A Graph-Theoretic Approach to Scheduling in Cognitive Radio Networks

Didem Gözüpek, Mordechai Shalom, and Fatih Alagöz

***Abstract*—We focus on throughput-maximizing, max-min fair, and proportionally fair scheduling problems for centralized cognitive radio networks. First, we propose a polynomial-time algorithm for the throughput-maximizing scheduling problem. We then elaborate on certain special cases of this problem and explore their combinatorial properties. Second, we prove that the max-min fair scheduling problem is NP-Hard in the strong sense. We also prove that the problem cannot be approximated within any constant factor better than 2 unless $P = \mathrm{NP}$. Additionally, we propose an approximation algorithm for the max-min fair scheduling problem with approximation ratio depending on the ratio of the maximum possible data rate to the minimum possible data rate of a secondary users. We then focus on the combinatorial properties of certain special cases and investigate their relation with various problems such as the multiple-knapsack, matching, terminal assignment, and Santa Claus problems. We then prove that the proportionally fair scheduling problem is NP-Hard in the strong sense and inapproximable within any additive constant less than $\log(4/3)$. Finally, we evaluate the performance of our approximation algorithm for the max-min fair scheduling problem via simulations. This approach sheds light on the complexity and combinatorial properties of these scheduling problems, which have high practical importance in centralized cognitive radio networks.**

***Index Terms*—Algorithmic graph theory, approximation algorithms, cognitive radio networks, dynamic spectrum access, resource allocation, scheduling.**

I. Introduction

WIRELESS networks are currently characterized by a fixed spectrum assignment policy. Due to the proliferation of wireless technologies and services, the demand for the radio spectrum continuously increases. This increasing demand together with the fixed spectrum assignment policy creates a shortage of spectrum. However, this shortage is artificial because studies show that a very small portion of the assigned spectrum is actually utilized [1]. This situation calls for techniques that utilize the radio spectrum more efficiently.

To overcome the inefficiency in the spectrum usage, the dynamic spectrum access (DSA) concept has been introduced by researchers in the wireless networking community. DSA hinges upon the idea of having an intelligent device that opportunistically utilizes the temporarily unused parts of the spectrum and vacates them as soon as the licensed owner of that spectrum band resumes its operation. These intelligent devices are called cognitive radios. The licensed owners of the spectrum are called primary users (PUs), and the cognitive radio devices are called the secondary users (SUs). PUs are unaware of the SUs, and SUs are obliged not to disturb the PUs. In a centralized cognitive radio network (CRN), cognitive base station (CBS) is the central entity that has cognitive capabilities; in other words, a CBS is aware of the DSA concept. The CBS controls and guides the SUs in its service area by ensuring that the PUs in the region are not disturbed by the data communication of the SUs with the CBS.

Opportunistic scheduling concept is based on the exploitation of the time-varying channel conditions in wireless networks to increase the overall performance of the system [2]–[4]. The opportunistic nature of cognitive radio networks is very much in line with the opportunistic scheduling paradigm; i.e., opportunistic scheduling enables efficient opportunistic utilization of the time-varying PU activities by the SUs. In our previous work [5] and its extension [6], we formulated throughput-maximizing, max-min fair, weighted max-min fair, and proportionally fair scheduling problems for centralized cognitive radio networks. All schedulers make frequency, time-slot, and data rate allocations to the SUs. Furthermore, all of them ensure that the PUs in the service area of the CBS are not disturbed, no collisions occur among the SUs, reliable communication of the SUs with the CBS is maintained, each SU is assigned at least one time-slot whenever possible, and the number of frequencies assigned to an SU in a particular time-slot is not more than the number of its transceivers (antennas) for data transmission. Throughput-maximizing scheduler (TMS) aims to maximize the overall throughput and therefore opportunistically favors the SUs that have better channel conditions such as being more distant to the active PUs. A scheme designed only to maximize the overall throughput can be unfairly biased, especially when there are users with persistently bad channel conditions. Therefore, maintaining some notion of fairness is a vital criterion that opportunistic schedulers should address. Max-min fair scheduler (MMFS) achieves max-min fairness by maximizing the throughput of the SU that has the minimum throughput among all the SUs. When MMFS is executed, an SU with very bad channel conditions can drive the total throughput of all SUs to very low values. On the other hand, the notion of proportional fairness aims to provide a tradeoff between user satisfaction and total throughput [7], [8]. Accordingly, proportionally fair scheduler (PFS) provides proportional fairness by maximizing the product of the SU throughput values and

