to have any impact on the U_{ift} values for a duration of T time slots in the considered centralized CRN cell; in other words, we can use U_{if} instead of U_{ift}. A scheduling period consists of T time slots. The value of T, in general, depends on the characteristics of the spectrum environment. For instance, a slowly varying spectrum environment like the TV broadcast bands utilized by an IEEE 802.22 network allows T to have a fairly large value. In particular, IEEE 802.22 mandates that SUs need to vacate a spectrum band within 2 s of the appearance of a PU in that band. As in [22], in the simulations part of this work, we have taken $T = 1$ s (consisting of ten time slots, where each time slot is $T_s = 100$ ms), which is sufficient for the proper operation of IEEE 802.22 networks.

Given the U_{if} values determined in the first stage, the work in [22] solves in the second stage the following binary integer linear program (ILP) to maximize the total throughput:

$$\max \left(\sum_{i=1}^{N} \sum_{f=1}^{F} \sum_{t=1}^{T} \frac{U_{if} X_{ift}}{T} \right) \tag{5}$$

$$s.t.$$

$$\sum_{f=1}^{F} \sum_{t=1}^{T} X_{ift} \geq 1; \forall\, i \in \mathcal{N} \tag{6}$$

$$\sum_{i=1}^{N} X_{ift} \leq 1; \forall\, f \in \mathcal{F}, \forall\, t \in \mathcal{T} \tag{7}$$

$$\sum_{f=1}^{F} X_{ift} \leq a_i; \forall\, i \in \mathcal{N}, \forall\, t \in \mathcal{T} \tag{8}$$

where $\mathcal{N}$ denotes the set of N SUs, $\mathcal{F}$ denotes the set of F frequencies in the CRN cell, and $\mathcal{T}$ denotes the set of T time slots in a scheduling period, i.e., $\mathcal{T} = \{1, 2, \ldots, T\}$. In addition, X_{ift} is a binary decision variable such that $X_{ift} = 1$ if SU i transmits with frequency f in time slot t and 0 otherwise, and a_i is the number of transceivers (antennas) of SU i. In this formulation, (6) guarantees that at least one time slot is assigned to every SU, whereas (7) ensures that at most one SU can transmit in a particular time slot and frequency, hence, obviating collisions between the SUs. Unlike the PUs that are supposed not to understand the signals received from the SUs and hence can treat them as interference, CBS is required to understand, properly decode, and differentiate between the signals sent from different SUs in its service area. Since the CBS does not have any spread spectrum technology, if more than one SU transmits to the CBS using the same frequency and time slot, CBS becomes unable to distinguish between the signals sent by the different SUs; in other words, collision occurs. Constraint (7) prevents collisions by guaranteeing that at most one SU can transmit in a particular time slot and frequency. Moreover, (8) represents the fact that an SU i cannot transmit at the same time using frequencies more than the number of its transceivers (antennas) a_i, because each transceiver can tune to at most one frequency at a time.

The formulation in [22, eq. (5)–(8)] maximizes the total throughput of all SUs in the CRN cell regardless of the minimum throughput requirements of the SUs. However, in a specific time period, each SU may be executing different applications with various requirements and priorities. For instance, a real-time application may require a high minimum data rate (throughput) for proper operation. Unless the resources are allocated to this SU such that its minimum throughput requirement is met, the allocated resources may be useless for this SU. For example, an SU may require at least 1 Mb/s data rate; that is to say, data rates between 10 and 100 kb/s may be indifferent from each other and may both be useless for this SU. If it is theoretically impossible to allocate its required minimum data rate to a particular SU, it makes more sense not to allocate any resources to this SU at all and allocate the resources instead to other SUs whose minimum data rate requirements can be met. On the other hand, an SU with a nonreal-time application may be satisfied with a lower minimum data rate requirement. An SU with an audio or video application may need a particular data rate in a specific time period. In contrast, an SU with an e-mail application may be satisfied with a lower data rate in that same time period. Furthermore, allocation of data rate higher than the minimum requirement of the SU may not make that SU happier. For instance, a data rate more than 100 Mb/s may not have an additional advantage for that SU; i.e., 101 and 200 Mb/s may be equally good. Therefore, instead of allocating these excess resources to this SU, it makes more sense to allocate some of these resources to another SU that actually needs them. Hence, in a real centralized CRN system, the major goal of the CBS operator is to maximize the number of SUs that are satisfied in terms of throughput.

We formulate in (9)–(11) the scheduling problem that maximizes the number of SUs that are satisfied in terms of throughput while at the same time assuring that the PUs in the service area of the CBS are not disrupted, reliable communication between the SUs and the CBS is maintained, and no collisions occur between the SUs, i.e.,

$$\max \left(\sum_{i=1}^{N} g\left(\Omega_i - \Omega_i^{\min}\right) \right) \tag{9}$$

$$s.t.$$

$$\Omega_i = \sum_{f=1}^{F} \sum_{t=1}^{T} \frac{U_{if} X_{ift}}{T} \tag{10}$$

$$(6) - (8) \tag{11}$$

where Ω_i in constraint (10) denotes the throughput of SU i in this scheduling period, and $\Omega_i^{\min}$ denotes the minimum throughput requirement of SU i. Moreover, $g(\cdot)$ in (9) is the step function that indicates the utility of SU i; i.e., the satisfaction of SU i from the throughput Ω_i is equal to 1 if and only if $\Omega_i \geq \Omega_i^{\min}$ and 0 otherwise. These types of utility functions are referred to in the literature as inelastic utility functions since the users have hard QoS requirements; i.e., utility is to equal zero when the QoS (throughput in our case) is lower than a prespecified threshold [23]. In the rest of this paper, we refer to our formulated problem in (9)–(11) as the MNSU problem.

Traditional data applications such as file transfer and electronic mail are elastic, i.e., they tolerate packet delays rather gracefully. These applications were predominantly used in the past when the Internet traffic was mostly best effort [24]. However, real time applications continuously increase, and their utility functions are inelastic. There are two commonly used ways to model inelastic utility functions: sigmoidal utility or discontinuous utility, where the latter is a special case of the former. In this paper, we focus on discontinuous inelastic utility functions, which have a wide range of applications, such as real-time IP applications and constant bit rate ATM flows [24]. Since the discontinuous utility function is a special case of the sigmoidal utility function, our impossibility (NP-hardness and inapproximability) results in Section IV are valid for sigmoidal utilities as well. Moreover, our proposed heuristic algorithms in Section V can easily be modified for the case of sigmoidal utilities, which is left as future work.

IV. Computational Complexity

A. Preliminaries

Approximation Algorithms: Unless $P = NP$, there is no algorithm that runs in time bounded by a polynomial in the input size and finds an optimal solution for all instances of an NP-hard problem. A sensible approach in this case is to relax the requirement of finding an optimal solution and try to find a solution that is "good enough" and runs in polynomial time. To this end, various heuristic techniques and metaheuristics such as simulated annealing, tabu search, and genetic algorithms have been designed [25]–[29]. These techniques usually lead to good results in practice. They are studied empirically; they might work well, but we may not understand why. Moreover, they do not provide a theoretical lower bound about how far the solution can get away from the optimal solution. There may be certain problem instances on which the (meta)heuristic performs very

bad, and we do not know for sure whether those instances occur in a real-life situation. In some applications, reliability on the performance of the algorithm might be very important. Even when those instances that yield poor solutions occur very rarely in practice, its consequences might be costly. Furthermore, the performance of metaheuristics usually depend heavily on numerous parameters, the values of which need to be tuned depending on the data. This may make the performance of the algorithm heavily dependent on the data; if the circumstances in the real-life scenario do not meet the expectations, the performance of the algorithm might be poor. To tackle this possibility, online parameter tuning might be employed; however, this may complicate the implementation [30].

A mathematical analysis accompanies each approximation algorithm, and this analysis quantifies how well the algorithm performs in all instances; i.e., it gives a theoretical lower bound for the worst-case performance of the algorithm on all problem instances compared to the optimal solution. In other words, there is an *a priori* guarantee for every input, and specific inputs may in fact perform much better than the performance guarantee. The field of approximation algorithms also provides us with the means of differentiating between different optimization problems in terms of their approximability. This way, we get a deeper mathematical understanding of the problem's structure. Furthermore, approximation algorithms also provide an analytically rigorous basis to study heuristics. Approximation algorithms for simpler versions of the problem give us some idea of how to design a heuristic that will perform well in practice for the actual problem. Moreover, by looking at the nature of the approximation algorithms, we can also design heuristics that will perform well for special problem instances [30].

Let Π be a maximization problem and $\rho \geq 1$. A (feasible) solution s of an instance I of Π is a ρ approximation if its objective function value $O_{\Pi}(s)$ is at least a factor ρ of the optimal objective function value $O^{*}_{\Pi}(I)$ of I, i.e., $O_{\Pi}(s) \geq (O^{*}_{\Pi}(I)/\rho)$. An algorithm ALG is said to be a ρ approximation algorithm for a maximization problem Π if ALG returns a ρ approximation for every instance I of Π supplied to it. A problem Π is said to be ρ approximable if there is a polynomial-time ρ approximation algorithm for it. Π is said to be ρ inapproximable if there is no polynomial-time ρ approximation algorithm for it unless $P = NP$.

Suppose that we have two optimization problems Π and Π' such that instances of one problem can be mapped onto instances of the other in a way that nearly optimal solutions to instances of the latter problem can be transformed back to yield nearly optimal solutions to the former. This way, if we have an approximation algorithm for problem Π' and an efficient approximation-preserving reduction from problem Π to problem Π', by composition, we obtain an approximation for problem Π. More formally, an approximation ratio preserving (polynomial time) reduction from a maximization problem Π to a maximization problem Π' is a pair of algorithms (f, g) such that (1) f transforms every instance I of Π to an instance $I' = f(I)$ of Π', and (2) g transforms every ρ approximation s' of $I' = f(I)$ to a ρ approximation $g(s')$ of I. We denote this fact by $\Pi \preceq_{APX} \Pi'$. Π and Π' are said to be equivalent under approximation preserving reductions if $\Pi \preceq_{APX} \Pi'$ and $\Pi' \preceq_{APX} \Pi$. If $\Pi \preceq_{APX} \Pi'$, then if there exists a ρ approximation algorithm for Π', we can then get a ρ approximation algorithm for Π. Likewise, if Π cannot have a ρ approximation algorithm unless $P = NP$, then Π' is also ρ inapproximable.

To analyze the complexity and approximability properties of the MNSU problem, we use BIN COVERING and MMFS problems, which we explain in the sequel.