Manuscript received May 30, 2012; revised July 25, 2013; accepted December 13, 2013; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor K. Kar. Date of publication January 27, 2014; date of current version February 12, 2015. This work was supported by the State Planning Organization of Turkey (DPT) under Grant No. DPT-2007K 120610 and the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. 109E256.

D. Gözüpek is with the Department of Computer Engineering, Gebze Institute of Technology, Kocaeli 41400, Turkey (e-mail: didem.gozupek@gyte.edu.tr).

M. Shalom is with the Department of Computer Science, Tel Hai College, Upper Galilee 12210 , Israel (e-mail: cmshalom@telhai.ac.il).

F. Alagöz is with the Department of Computer Engineering, Bogazici University, Istanbul 34342, Turkey (e-mail: fatih.alagoz@boun.edu.tr).

Digital Object Identifier 10.1109/TNET.2013.2297441

thereby provides a tradeoff between TMS and MMFS [6]. The work in [5] formulates the TMS problem as an integer linear programming problem and provides its numerical evaluation. The work in [6] extends [5] by also formulating MMFS and PFS problems through integer programming formulations and provides heuristic algorithms.

In this paper, we present a formal combinatorial analysis to the TMS, MMFS, and PFS problems formulated in [5] and [6] and provide computational hardness results. Our contributions can be summarized as follows.

1) In Section III-A, we propose more compact equivalent integer programming formulations (with respect to those considered in [5] and [6]) for all three problems.
2) In the same section, we also propose a polynomial-time algorithm for the TMS problem (Theorem 1).
3) In Section III-B, we investigate the combinatorial properties of certain special cases of the TMS problem and analyze their relationship with various combinatorial optimization problems in the literature such as the multiple-knapsack and terminal assignment problems.
4) In Section III-C, we investigate the hardness of approximation of the MMFS problem and some of its special cases.
5) In the same section, we propose an approximation algorithm for the MMFS problem with approximation ratio depending on the ratio of the maximum possible data rate to the minimum possible data rate of a secondary user (Theorem 2) and evaluate its performance via simulations.
6) In Section III-D, we prove that the PFS problem is NP-Hard in the strong sense and inapproximable within any additive constant less than $\log(4/3)$ even in its special cases.

Graph-theoretic techniques have previously been used for peer-to-peer networks [9] and wavelength-division multiplexing (WDM) networks [10]. Some scheduling problems in wireless networks have also been addressed by using graph-theoretic techniques [11], [12]. On the other hand, graph-theoretic approaches in CRNs are mainly based on simple variants of graph coloring problems. Authors in [13] formulate a spectrum allocation problem considering the different channel availabilities at different nodes in a CRN and show that it is a list coloring problem. The work in [14] also addresses dynamic spectrum allocation problem using list coloring and proposes centralized and decentralized suboptimal algorithms. The authors of [15] reduce the spectrum allocation problems that they consider to a variant of graph coloring problem. Unlike the works in [5] and [6], the work in [15] does not have a time aspect, and therefore it focuses mainly on channel allocation rather than scheduling. The authors in [16] also focus on centralized and distributed DSA problem in CRNs by proving NP-hardness of their problem and presenting an approximation algorithm. Unlike [6], the work in [16] does not focus on max-min fairness. Furthermore, the schedulers proposed in [5] and [6] have also a temporal notion of fairness since they ensure that each SU is assigned at least one time-slot. This temporal notion of fairness also does not exist in [16].

To the best of our knowledge, very few works provide a combinatorial approach for scheduling in CRNs [17], [18]. Unlike most work in the literature that uses graph coloring arguments, we use various techniques from matching theory. We relate our scheduling problems to numerous combinatorial optimization problems such as knapsack, terminal assignment, generalized assignment, partition, and Santa Claus problems.