*B*in *C*overing *Problem:* The BIN COVERING problem is basically the covering version of the bin packing problem. Given n items with sizes $c_1, \ldots, c_n \in (0, 1]$, maximize the number of bins opened so that each bin has items summing to at least 1 [31]. The BIN COVERING problem cannot be approximated within any constant factor better than 2 unless $P = NP$. This result is based on the observation that a 2-approximation algorithm for the BIN COVERING problem applied to instances in which the total size of the items is 2 would solve the PARTITION problem [32]. The BIN COVERING problem exists in the literature also under the name of dual bin packing problem [31].

MMFS Problem: In our previous work [33], we have formulated the following ILP problem:

$$\max\ Z' \tag{12}$$

$$s.t.$$

$$Z' \leq \frac{\sum_{f=1}^{F}\sum_{t=1}^{T} U_{if} X_{ift}}{T}; \forall\, i \in \mathcal{N} \tag{13}$$

$$(6)-(8) \tag{14}$$

where the objective function in (12) and constraint (13) together maximize the throughput value of the SU that has the minimum throughput. If $U_{i'f} = U_{i''f} \forall i' \neq i''$ and i', $i'' \in \mathcal{N}$, then we can use U_f instead of U_{if}. We have shown in [33] that even when $T = 1, a_i \geq F, U_{if} \in \{0, U_f\}$, and each frequency f has a nonzero U_f value for at most two SUs, MMFS cannot be approximated within any constant factor better than 2 unless $P = NP$. Moreover, we have also shown in [33] that even very special cases of MMFS are NP-hard in the strong sense.

B. Complexity of the MNSU Problem

Theorem 1: BIN COVERING $\preceq_{APX}$ $MNSU$.

Proof: We can show that the BIN COVERING is a special case of MNSU as follows: Let $U_{if'} = U_{if''}, \forall f' \neq f''$ and f', $f'' \in \mathcal{F}$; i.e., let U_{if} values be the same for all f corresponding to a particular i. In other words, let us use U_i in lieu of U_{if}. Let us set $c_i = (U_i/\Omega_i) \leq 1$. Moreover, let us set $T = 1$ and $a_i \geq F, \forall i \in \mathcal{N}$. This special case corresponds to BIN COVERING, where c_i values correspond to the item sizes, and the SUs correspond to the bins. ■

Corollary 1: It is proved in [31] that BIN COVERING is NP-hard in the strong sense. Hence, due to Theorem 1, MNSU is also NP-hard in the strong sense.

Corollary 2: It is proved in [32] that BIN COVERING cannot be approximated within any constant factor better than 2 unless $P = NP$. Hence, due to Theorem 1, MNSU also cannot be approximated within any constant factor better than 2 unless $P = NP$.

Theorem 2: There is a polynomial time reduction from $MMFS$ to $MNSU$.

Proof: Assume that we have a polynomial-time algorithm A for MNSU. We can use algorithm A to solve MMFS in polynomial time as follows: Let Ω_{MMFS}^{OPT} be the optimum solution of MMFS and Ω_{MMFS}^{UB} be an upper bound for the optimum solution of MMFS; i.e., $\Omega_{MMFS}^{OPT} \leq \Omega_{MMFS}^{UB}$. Let us set Ω_{MMFS}^{UB} to a very large constant. We can set our initial guess for Ω_{MMFS}^{OPT} equal to Ω_{MMFS}^{UB}; i.e., we set $\Omega_i = \Omega_{MMFS}^{UB} \forall i \in \mathcal{N}$ and execute algorithm A. If the result of algorithm A is equal to N, i.e., if all SUs can be satisfied, then it means that the result for MMFS is equal to Ω_{MMFS}^{UB}. That is to say, if all SUs can be satisfied, then it means that it is possible to assign each SU a throughput value of $\Omega_i = \Omega_{MMFS}^{UB}$; therefore, the throughput of the SU that has the minimum throughput is at least Ω_{MMFS}^{UB}. Otherwise, we can iteratively set a new value for our guess for Ω_{MMFS}^{OPT} by doing a binary search between 0 and Ω_{MMFS}^{UB}, and execute algorithm A in each iteration of the binary search to check whether our guess was true or not. Since binary search and the rest of the operations run in polynomial time, we can find in this way Ω_{MMFS}^{OPT} in polynomial time using algorithm A. ■

Corollary 3: Due to Theorem 2, MNSU is at least as hard as MMFS.

The MNSU problem can be converted to a binary integer programming problem (will be explained in detail in Section V), which may in general be NP-hard. However, this does not necessarily imply that each and every integer programming problem has to be NP-hard. Some certain special cases of integer programming problems may be solvable in polynomial time. For instance, integer programming formulations for maximum weighted bipartite matching and minimum spanning tree problems are solvable in polynomial time. Corollary 1 shows that MNSU is indeed NP-hard in the strong sense. Our proof in Theorem 1 about the fact that BIN COVERING is a special case of MNSU not only leads to the 2-inapproximability result via Corollary 2 but gives insight into the combinatorial nature of possible efficient solutions to MNSU as well. For instance, the authors in [31] present approximation algorithms for the BIN COVERING problem. Since BIN COVERING is a special case of MNSU, these approximation algorithms cannot be directly applied to the MNSU problem. However, due to the similarities in the combinatorial nature of both problems, some approaches used in these approximation algorithms can inspire and motivate part of the possible heuristic techniques to solve the MNSU problem. As also mentioned in [30], approximation algorithms for simpler versions of a problem give us some idea of how to design a heuristic that will perform well in practice for the actual problem. For instance, our proposed best first resource assignment (BFRA) problem in Section V-A uses some of the ideas in the approximation algorithms for BIN COVERING in [31].

V. PROPOSED ALGORITHMS

Corollaries 1–3 corroborate that MNSU is a computationally very difficult problem. Therefore, designing heuristic algorithms for MNSU is of paramount importance. To this end, we propose two heuristic algorithms for MNSU.

First, we convert MNSU to a binary ILP (BILP) problem as follows: Since the utility function $g((\Omega_i - \Omega_i^{\min}))$ in (9) produces either 1 or 0, we can introduce a variable s_i that is equal to 1 if SU i is satisfied in terms of throughput and 0 otherwise. Therefore, the solution of MNSU formulated in (9)–(11) is equivalent to the solution of the following BILP:

$$\max \left(\sum_{i=1}^{N} s_i \right) \tag{15}$$

$$s.t.$$

$$\sum_{f=1}^{F} \sum_{t=1}^{T} \frac{U_{if} X_{ift}}{T} \geq \Omega_i^{\min} \times s_i; \forall\, i \in N \tag{16}$$

$$(6) - (8). \tag{17}$$

Constraint (16) models the behavior of the function $g(\Omega_i - \Omega_i^{\min})$. More precisely, if an SU i is satisfied, i.e., if $s_i = 1$, then its throughput (Ω_i) is greater than or equal to its minimum throughput requirement $(\Omega_i^{\min})$.

Optimization software CPLEX [34] can provide efficient solutions to ILP problems. Whenever CPLEX finds an optimal solution, it indicates this fact in its output. If it takes too much time for CPLEX to converge to optimality, a tolerance value can be provided to the software such that the computation terminates once a solution within the given percentage of the optimal solution is found. The parameter *epgap* serves this purpose. Therefore, the BILP in (15)–(17) can be solved efficiently using CPLEX. However, in a real-life scenario, a commercial optimization software may not be available at the CBS. Moreover, the running times of an optimization software may be so high that it may be unrealistic to use it in a CBS scheduler. Therefore, we propose in this paper two heuristic algorithms for MNSU. We compare the performance of our heuristics with the values obtained from CPLEX. Recall that we have proved in Section IV that MNSU is NP-hard in the strong sense. Therefore, optimal solutions cannot be obtained in polynomial time unless $P = NP$. As the problem size gets larger, CPLEX running times become prohibitive, and we have to resort to using the epgap parameter to obtain solutions in a reasonable amount of time. In our simulations, when finding the optimal solution takes too much time, we experimentally set epgap to as small a value as possible such that we obtain a solution in a reasonable amount of time. The highest epgap value we used is 0.2, which occurs in only a few of the experiments. The average of the epgap values we used in all experiments is 0.034. Therefore, each CPLEX solution is either equal to the optimal solution or very close to it, and hence, they constitute a reasonable performance benchmark for our heuristic algorithms.

Note that our MNSU formulation does not differentiate between SUs that have different minimum throughput requirements and simply aims to maximize the number of satisfied SUs. Therefore, SUs with smaller minimum throughput requirement may be favored in some practical situations. If the CBS operator gives more importance to SUs that have higher minimum throughput requirements such as SUs with real-time applications, then it may assign a weight r_i that indicates the

relative importance of satisfying SU i. If the CBS operator gives more importance to satisfying SUs with higher minimum throughput requirements, then it can set the r_i value of these users to a higher value. We can then formulate this new problem as a BILP with objective function $\max(\sum_{i=1}^{N} r_i s_i)$ subject to the same constraints. Our heuristic algorithms can be modified accordingly. In this paper, we focus on the MNSU formulation and leave this modification for future work.

We refer by $FTRP_{ft}$ (frequency and time slot resource pair) to the resource pair that consists of frequency f and time slot t. To maximize the number of satisfied users, two important factors have to be considered while designing a strategy for assigning the resources to the SUs: the number of packets that the SUs need to get satisfied ($\Omega_i^{\min} \times T$) and the number of packets each SU i can transmit by each frequency and time slot resource pair ($FTRP$), i.e., U_{if}.

At the beginning of the execution of the algorithm in each scheduling period, initially, the packet needs (PN_i) of each SU i to get satisfied is equal to $\Omega_i^{\min} \times T$. This PN_i value decreases as the $FTRP$s are assigned to SU i throughout the algorithm. It makes sense to assign the resources primarily to the SUs who have less packet need because by this way more SUs can be satisfied with the same amount of resources. For instance, instead of assigning all resources to one SU who needs 400 packets per scheduling period, we can distribute these resources to five SUs, each of which needs only 80 packets, and make five SUs satisfied instead of one. Moreover, since U_{if} values vary from one SU to another, it makes more sense to assign a resource pair that consists of a frequency and time slot to the SU for which the corresponding U_{if} value is maximum.

Taking these factors into account, we propose two computationally efficient heuristic algorithms for MNSU: 1) BFRA and 2) resource assignment with partial backtracking (RAPB) algorithms. BFRA is a greedy algorithm designed to meet the requirements and constraints of the MNSU problem. It takes into account the unique constraints of the MNSU problem such as the fact that each SU may have a different number of antennas, the scheduling period consists of multiple time slots, and each SU has to be assigned at least one time slot. Possible greedy algorithms for other problems such as the BIN COVERING problem cannot be directly applied to the MNSU problem because MNSU is a much more generalized problem having additional features and constraints. In other words, since the MNSU problem has unique constraints, such as ensuring that each SU is assigned at least one time slot and that each SU possibly has multiple antennas, the existing algorithms in the literature fail to address these types of constraints. Furthermore, our algorithms are easily implementable since they have low complexity while at the same time yielding very good performance. Another important property of our algorithms is that they not only take the number of satisfied users but the total throughput of the SUs into account as well. This property also does not exist in other algorithms in the literature.

In both BFRA and RAPB, each $FTRP_{ft}$ has a weight W_{ift} for each SU i, which is equal to the number of packets that SU i can transmit using frequency f in time slot t. Since the number of packets an SU can send by using a frequency (U_{if}) is the same for all time slots of a scheduling period, W_{ift} is equal to the corresponding U_{if} value. Both BFRA and RAPB algorithms search for an available SU i for which $|PN_i - W_{ift}|$ is minimum for each $FTRP_{ft}$. This value is called "absolute need weight difference" and shown as D_{ift} for SU i and $FTRP_{ft}$. Furthermore, each of these algorithms uses two first-in–first-out queues, namely, resource queue (RQ) and excess RQ (ERQ). RQ contains the unassigned $FTRP$s, which may be assigned to make more SUs satisfied, whereas $FTRP$s in ERQ are used to increase the throughput of the already-satisfied SUs.

A. BFRA Algorithm

The goal of BFRA is to find a computationally efficient solution to the MNSU problem. BFRA finds a feasible solution to MNSU (a solution that satisfies all the constraints of MNSU) and at the same time yields satisfactory results in terms of both the number of satisfied users and the total throughput. The primary objective of BFRA is to find a solution that results in a high number of satisfied users, whereas its secondary objective is to yield high total throughput. We explain BFRA in Steps 1 to 3 and provide a flowchart in Fig. 2.

Step 1) Generate $F \times T$ number of $FTRP$s. For each SU i, find and assign an $FTRP_{xy}$ for whom the W_{ixy} value is maximum among available $FTRP$s. Update PN_i as $PN_i \leftarrow PN_i - W_{ixy}$, and if $PN_i \leq 0$, then mark that SU as satisfied. When each SU is assigned an $FTRP$, put the remaining $(F \times T) - N$ number of $FTRP$s into the RQ.

Step 2) Pop the head $FTRP$ of the queue. Let us call it $FTRP_{ft}$. Search for the unsatisfied SUs that have an available antenna for time slot t. If there is no such SU, put the $FTRP$ to the ERQ. Otherwise, find the SU i for which D_{ift} is minimum, and assign $FTRP_{ft}$ to SU i while updating the PN_i value as $PN_i \leftarrow PN_i - W_{ift}$. If $PN_i \leq 0$, then mark SU i as satisfied.

If either the RQ is empty or all SUs are satisfied, go to Step 3; otherwise, repeat Step 2.

Step 3) For each $FTRP$ in the ERQ, look for a satisfied SU for which the weight of the $FTRP$ is maximum among the SUs that have an available antenna for the time slot associated with this $FTRP$.

In Step 1, first of all, each SU is assigned one $FTRP$ to satisfy constraint (6). In Step 2, resources are assigned to the unsatisfied SUs considering the antenna constraints, packet needs of the SUs, and weights of the $FTRP$s subject to different SUs. Notice that BFRA algorithm tends to select the available SU, whose D_{ift} value is the smallest. In other words, in each assignment decision, the algorithm searches for the SU that would either come closest to its satisfaction limit or become satisfied without gaining much excess resource as a result of this assignment. In other words, our algorithm aims to make the throughput of the satisfied SUs only slightly higher than their required minimum throughput threshold. While designing this behavior of our algorithms, we have been inspired by the methods used by the approximation algorithms for BIN

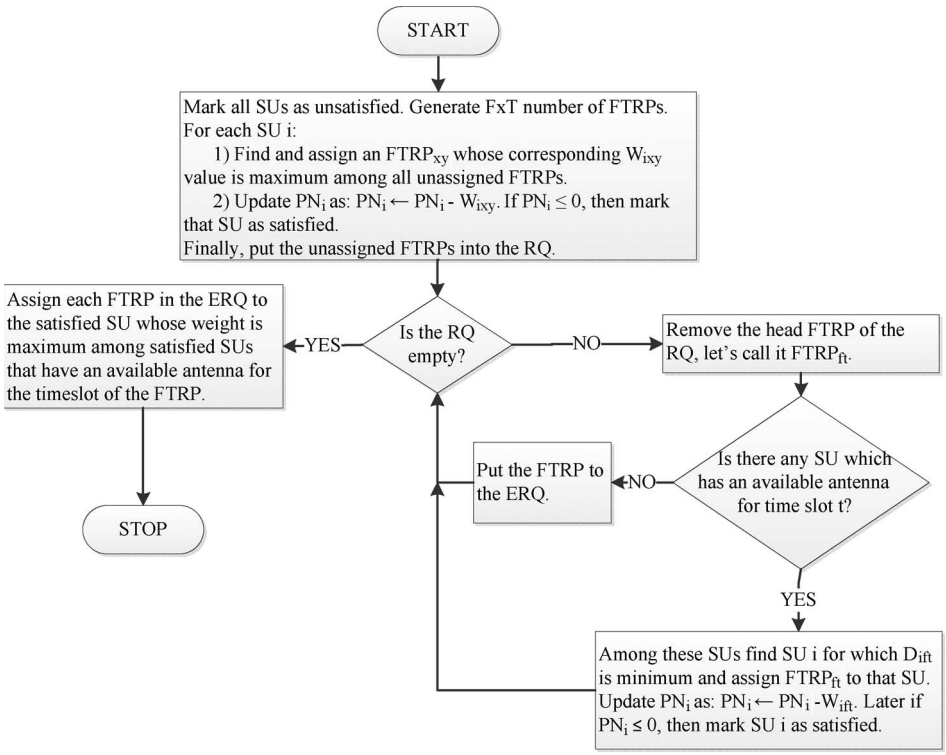

Fig. 2. Action flow diagram of BFRA.

COVERING by the work in [31]. Moreover, although the major goal of BFRA is to maximize the number of satisfied SUs, it also aims to increase the total throughput of the SUs in the CRN by assigning the excess $FTRP$s (i.e. $FTRP$s which could not be assigned to unsatisfied SUs due to the antenna constraints of the SUs) to the already-satisfied SUs in Step 3.

B. RAPB Algorithm

BFRA is a greedy algorithm with very low computational complexity, which we discuss in detail in Section V-C. However, its performance is highly dependent on the order of the $FTRP$s in RQ. That is, although the posterior $FTRP$s in the queue have larger weights, $FTRP$s that are close to the front of the queue are more privileged in terms of being assigned to an SU. This situation arises because BFRA does not allow the replacement of an $FTRP$ with another $FTRP$ once it is assigned to an SU.

RAPB compensates this drawback of BFRA by replacing an already assigned $FTRP$ with a recently popped $FTRP$, whose W_{ift} value is larger for that particular SU. We explain RAPB in Steps 1–6 and provide an action flow diagram in Fig. 3. The major difference between BFRA and RAPB is this backtracking feature, which makes RAPB independent of the order in which $FTRP$s are put into the queue.

We explain below the steps of RAPB. Notice that Step 1 and Step 6 of RAPB are the same as in BFRA.

Step 1) Generate $F \times T$ number of $FTRP$s. For each SU i, find and assign $FTRP_{xy}$ for whom the W_{ixy} value is maximum among available $FTRP$s. Update PN_i as $PN_i \leftarrow PN_i - W_{ixy}$, and if $PN_i \leq 0$, then mark that SU as satisfied. When each SU is assigned an $FTRP$, put the remaining $(F \times T) - N$ number of $FTRP$s into the RQ.

Step 2) If the RQ is empty, go to Step 6. Otherwise, pop the head $FTRP$ of the queue; let us call it $FTRP_{ft}$. Mark all unsatisfied SUs as unchecked, and go to Step 3.

Step 3) Find the SU i for which D_{ift} is minimum among all unsatisfied and unchecked SUs. If that SU has an available antenna for the time slot to which the $FTRP$ belongs, then assign the $FTRP$ to it; otherwise, go to Step 4. If the assignment is valid, update PN_i as $PN_i \leftarrow PN_i - W_{ift}$, and before going to Step 2, check whether $PN_i \leq 0$. If yes, mark this SU as satisfied.

Step 4) Compare the weight of the $FTRP$, i.e., W_{ift}, with the weights of the already assigned $FTRP$s that occupy an antenna for time slot t.

If W_{ift} is bigger than their smallest one, then take the $FTRP$ with the smallest weight from SU i, put it at the end of RQ, and instead assign $FTRP_{ft}$ to SU i. If we denote the weight of the taken $FTRP$ as W_{ikt}, then update PN_i as $PN_i \leftarrow PN_i + W_{ikt} - W_{ift}$, and mark this SU as satisfied if $PN_i \leq 0$. Go to Step 2.

On the contrary, if W_{ift} is the smallest one, mark that SU as checked and continue with Step 5.

Step 5) If there are any unchecked users among unsatisfied SUs, go to Step 3. Otherwise, put the $FTRP$ to ERQ, and go to Step 2.