The rest of the paper is organized as follows. In Section II, we provide the background information related to the TMS, MMFS, and PFS scheduling problems. We discuss our proposed solutions and provide NP-hardness proofs as well as remarks about certain special cases of these problems in Section III. We elaborate on the practical implications of our results in Section IV and present the simulation results in Section V. Finally, we conclude the paper in Section VI.

II. Background

In all of the schedulers in this work, the scheduling decisions are made for a duration of T time-slots during which the network conditions are assumed to be fairly stable due to the SU and PU velocities as well as the PU spectrum occupancies. In the first stage of the schedulers, U_{if} values denoting the maximum number of packets per time-slot that can be sent by SU i using frequency f in the entire scheduling period are found. U_{if} values are calculated such that the PUs are guaranteed not to be disturbed and the reliable communication of the SUs with the CBS is ensured. In order to derive a formula for U_{if}, in [6] we first find the following expression for P_{xmt}^{ift}, which denotes the maximum permissible transmission power for SU i using frequency f in time-slot t:

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

where $P_{IF_{\max}}^{fj}$ denotes the maximum tolerable interference power of PU j for frequency f, d_{ijt} is the distance between SU i and PU j in time-slot t, λ_f is the wavelength of frequency f, h_{ijt} is the fading coefficient of the channel between SU i and PU j in time-slot t, and Φ_{CBS}^{ft} denotes the set of PUs that are in the coverage area of the CBS and carrying out their communication using frequency f in time-slot t. Note here that it is possible that $|\Phi_{\text{CBS}}^{ft}| > 1$ because primary users are allowed to use the same frequency at the same time in some wireless technologies. For instance, users in a cellular network can use the same frequency as long as the distance between them is larger than a particular distance referred to as the frequency reuse distance. In essence, $(\frac{\lambda_f}{4\pi d_{ijt}} \times |h_{ijt}|)^2$ 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 [6].

Subsequently, using Shannon's capacity function for Gaussian channels, we derive the following formula to ensure reliable communication between the SUs and the CBS [6]:

$$U_{ift} = \left\lfloor \ln\left(1 + \left(P_{xmt}^{ift} \times \frac{\left|\frac{\lambda_f}{4\pi d_{it}^{\text{CBS}} \times h_{it}^{\text{CBS}}}\right|^2}{\xi}\right)\right)\right\rfloor \tag{2}$$

where U_{ift} denotes the maximum number of packets that can be sent by SU i using frequency f in time-slot t of the pertinent scheduling period, d_{it}^{CBS} represents 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 ξ

symbolizes the sum of noise power and interference power from the PUs to each SU.

Parameters such as d_{it}^{CBS} and h_{ijt} are detected by the CBS via techniques such as geolocation system and common control channel. All the parameters are assumed to remain constant for each time-slot t of the scheduling period since the scheduling period is a time duration during which the network conditions remain fairly stable. Therefore, U_{if} instead of U_{ift} is used to represent the maximum number of packets that can be sent by SU i using frequency f in any time-slot of the considered scheduling period. Schedulers are executed by the CBS; thus, centralized scheduling is considered.

In all of the scheduling problems, $\mathcal{N}$ denotes the set of N SUs, $\mathcal{F}$ represents the set of F frequencies, and $\mathcal{T}$ denotes the set of T time-slots. In other words, $\mathcal{N} = \{1, 2, \ldots, N\}$, $\mathcal{F} = \{1, 2, \ldots, F\}$, and $\mathcal{T} = \{1, 2, \ldots, T\}$. Furthermore, a_i denotes the number of antennas of SU i and X_{ift} is a binary decision variable that equals 1 if SU i transmits using frequency f in time-slot t, and 0 otherwise. We use the standard vector notation and denote, for instance, by X the vector of all values X_{ift}.