Step 6) If there exist some $FTRP$s in ERQ, then for each $FTRP$ in the queue, look for a satisfied SU for which the weight of the $FTRP$ is maximum among

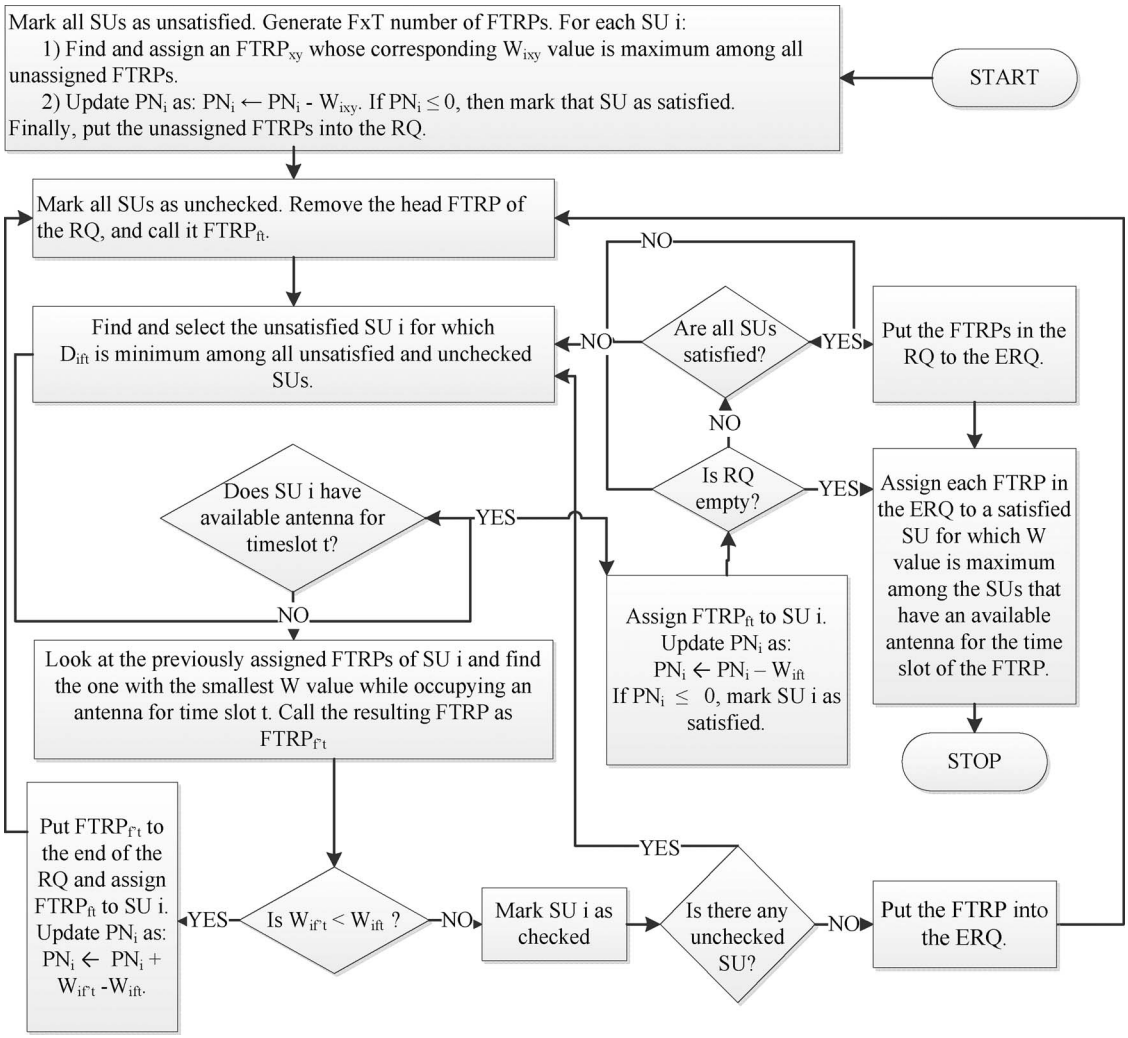

Fig. 3. Action flow diagram of RAPB.

SUs that have an available antenna for the time slot of the $FTRP$.

In cases where the computation time is not critical and resources are scarce, Step 3 may be extended to search for the unsatisfied and unchecked SU i with the minimum D_{ift} value while $D_{ift} < k$ for any positive number k. If no such SU is available, then the $FTRP$ is put at the end of the RQ. This modification limits the number of packets that can excessively be assigned to an SU in Step 3.

C. Computational Complexity Analysis

Worst-Case Performance Analysis of BFRA: In the worst-case scenario, assigning an $FTRP$ to each SU in Step 1 has complexity of $O(NF)$ because for each SU we find the frequency with which the SU can send the maximum number of packets. Then, for each $FTRP_{ft}$ in the RQ, we search for the SU i for which D_{ift} is minimum among the SUs that have an available antenna for time slot t. This search is $O(N)$ for each $FTRP$ since it is sufficient to scan the SU list once to learn both of these attributes. Since there are $FT-N$ number of $FTRP$s in the queue, Step 2 is $O((FT-N)\times N)$. Finally, if we assume that each SU has A number of antennas, the complexity of Step 3 is $O((FT-NA)\times N)$, which is equal to $O(FTN-AN^2)$. Here, $(FT-NA)$ is the number of $FTRP$s that are in ERQ. Eventually, the worst-case complexity of the BFRA algorithm is $O(NF)+O((FT-N)\times N)+O((FT-NA)\times N)$. We can assume that in general $N>A$ since the number of antennas of SUs cannot be high in practice due to expensive hardware. Therefore, the worst-case complexity roughly becomes $O(FTN)$.

Worst-Case Performance Analysis of RAPB: Since Step 1 of RAPB is the same as Step 1 of BFRA, in the worst-case scenario, this step has $O(NF)$ computational complexity. In Steps 2–5, $FT-N$ number of $FTRP$s are assigned to the unsatisfied SUs. In the worst case, each $FTRP$ is put back into RQ for $N-1$ times and thus assigned to an SU for N times. Before the assignment of each $FTRP_{ft}$ in the RQ, first, the SU i with minimum D_{ift} (absolute need weight difference) value is found, and this operation has $O(N)$ complexity. Furthermore, if we assume that all SUs have A number of antennas, then A comparisons can be made in each assignment to find an available antenna. Consequently, the computational complexity of the assignments of $FT-N$ number of $FTRP$s throughout Steps 2–5 is $O((FT-N)\times(N+A)\times N)$. Finally, Step 6 is equivalent to Step 3 of BFRA and has complexity of $O((FT-NA)\times N)$. Furthermore, note that MNSU has a feasible solution if and only if $FT\geq N$ due to constraint (6).

Hence, again assuming that generally $N > A$, the complexity of RAPB is $O(FTN^2)$.

VI. Numerical Evaluation

For the simulations, we use the same CRN environment explained in [22], in which a CRN cell has 600 m radius, and each scheduling period consists of ten time slots, where each time slot is 100 ms. U_{if} values for 5000 scheduling periods in each set of simulations are obtained by taking PU velocities, SU velocities, and the number of PUs into account. The dynamic nature of the spectral environment is due to the physical mobility of the SUs and PUs and the changing spectrum occupancy behavior of the PUs. Accordingly, U_{if} values in each scheduling period are possibly different owing to these changing network conditions.

We consider a random way point mobility model for SUs and PUs as in [22]. We denote by V_p, V_s, and M the velocities of the PUs, SUs, and number of PUs in the CRN cell, respectively. In all simulations, we take $V_p = V_s = 13$ m/s, $M = 20$, and the staying duration between the movement periods equal to 10 s. In the simulations, all SUs managed by the same CBS have the same number of antennas, i.e., $a_i = a\ \forall i \in \mathcal{N}$. Moreover, in some simulation sets, the packet need of each SU i is equal to each other, i.e., $PN_i = PN$. In some other simulation sets, the packet need of each SU differs from each other by a value ΔPN. When we use PN, we implicitly refer to the case where each SU has the same PN value. Hence, in the simulation graphs, a and PN are used to represent the number of antennas and the packet need of each SU in the current simulation set. Each simulation set consists of 5000 scheduling periods, and for different simulation sets, we vary N between 5 and 30, F between 3 and 30, and, finally, PN between 20 and 300.

The spectrum usage behavior of the PUs is also as in [22], where a finite state model is used. Fig. 4 illustrates the finite state model used to simulate the PU spectrum occupancy behavior. Each PU is either in the ON state or OFF state. The ON state encompasses one of the F substates, each corresponding to being active using a frequency among a total of F frequencies. The probability of staying in the ON or OFF states is p_S. At the end of each scheduling period, each PU either stays in the same state with probability p_S or changes its state with probability $1 - p_S$. While switching from the OFF state to the ON state, the probability of selecting each frequency is equally likely; therefore, the probability of transition from OFF state to any frequency is $(1 - p_S)/F$. In a slowly varying spectral environment, the p_S value is usually high; hence, we selected the p_S value as 0.9 in our simulations.

In Fig. 5, we set $F = 15$, $a = 3$, and $N = 5$, 15, and 30 and vary PN between 20 and 300. We compare $BFRA$ and $RAPB$ in terms of the number of satisfied users (N_s) and average total throughput. We observe that both algorithms have similar performance in terms of N_s and total throughput. In particular, N_s decreases as the number of packets each SU needs to be satisfied (PN) increases. The reason for this behavior is that the amount of resources (F) remains constant in this simulation set, whereas the amount of resources that each SU needs increases; therefore, less number of SUs can be satisfied.

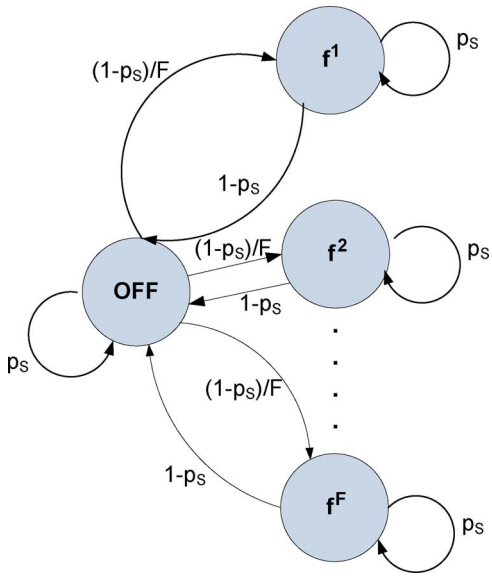

Fig. 4. PU spectrum occupancy model [22].

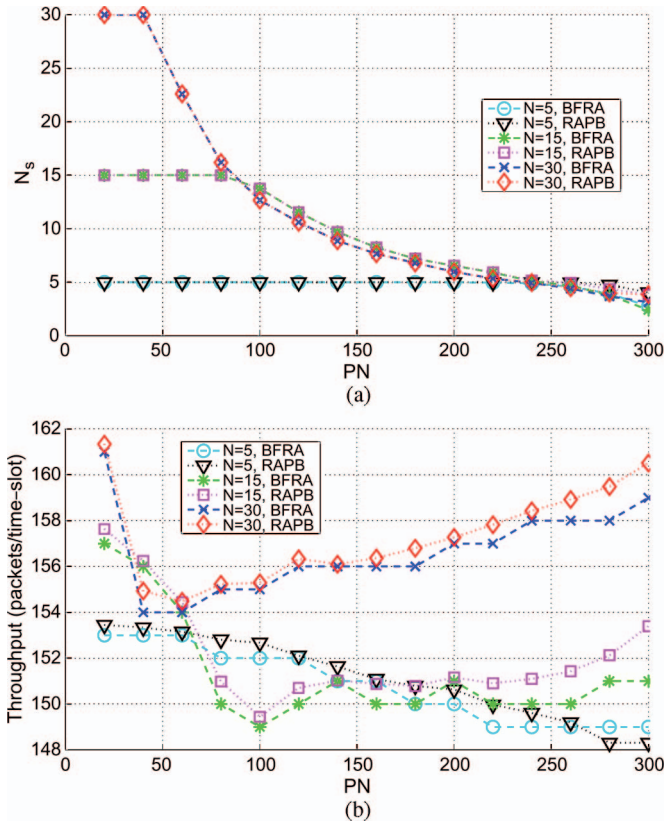

Fig. 5. Comparison of $BFRA$ and $RAPB$ results for varying N and PN. (a) Comparison of N_s results of $BFRA$ and $RAPB$ for varying N and PN. (b) Comparison of throughput results of $BFRA$ and $RAPB$ for varying N and PN.