A. TMS Problem

Given the values for N, F, T, a_i, and U_{if}, the TMS problem is to find the vector X given by the following binary integer linear programming (ILP) formulation:

$$\max \sum_{i=1}^{N} \sum_{f=1}^{F} \sum_{t=1}^{T} U_{if} X_{ift} \tag{3}$$

s.t.

$$\sum_{f=1}^{F} \sum_{t=1}^{T} X_{ift} \geq 1 \qquad \forall i \in \mathcal{N} \tag{4}$$

$$\sum_{i=1}^{N} X_{ift} \leq 1 \qquad \forall f \in \mathcal{F}, \forall t \in \mathcal{T} \tag{5}$$

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

$$X_{ift} \in \{0, 1\} \qquad \forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T}. \tag{7}$$

In this formulation, the objective function in (3) maximizes the total throughput of all the SUs in the centralized CRN governed by the CBS. The constraint in (4) guarantees that at least one time-slot is assigned to every SU and hence provides a temporal notion of fairness. Besides, (5) ensures that at most one SU can transmit in a particular time-slot and frequency, and hence prevents collisions between the SUs. Moreover, (6) represents the fact that an SU i cannot transmit at the same time using frequencies more than the number a_i of its transceivers (antennas) since each transceiver can be tuned to at most one frequency at a time. The rationale for these constraints can be found in [5]. In this paper, we focus on the combinatorial properties of the TMS problem and propose a polynomial-time algorithm for it. We also investigate certain special cases of the problem.

B. MMFS Problem

In order to provide fairness for a longer timescale, the work in [6] proposes a windowing mechanism that considers the throughput of the SUs in the recent scheduling periods. Through this mechanism, temporary throughput losses of the SUs due to rapidly changing network conditions can be compensated in the subsequent scheduling periods. The number of considered scheduling periods in the recent past is referred to as the window size, during which the changes in the network conditions are considered to be important. If the window size is too large, the scheduler will be too responsive to small changes in the network conditions. If it is too small, the scheduler will be inflexible in compensating for the temporary fluctuations in the network conditions. Both MMFS and PFS have this windowing mechanism.

The aggregate average throughput of SU i in the last φ scheduling periods is defined as R_{φ}^{i}, in packets per time-slot. In other words, φ is the window size. All of the R_{φ}^{i} values are initialized to 0 for all the SUs. Let us define $\overline{R_{\varphi}^{i}}$, which is based on an exponentially weighted low pass filter, as follows:

$$\overline{R_{\varphi}^{i}} = \left(1 - \frac{1}{\min(k, \varphi)}\right) R_{\varphi}^{i} + \frac{1}{\min(k, \varphi)} \frac{\sum_{f=1}^{F} \sum_{t=1}^{T} U_{if} X_{ift}}{T}. \tag{8}$$

Here, $\frac{\sum_{f=1}^{F} \sum_{t=1}^{T} U_{if} X_{ift}}{T}$ denotes the throughput of SU i in the current scheduling period k and $\frac{1}{\min(k,\varphi)}$ is the weight given to it. On the other hand, $\left(1 - \frac{1}{\min(k,\varphi)}\right)$ is the weight given to the value of R_{φ}^{i}, i.e., the aggregate average throughput of SU i at the start of the current scheduling period. At the end of each scheduling period k, the value for R_{φ}^{i} is updated as $R_{\varphi}^{i} \leftarrow \overline{R_{\varphi}^{i}}$. Since both k and φ are constant in a particular scheduling execution, without loss of generality we use φ instead of $\min(k, \varphi)$ in the rest of this paper.

Given the values for $N, F, T, a_i, \varphi, R_{\varphi}^{i}$, and U_{if}, MMFS problem is to find the vector X given by the following ILP formulation:

$$\max \quad Z \tag{9}$$

s.t.

$$Z \leq \left(1 - \frac{1}{\varphi}\right) R_{\varphi}^{i} + \frac{1}{\varphi} \frac{\sum_{f=1}^{F} \sum_{t=1}^{T} U_{if} X_{ift}}{T} \qquad \forall i \in \mathcal{N} \tag{10}$$

(4), (5), (6), and (7)

where (9) and (10) together maximize $\min_{i \in \mathcal{N}} \overline{R_{\varphi}^{i}}$. In this paper, we focus on the graph-theoretical properties of the MMFS problem and show that it cannot be approximated within any constant factor better than 2. Furthermore, we propose an approximation algorithm for it and investigate the combinatorial properties of some special cases.