Fig. 5(b) shows that as PN increases, total throughput does not have a predictable behavior. There are two main reasons for this behavior: The first one is the fact that both $BFRA$ and $RAPB$ are primarily designed to maximize the number of satisfied users, and increasing the total throughput is their secondary objective. They firstly assign the resources to satisfy as many SUs as possible, and then they assign the remaining resources in a way that increases the total throughput. Therefore, the

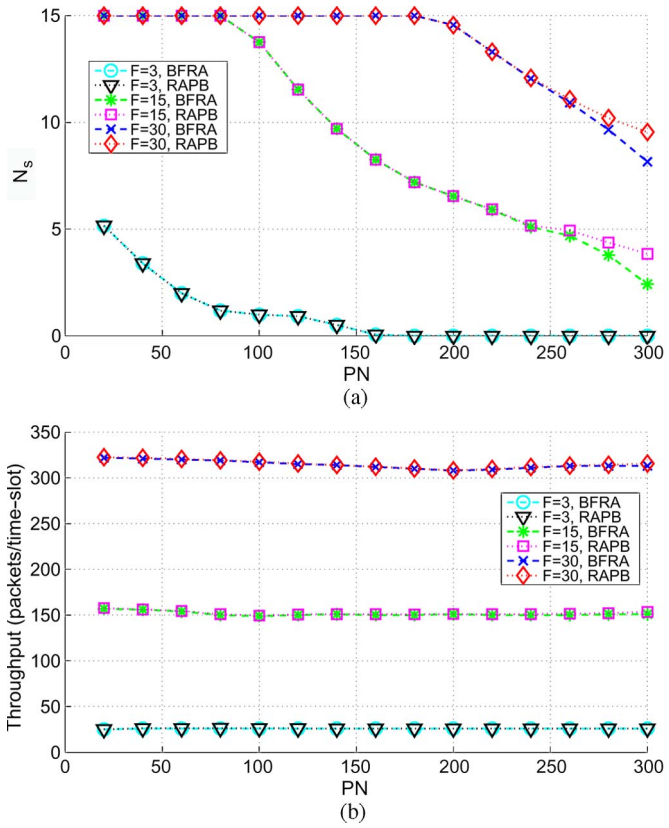

Fig. 6. Comparison of $BFRA$ and $RAPB$ results for varying F and PN. (a) Comparison of N_s results of $BFRA$ and $RAPB$ for varying F and PN. (b) Comparison of throughput results of $BFRA$ and $RAPB$ for varying F and PN.

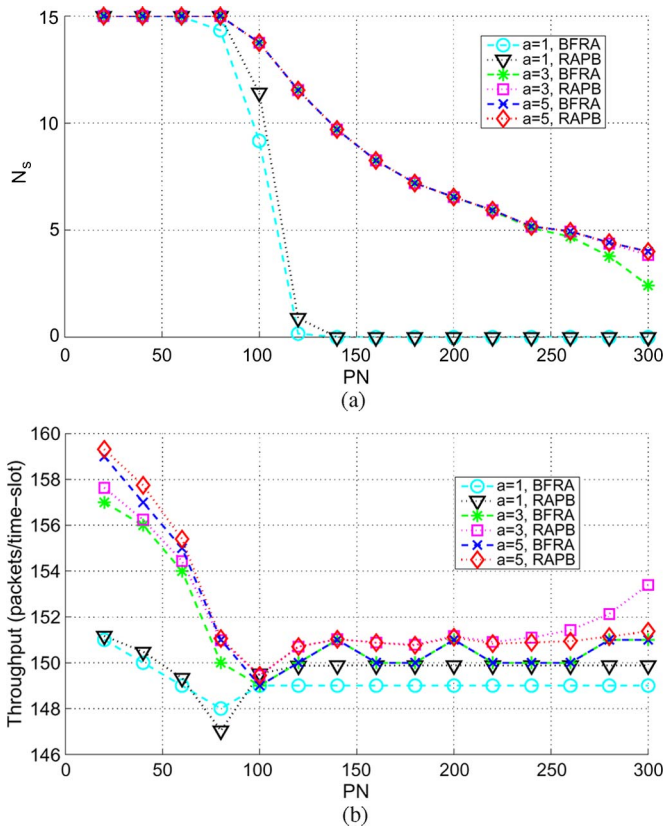

Fig. 7. Comparison of $BFRA$ and $RAPB$ results for varying a and PN. (a) Comparison of N_s results of $BFRA$ and $RAPB$ for varying a and PN. (b) Comparison of throughput results of $BFRA$ and $RAPB$ for varying a and PN.

values of the remaining U_{if} values play an important role in the resulting throughput. The second reason is the heterogeneity in the U_{if} values, i.e., U_{if} values may change depending on the spectral environment, and they depend on both SU i and frequency f. Hence, the total throughput values of $BFRA$ and $RAPB$ depend on the U_{if} values of the remaining resources. The important observation is that the total throughput of $RAPB$ is in most cases higher than the one of $BFRA$ due to its backtracking capability.

In Fig. 6, we set $N = 15$, $a = 3$, and $F = 3$, 15, and 30 and compare the performance of $RAPB$ and $BFRA$. We observe that both algorithms have very similar performance in terms of total throughput and N_s. Furthermore, higher number of frequencies yield higher throughput and higher number of satisfied SUs since increasing number of resources (F) implies more opportunities for the SUs in terms of throughput.

Fig. 7 illustrates the simulation results for $N = 15$, $F = 15$, and $a = 1$, 3, and 5. We observe that in these simulations, the performance of $RAPB$ is similar to or in some cases better than $BFRA$ in terms of both N_s and total throughput. Furthermore, when $PN > 80$, N_s values decrease dramatically for $a = 1$ because the maximum number of packets the SUs can transmit using only one antenna is not sufficient for their minimum needs.

We have also evaluated the performance of our algorithms for the cases where SUs have different PN values. In these experiments, PN values of the N number of SUs are the elements of one of these sets: $Set\ 1 = 40, 45, 50, \ldots, 40 + 5(\mathrm{N}-1)$, $Set\ 2 = 40, 50, 60, \ldots, 40 + 10(\mathrm{N}-1)$, $Set\ 3 = 40, 55, 70, \ldots, 40 + 15(N-1)$. ΔPN refers to the difference in the PN values of the subsequent SUs. For instance, $\Delta PN = 5$, 10, and 15 for Set 1, Set 2, and Set 3, respectively.

In Fig. 8, we have compared both algorithms when each SU has a different packet need value, i.e., $PN_i \neq PN_{i'}\ \forall i \neq i'$. That is to say, the set of PN values of the SUs is equal to Set 1, Set 2, or Set 3. In Fig. 8(a), N varies between 5 and 30, whereas $F = 15$, and $a = 3$. In Fig. 8(b), $N = 15$, $a = 3$, and F varies between 3 and 30. Finally, in Fig. 8(c), F and N are equal to 15, whereas a varies between 1 and 5. Moreover, we observe that in Fig. 8(a)–(c), the N_s results of $RAPB$ are either equal to or slightly better than the outputs of the $BFRA$ algorithm. As the ΔPN value gets larger, the total packet needs of the SUs increase. Therefore, the number of satisfied users decreases as ΔPN increases, although each SU has a different PN from one another.

After the comparison of $BFRA$ and $RAPB$ simulation results, we have observed that the $RAPB$ results are slightly better than the results of $BFRA$. Therefore, we have compared the results of $RAPB$ with the values obtained from CPLEX. In Fig. 9, we compare the outputs of the $RAPB$ algorithm with the CPLEX solutions for $F = 15$, $a = 3$, $N = 5$, 10, 15, and PN varying between 20 and 300. We can see in Fig. 9 that $RAPB$ yields very close performance to CPLEX solutions in terms of N_s.

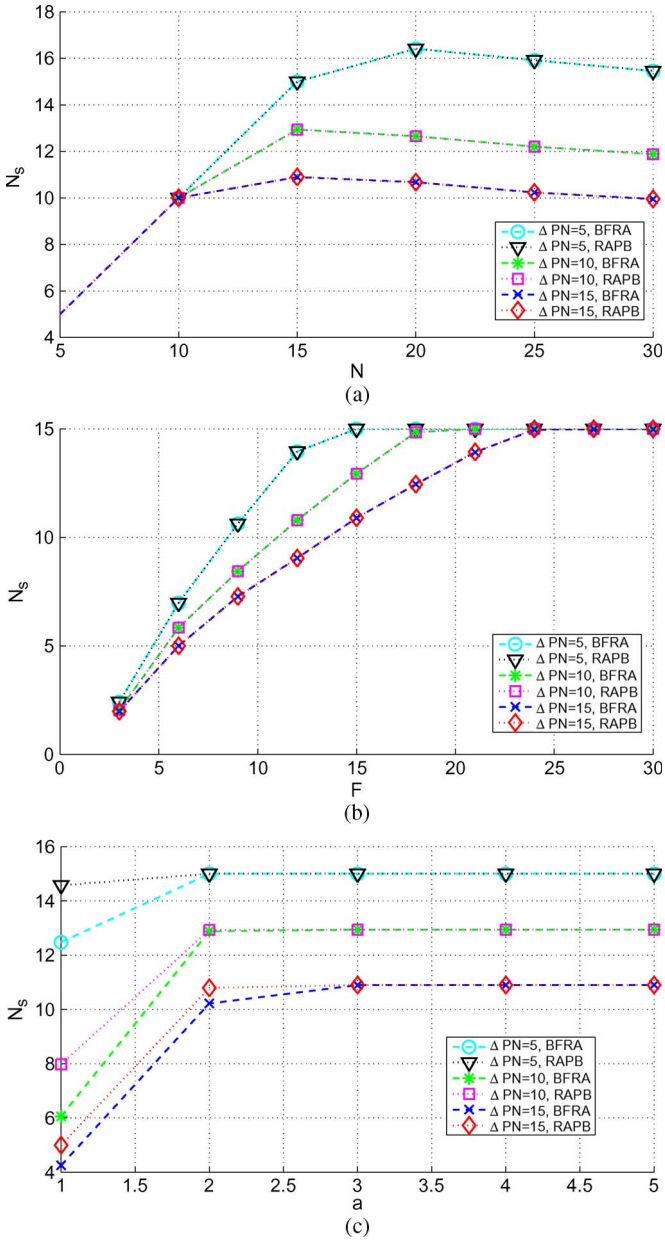

Fig. 8. Comparison of $BFRA$ and $RAPB$ when $\Delta PN = 5$, 10, and 15. (a) Comparison of $BFRA$ and $RAPB$ for varying N when $\Delta PN = 5$, 10, and 15. (b) Comparison of $BFRA$ and $RAPB$ for varying F when $\Delta PN = 5$, 10, and 15. (c) Comparison of $BFRA$ and $RAPB$ for varying a when $\Delta PN = 5$, 10, and 15.

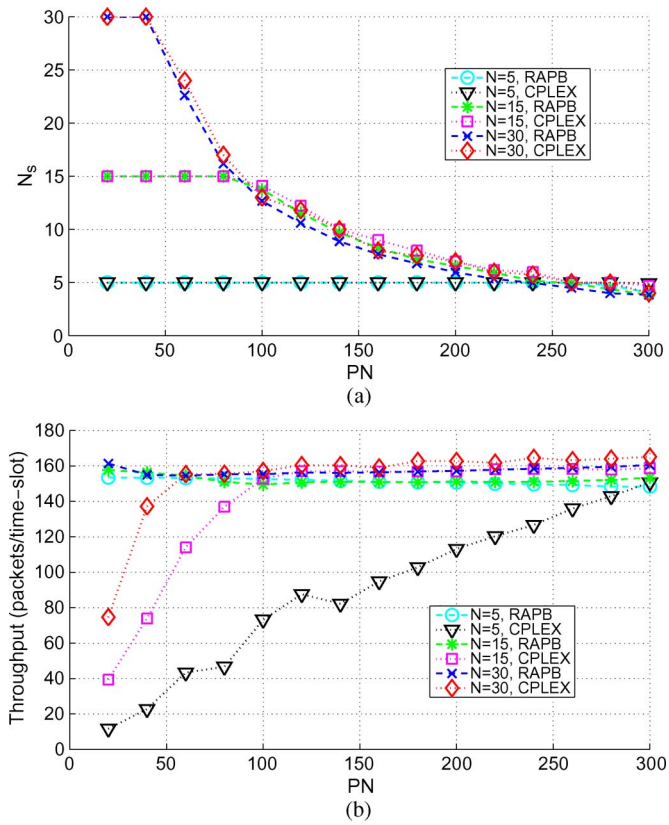

Fig. 9. Comparison of $RAPB$ results with $CPLEX$ solutions for varying N and PN. (a) Comparison of N_s results of $RAPB$ and $CPLEX$ for varying N and PN. (b) Comparison of throughput results of $RAPB$ and $CPLEX$ for varying N and PN.

Furthermore, we see that the throughput performance of $RAPB$ is similar to or better than the performance of CPLEX solutions. In both of our algorithms, we also aim at increasing the total throughput of the SUs. Thus, after calculating the number of satisfied users, we continue to assign the unassigned resources to the SUs to increase their throughput. CPLEX, on the other hand, only aims to maximize the number of satisfied SUs without paying any attention to total throughput. Therefore, CPLEX stops its execution once the maximum possible value for the number of satisfied users is reached and does not assign the remaining frequency and time slot pairs to the other SUs. Hence, throughput values of our algorithms are frequently better than the throughput values obtained from CPLEX solutions.

In Fig. 10, we compare the $RAPB$ algorithm and the CPLEX solutions for $N = 15$, $a = 3$, and $F = 3$, 15, and 30, where PN varies between 20 and 300. The PN values of the SUs are equal in a scheduling period. In this figure, we observe that the N_s results of the $RAPB$ algorithm are very close to the CPLEX solutions. Moreover, $RAPB$ throughput values are again better than the throughput values of the CPLEX solutions for the aforementioned reason.

We have illustrated in Fig. 11 that the performance of $RAPB$ for $N = 15$, $F = 15$, $a = 1$, 3, and 5, and PN varying between 20 and 300. In these simulations, $RAPB$ yields very close performance to the CPLEX solutions in terms of N_s. The increasing packet need (PN) decreases the N_s values. Moreover, the resulting total throughput values are better than the CPLEX throughput values, particularly for $a = 1$. Furthermore, we also observe that N_s drastically drops when $PN = 120$ for the case with $a = 1$. The reason for this behavior is as follows: When we analyze the U_{if} values for this simulation set, we see that the highest U_{if} value is 12, which occurs rarely in practice. Most of the U_{if} values are either 10 or 11. When $PN = 120$ and $a = 1$, all SUs need to be assigned frequencies with $U_{if} = 12$ for all of the $T = 10$ time slots for all of them to be satisfied. Since this assignment scenario can rarely be accomplished due to the fact that $U_{if} = 12$ rarely occurs, N_s is low. On the other hand, when

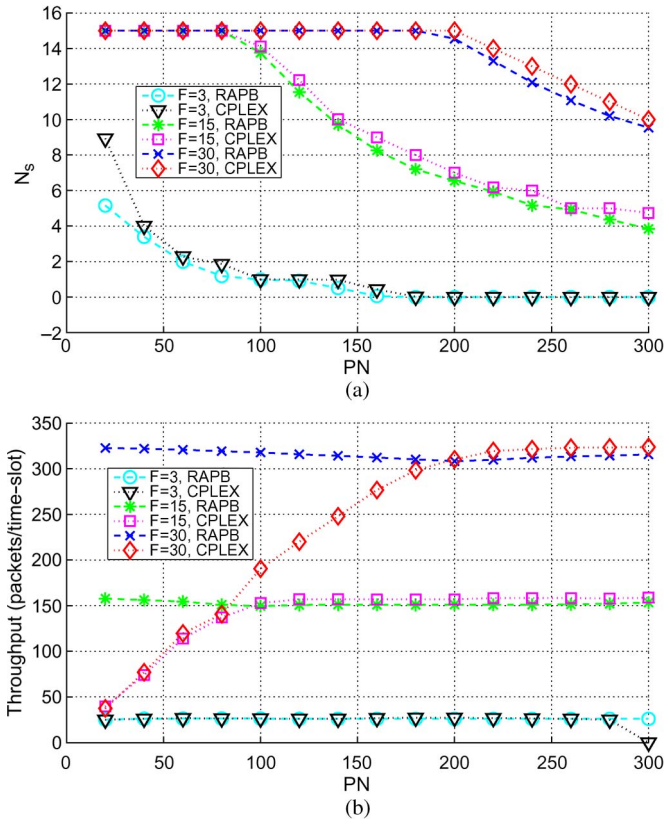

Fig. 10. Comparison of $RAPB$ results with $CPLEX$ solutions for varying F and PN. (a) Comparison of N_s results of $RAPB$ and $CPLEX$ for varying F and PN. (b) Comparison of throughput results of $RAPB$ and $CPLEX$ for varying F and PN.

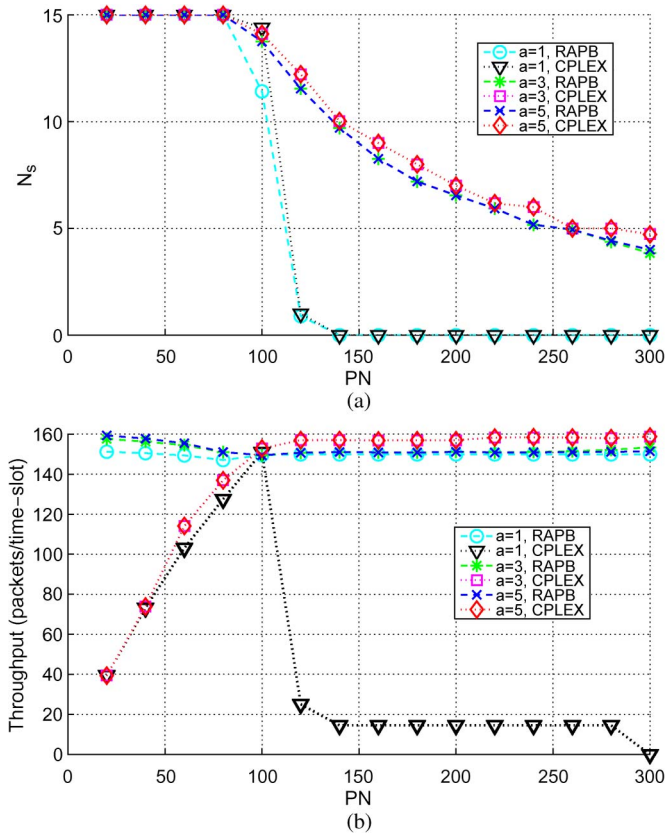

Fig. 11. Comparison of $RAPB$ results with $CPLEX$ solutions for varying a and PN. (a) Comparison of N_s results of $RAPB$ and $CPLEX$ for varying a and PN. (b) Comparison of throughput results of $RAPB$ and $CPLEX$ for varying a and PN.

$PN = 100$ and $a = 1$, all SUs need to be assigned frequencies with at least $U_{if} = 10$ for all of the $T = 10$ time slots for all of them to be satisfied. Since $U_{if} = 10$ and $U_{if} = 11$ frequently occur in this simulation scenario, N_s is high. Nevertheless, when $a = 3$ or $a = 5$, SUs have the opportunity to be satisfied by having other frequencies with lower U_{if} values assigned to their remaining antennas, and hence, they do not have to rely on merely the frequencies with $U_{if} = 12$ to get satisfied. Therefore, N_s values can still be maintained to be high when $PN = 120$ and $a = 3$ or $a = 5$. The sudden drop of the CPLEX total throughput at $PN = 120$ for $a = 1$ happens for the same reason as the drop in N_s values since CPLEX does not assign the remaining frequencies to the unsatisfied users, and only the throughput of the satisfied users contribute to the total throughput in the CPLEX solutions.

We have illustrated the outputs of the $RAPB$ algorithm and the CPLEX results of the simulation sets in which the PN values of the SUs are the elements of $Set1$, $Set2$, or $Set3$ in Figs. 12–14. In these figures, for ΔPN, $RAPB$, and ΔPN, $CPLEX$ denotes the difference between the PN values of each consecutive SU in the $RAPB$ and CPLEX simulation sets, respectively.

In Fig. 12, we have illustrated the simulations where N varies between 5 and 30, $F = 15$, and $a = 3$. When the ΔPN value is greater, the number of SUs that have greater packet need to get satisfied increases. Therefore, it becomes more difficult to satisfy the SUs. Consequently, in this figure, we observe that N_s decreases as ΔPN increases in both $RAPB$ outputs and CPLEX solutions. We also show that the $RAPB$ throughput values are again either equal to or better than the CPLEX solutions. Furthermore, as N increases, N_s also initially increases as long as the resources (frequencies) are sufficient to satisfy the SUs. However, N_s does not increase after some point since the resources in the system are finite, and irrespective of how large N is, only a certain maximum number of SUs can be satisfied.