C. PFS Problem

Given the values for N, F, T, a_i, φ, R^i_φ, and U_{if}, the PFS problem is to find the vector X given by the following binary integer programming formulation:

$$\max \sum_{i=1}^{N} \log \left(\left(1-\frac{1}{\varphi}\right) R^i_\varphi + \frac{1}{\varphi} \frac{\sum_{f=1}^{F}\sum_{t=1}^{T} U_{if} X_{ift}}{T} \right) \quad (11)$$

s.t.

(4), (5), (6), and (7).

This integer programming formulation of the PFS problem maximizes a nonlinear concave objective function subject to linear constraints. In this paper, we prove that the PFS problem is NP-Hard in the strong sense.

III. Proposed Solutions

A. Preliminaries

Approximation Algorithms: 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., $O_\Pi(s) \geq \frac{O^*_\Pi(I)}{\rho}$. 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 $P = NP$. 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 $\Pi \preceq_{\mathrm{APX}} \Pi'$. Π and Π' are said to be equivalent under approximation preserving reductions if $\Pi \preceq_{\mathrm{APX}} \Pi'$ and $\Pi' \preceq_{\mathrm{APX}} \Pi$. A polynomial-time approximation scheme (PTAS) for a problem Π is an algorithm *ALG* that takes as input both the instance I and an error bound ϵ, runs in time polynomial in $|I|$, and has approximation ratio $(1+\epsilon)$. In fact, such an algorithm *ALG* is a family of algorithms ALG_ϵ that has approximation ratio $(1+\epsilon)$ for any instance I. The running time of a PTAS is required to be polynomial in $|I|$ for every fixed ϵ but can be different for different ϵ.

Matchings: **I**-*matching* and **I**-*factor* are defined as follows [19]: Let $G = (V, E)$ be a (multi)graph with weight function $w : E \to \mathbb{R}$ on its edges, and let $\mathbf{I}$ be a function associating an interval of natural numbers with each vertex in V. We denote by $\delta_G(v)$ the set of incident edges of v in G, i.e., $\delta_G(v) = \{e \in E \,|\, v \in e\}$, and $d_G(v) = |\delta_G(v)|$ is the degree of v in G. An **I**-*matching* is a function $\mathbf{m}$: $E \to \mathbb{N}$ such that for $v \in V, \sum_{e\in\delta_G(v)} \mathbf{m}(e)$ lies in the interval $\mathbf{I}(v)$. An **I**-*factor* is an **I**-*matching* such that $\mathbf{m}$: $E \to \{0,1\}$. A *matching* is an **I**-*factor* such that $\mathbf{I}(v) = \{0,1\}$ for each $v \in V$. In particular, if $\mathbf{I}(v) = \{1\}$ for each $v \in V$, it is called a *perfect matching*. A *maximum weighted* **I**-*factor* is an **I**-*factor* $\mathbf{m}$ such that $\sum_{e\in E} \mathbf{m}(e) \cdot w(e)$ is maximized. A maximum weighted **I**-*factor* can be found in polynomial time [20], [21]. An **I**-*factor* $\mathbf{m}$ corresponds to a sub(multi)graph M of G such that the multiplicity of the edges of G in M is given by the function $\mathbf{m}$. With slight abuse of notation, M will also be called an **I**-*factor*.

Let $G = (U, V, E)$ be an edge weighted bipartite (multi)graph. Let $b(u) = \sum_{e\in\delta_G(u)} \mathbf{m}(e) \cdot w(e), \forall u \in U$. A *max-min weighted* **I**-*factor* is an **I**-*factor* $\mathbf{m}$ such that $\min_{u\in U} b(u)$ is maximized. In the rest of this paper, we use the terms *maximum* **I**-*factor* and *max-min* **I**-*factor* to mean *maximum weighted* **I**-*factor* and *max-min weighted* **I**-*factor*, respectively. Since our focus is on edge weighted graphs, the implication to the *weighted* case is implicit.