In Fig. 13, we have illustrated the simulations where F varies between 3 and 30, $N = 15$, and $a = 3$. We observe that for $F = 24$, all simulation sets saturate. Since the experiments have been done for $N = 15$, it is natural that saturation occurs when $N_s = 15$ (all SUs are satisfied). Furthermore, we see that the PN set that consists of the smallest values ($\Delta PN = 5$) saturates first. The reason for this behavior is because more SUs can be satisfied when the minimum throughput requirements of the SUs are smaller. Furthermore, N_s increases as F increases because it is easier to satisfy the throughput requirements when there are more resources (frequencies) in the system. We also observe that before saturation, the N_s results of the $RAPB$ algorithm are at times lower than CPLEX solutions, but the difference is very little: either one or two SUs. The reason for this behavior is the suboptimal nature of our proposed algorithm, which manifests itself more when it is more difficult

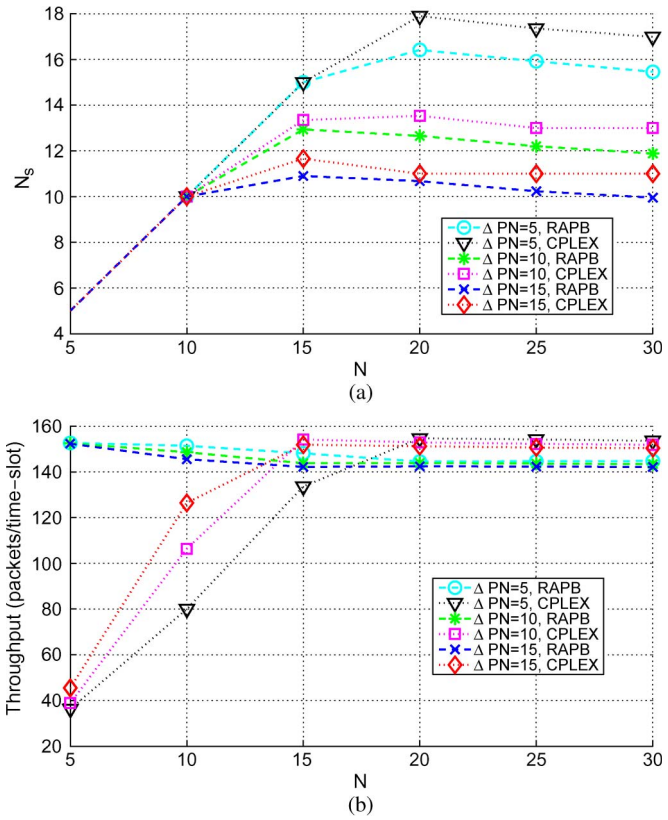

Fig. 12. Comparison of $RAPB$ results with $CPLEX$ solutions when $\Delta PN = 5$, 10, and 15 for varying N. (a) Comparison of N_s results of $RAPB$ and $CPLEX$ for varying N when $\Delta PN = 5$, 10, and 15. (b) Comparison of throughput results of $RAPB$ and $CPLEX$ for varying N when $\Delta PN = 5$, 10, and 15.

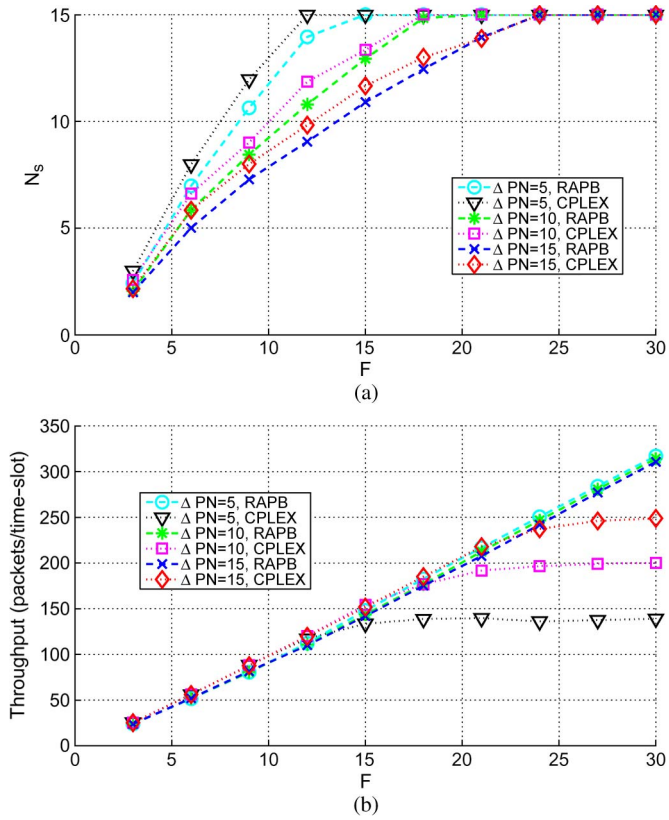

Fig. 13. Comparison of $RAPB$ results with $CPLEX$ solutions when $\Delta PN = 5$, 10, and 15 for varying F. (a) Comparison of N_s results of $RAPB$ and $CPLEX$ for varying F when $\Delta PN = 5$, 10, and 15. (b) Comparison of throughput results of $RAPB$ and $CPLEX$ for varying F when $\Delta PN = 5$, 10, and 15.

to satisfy the throughput needs of all SUs, i.e., when the number of frequencies is small. However, we observe in Fig. 13(b) that our algorithm yields more throughput than CPLEX since unlike CPLEX, it takes total throughput into account. Therefore, we can conclude that our algorithm yields more total throughput than CPLEX while at the same time yielding N_s (number of satisfied SUs) values that are very close to CPLEX.

Finally, in Fig. 14, we have illustrated simulations where $N = 15$, $F = 15$, and a varies between 1 and 5. In this figure, we observe that N_s values saturate when $a = 2$ for all simulation sets with different ΔPN values. This is because when $a = 2$, most of the $FTRP$s belonging to 15 frequencies are already assigned to some SUs, and few resources are left to be assigned to the SUs, even if they have available antennas. The N_s results of our proposed algorithm are very close to the CPLEX solutions. However, Fig. 14(b) shows that our algorithm RAPB yields considerably higher throughput than CPLEX. This behavior is because CPLEX only takes N_s into account, whereas RAPB gives importance to both N_s and total throughput. In other words, CPLEX does not assign the remaining frequencies to the unsatisfied users, and only the throughput of the satisfied users contributes to the total throughput in the CPLEX solutions. This behavior illustrates the major strength of our algorithm, i.e., it yields higher total throughput than CPLEX while still achieving N_s values that are close to the CPLEX solutions.

In our simulations, we have observed the effects of varying N, F, a, and PN parameters on our proposed algorithms' performances. Using these various input sets, we have compared the N_s and throughput results of the $BFRA$ and $RAPB$ algorithms both with each other and with CPLEX solutions. We conclude that $BFRA$ and $RAPB$ algorithms yield very close performance in terms of N_s, and the obtained results are very close to the corresponding CPLEX solutions. Furthermore, the total throughput performance of our proposed algorithms is better than the one of the CPLEX solutions. Moreover, our algorithms have low computational complexity, and therefore, they are suitable for practical use in CRNs.

VII. Conclusion

In this paper, we have introduced and formulated a scheduling problem for centralized CRNs, in which the objective is to maximize the number of SUs that are satisfied in terms of throughput. We have proved that the problem is NP-hard in the strong sense and cannot be approximated within any constant factor better than 2 unless $P = NP$. Furthermore, we have presented two computationally efficient heuristic algorithms for the problem. Through our numerous simulations with varying input parameters, we have observed that

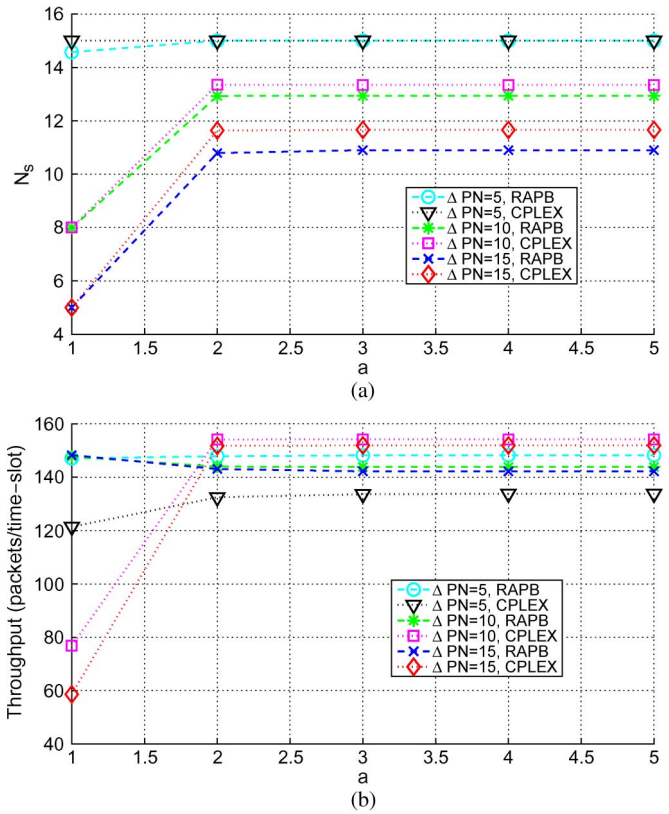

Fig. 14. Comparison of $RAPB$ results with $CPLEX$ solutions when $\Delta PN = 5$, 10, and 15 for varying a. (a) Comparison of N_s results of $RAPB$ and $CPLEX$ for varying a when $\Delta PN = 5$, 10, and 15. (b) Comparison of throughput results of $RAPB$ and $CPLEX$ for varying a when $\Delta PN = 5$, 10, and 15.

our proposed algorithms yield very close performance to the CPLEX solutions in terms of the number of satisfied SUs in the network. Moreover, our algorithms achieve better results than the CPLEX solutions in terms of network throughput because unlike CPLEX, we also take the total throughput into account.

In practice, the minimum throughput requirement of a particular SU may have a dynamic temporal behavior, depending on the application it is executing. As future work, we plan to devise a more realistic modeling of the changing minimum throughput requirements of the SUs to better assess the performance of our proposed algorithms.

Extending our proposed scheduling schemes to a distributed environment might also be an interesting future work. To this end, auction or game-theory-based techniques might be utilized. Each SU might bid for a subset of the time slots, and available frequencies such that the possible throughput that it gets by using its bidded resources is slightly higher than its minimum throughput requirement. In a distributed network, each SU has little knowledge about the channel conditions of the other SUs. Therefore, a distributed learning mechanism based on the outcome of past scheduling decisions might accompany the auction-based distributed algorithm.

REFERENCES

[1] I. Akyildiz, W. Lee, M. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Comput. Netw.*, vol. 50, no. 13, pp. 2127–2159, Sep. 2006.

[2] R. Urgaonkar and M. Neely, "Opportunistic scheduling with reliability guarantees in cognitive radio networks," *IEEE Trans. Mobile Comput.*, vol. 8, no. 6, pp. 766–777, Jun. 2009.

[3] D. Xue and E. Ekici, "Guaranteed opportunistic scheduling in multi-hop cognitive radio networks," in *Proc. IEEE INFOCOM*, 2011, pp. 2984–2992.

[4] C. Tian and D. Yuan, "Cross-layer opportunistic scheduling for multiclass users in cognitive radio networks," in *Proc. IEEE Int. Conf. WiCOM*, 2008, pp. 1–4.

[5] R. Wang, V. Lau, and Y. Cui, "Decentralized fair Scheduling in two-hop relay-assisted cognitive OFDMA systems," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 1, pp. 171–181, Feb. 2011.

[6] R. Santos, F. Lima, W. Freitas, and F. Cavalcanti, "QoS based radio resource allocation and scheduling with different user data rate requirements for OFDMA systems," in *Proc. IEEE PIMRC*, 2007, pp. 1–5.

[7] E. Rodrigues and F. Casadevall, "Adaptive radio resource allocation framework for multi-user OFDM," in *Proc. IEEE VTC*, 2009, pp. 1–6.

[8] Z. Zhang, Y. He, and E. Chong, "Opportunistic scheduling for OFDM systems with fairness constraints," *EURASIP J. Wireless Commun. Netw.*, vol. 2008, pp. 1–12, Jan. 2008.

[9] S. Pal, S. Kundu, M. Chatterjee, and S. Das, "Combinatorial reverse auction based scheduling in multi-rate wireless systems," *IEEE Trans. Comput.*, vol. 56, no. 10, pp. 1329–1341, Oct. 2007.

[10] H. Mahmoud, T. Yucek, and H. Arslan, "OFDM for cognitive radio: Merits and challenges," *IEEE Wireless Commun.*, vol. 16, no. 2, pp. 6–15, Apr. 2009.

[11] G. Aristomenopoulos, T. Kastrinogiannis, V. Kaldanis, G. Karantonis, and S. Papavassiliou, "A novel framework for dynamic utility-based QoE provisioning in wireless networks," in *Proc. IEEE GLOBECOM*, 2010, pp. 1–6.

[12] T. Kastrinogiannis and S. Papavassiliou, "Utility based short-term throughput driven scheduling approach for efficient resource allocation in CDMA wireless networks," *Wirel. Pers. Commun.*, vol. 52, no. 3, pp. 517–535, 2010.

[13] C. Uc-Rios and D. Lara-Rodriguez, "A low complexity scheduling for maximizing satisfied users in wireless networks," in *Proc. IEEE ICSPCS*, 2010, pp. 1–5.

[14] C. Raman, J. Singh, R. Yates, and N. Mandayam, "Scheduling variable rate links—Centralized and decentralized approaches," in *Cognitive Wireless Networks: Concepts, Methodologies and Visions Inspiring the Age of Enlightenment of Wireless Communications*. Berlin, Germany: Springer-Verlag, 2007, p. 285.

[15] K. Khalil, M. Karaca, O. Erçetin, and E. Ekici, "Optimal scheduling in cooperate-to-join cognitive radio networks," in *Proc. IEEE INFOCOM*, 2011, pp. 3002–3010.

[16] Q. Cheng and B. Kollimarla, "Joint channel and power allocation based on user satisfaction for cognitive radio," in *Proc. IEEE Conf. Inf. Sci. Syst.*, 2009, pp. 579–584.

[17] J. Cho, S. Jang, W. Jung, and J. Kim, "Power allocation and user satisfaction-based switching method in cognitive radio MIMO system," in *Proc. IEEE ICTC*, 2010, pp. 397–402.

[18] M. Nguyen and H. Lee, "Effective scheduling in infrastructure-based cognitive radio network," *IEEE Trans. Mobile Comput.*, vol. 10, no. 6, pp. 853–867, Jun. 2011.

[19] J. Tang, S. Misra, and G. Xue, "Joint spectrum allocation and scheduling for fair spectrum sharing in cognitive radio wireless networks," *Comput. Netw.*, vol. 52, no. 11, pp. 2148–2158, 2008.

[20] R. Yates, C. Raman, and N. Mandayam, "Fair and efficient scheduling of variable rate links via a spectrum server," in *Proc. IEEE ICC*, 2006, vol. 11, pp. 5246–5251.

[21] C. Peng, H. Zheng, and B. Zhao, "Utilization and fairness in spectrum assignment for opportunistic spectrum access," *Mobile Netw. Appl.*, vol. 11, no. 4, pp. 555–576, Aug. 2006.

[22] D. Gözüpek and F. Alagöz, "An interference aware throughput maximizing scheduler for centralized cognitive radio networks," in *Proc. IEEE Int. Symp. PIMRC*, 2010, pp. 1–6.

[23] J. Hou, J. Yang, and S. Papavassiliou, "Integration of pricing with call admission control to meet QoS requirements in cellular networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 9, pp. 898–910, Sep. 2002.

[24] P. Hande, Z. Shengyu, and C. Mung, "Distributed rate allocation for inelastic flows," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1240–1253, Dec. 2007.

[25] J. Dreo, A. Petrowski, E. Taillard, and P. Siarry, *Metaheuristics for Hard Optimization Simulated Annealing, Tabu Search, Evolutionary and Genetic Algorithms, Ant Colonies, ... Methods and Case Studies*. New York: Springer-Verlag, 2006.
[26] T. El-Ghazali, *Metaheuristics: From Design to Implementation*. Chichester, U.K.: Wiley, 2009.
[27] D. Gözüpek and F. Alagöz, "Genetic algorithm-based scheduling in cognitive radio networks under interference temperature constraints," *Int. J. Commun. Syst.*, vol. 24, no. 2, pp. 239–257, Feb. 2011.
[28] S. Kim, E. Lee, M. Choi, H. Jeong, and S. Seo, "Design optimization of vehicle control networks," *IEEE Trans. Veh. Technol.*, vol. 60, no. 7, pp. 3002–3016, Sep. 2011.
[29] D. Gözüpek, G. Genç, and C. Ersoy, "Channel assignment problem in cellular networks: A reactive tabu search approach," in *Proc. IEEE Int. Symp. Comput. Inf. Sci.*, 2009, pp. 298–303.
[30] D. Williamson and D. Shmoys, *The Design of Approximation Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2011.
[31] S. Assmann, D. Johnson, D. Kleitman, and J. Leung, "On a dual version of the one-dimensional bin packing problem," *J. Algorithms*, vol. 5, no. 4, pp. 502–525, 1984.
[32] K. Jansen and R. Solis-Oba, "An asymptotic fully polynomial time approximation scheme for bin covering," in *Proc. 13th Int. Symp. Algor. Comput.*, 2002, vol. 2518, Lecture Notes in Computer Science, pp. 175–186.
[33] D. Gözüpek, M. Shalom, and F. Alagöz, "A graph theoretic approach to scheduling in cognitive radio networks," *IEEE/ACM Trans. Netw.*, 2012, submitted for publication. [Online]. Available: http://www.cmpe.boun.edu.tr/~gozupek/gozupek-GraphCogSch-2012.pdf
[34] CPLEX, 2011. [Online]. Available: http://www.ilog.com/products/cplex

Didem Gözüpek received the Ph.D. degree in computer engineering from Bogazici University, Istanbul, Turkey, in 2012, the M.S. degree in electrical engineering from the New Jersey Institute of Technology (NJIT), Newark, in 2005, and the B.S. degree in telecommunications engineering (High Honors) from Sabancý University, Istanbul.

From 2005 to 2008, she was an R&D Engineer with a telecommunications company in Istanbul. She is currently a Researcher with the Telematics & Informatics Research Center, Bogazici University. Her research interests are scheduling and resource allocation in communication networks, algorithmic graph theory, and approximation algorithms.

Dr. Gözüpek received the Best Ph.D. Thesis Award from Bogazici University in 2012 and the Aselsan Ph.D. Fellowship in 2011.

Başak Eraslan received the B.S. degree (Honors) in computer engineering from the Middle East Technical University, Ankara, Turkey, in 2007 and the M.S. degree in computer engineering from Bogazici University, Istanbul, Turkey, in 2011.

From 2007 to 2012, she was an R&D engineer with The Scientific and Technological Research Council of Turkey–Center of Research for Advanced Technologies of Informatics and Information Security, Kocaeli, Turkey. She was involved with the Software Defined Radio SCA Testbed Project and the Remote Management Center of IP Crypto Devices Project. She is currently with the Department of Computer Engineering, Bogazici University. Her current research interests include cognitive radio networks, dynamic spectrum access technologies, and resource allocation algorithms. She is a member of the Software Defined Radio Forum.

Fatih Alagöz received the B.Sc. degree in electrical engineering from the Middle East Technical University, Ankara, Turkey, in 1992 and the M.Sc. and D.Sc. degrees in electrical engineering from The George Washington University, Washington, DC, in 1995 and 2000, respectively.

He is an Associate Professor with the Department of Computer Engineering, Bogazici University, Istanbul, Turkey. From 2001 to 2003, he was with the Department of Electrical Engineering, United Arab Emirates University, Al Ain, UAE. In 1993, he was a Research Engineer with a missile manufacturing company called, Muhimmatsan AS, Turkey. He is currently with the Department of Computer Engineering, Bogazici University. He has contributed/managed ten research projects for various agencies/organizations, including the US Army of Intelligence Center, the Naval Research Laboratory, the UAE Research Fund, the Turkish Scientific Research Council, the State Planning Organization of Turkey, BAP, etc. He has edited five books and published more than 100 scholarly papers in selected journals and conferences. His current research interests are in the areas of satellite networks, wireless networks, sensor networks, and ultra-wideband communications.

Dr. Alagöz is a member of the IEEE Satellite and Space Communications Technical Committee. He has numerous professional awards. He is the Satellite Systems Advisor to the Kandilli Earthquake Research Institute, Istanbul. He has served on several major conference technical committees and organized and chaired many technical sessions at many international conferences.