Santa Claus *Problem:* The Santa Claus problem is defined in [22]: Santa Claus has a set of presents that she wants to distribute among a set of kids. Each present has a different value for different kids. The happiness of a kid is the sum of the values of the presents she gets. Santa's goal is to distribute the presents in such a way that the least happy kid is as happy as possible. The problem is also known as the Max-Min Fair Allocation Problem [23].

In the sequel, we propose equivalent simpler ILP formulations for the TMS, MMFS, and PFS problems.

Lemma 1: Let Π be an optimization problem that involves the variables X_{ift} only in the form $\sum_{t=1}^{T} X_{ift}$ in its objective function O_π and also in all its constraints except constraints (5)–(7). Let Π' be the optimization problem obtained from Π by substituting

$$Y_{if} = \sum_{t=1}^{T} X_{ift} \qquad \forall i \in \mathcal{N},\ \forall f \in \mathcal{F} \quad (12)$$

in O_Π and all the constraints except (5)–(7) and by replacing the constraints (5)–(7) by the following constraints:

$$\sum_{i=1}^{N} Y_{if} \leq T \qquad \forall f \in \mathcal{F} \quad (13)$$

$$\sum_{f=1}^{F} Y_{if} \leq a_i \cdot T \qquad \forall i \in \mathcal{N} \quad (14)$$

$$Y_{if} \in \mathbb{N} \qquad \forall i \in \mathcal{N},\ \forall f \in \mathcal{F}. \quad (15)$$

Then, Π and Π' are equivalent under approximation-preserving reductions.

Proof: It is sufficient to show that any solution X of Π can be converted in polynomial time to a solution Y of Π' with $O_{\Pi'}(Y) = O_\Pi(X)$, and vice versa. In particular, this holds also for an optimal solution X^*; thus, both problems have the same optimum, and a ρ-approximate solution for one problem corresponds to a ρ-approximate solution for the other.

One direction is immediate. Indeed, given a solution X of Π, the solution Y defined by (12) clearly satisfies all the constraints except (13)–(15) because these constraints were obtained by substituting (12) in the original constraints that are satisfied by our assumption. Moreover, $O_{\Pi'}(Y) = O_\Pi(X)$ for the same reason. On the other hand, note that the constraints (13)–(15) are obtained by summing up T inequalities of type (5)–(7), respectively, each of which is satisfied by our assumption.

Now assume that we are given a solution Y of Π'. Any decomposition of Y into $X_{ift}, \forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T}$ satisfying (12) clearly satisfies all the constraints except (5)–(7) and also $O_\pi(X) = O_{\pi'}(Y)$ for the same reason as in the previous paragraph. In the sequel, we give a polynomial-time algorithm that finds such a decomposition and also satisfies the constraints (5)–(7). From the vector Y, we build a bipartite multigraph $G = (U, V, E)$ such that $U = \{u_1, \ldots, u_N\}, V = \{v_1, \ldots, v_F\}$, and there are Y_{if} parallel edges connecting $u_i \in U$ and $v_f \in V$. The degree of a vertex $v_f \in V$ is at most T by constraint (13), and the degree of a vertex $u_i \in U$ is at most $a_i \cdot T$ by constraint (14). Consider the bipartite graph $G' = (U', V, E')$ obtained from G by replacing each vertex $u_i \in U$ with a_i vertices in U' and dividing the at most $a_i \cdot T$ edges adjacent to u_i to these new vertices such that each vertex receives at most T edges. The degree of each vertex of G' is at most T. Let $G'' = (U'', V'', E'')$ be the graph obtained from G' by adding $\|U'| - |V\|$ dummy vertices to either U' or V so that $|U''| = |V''|$ and adding dummy edges as long as there are vertices with degree less than T. G'' is a T-regular bipartite graph. A well-known classical result by the works of König, Hall, and Tutte [24]–[26] implies that such a graph contains a perfect matching and it can be found in polynomial time using the Hungarian algorithm. Removing a perfect matching from G'', one remains with a $(T-1)$-regular bipartite graph. Applying this inductively, G'' is partitioned into T perfect matchings $M_1'', \ldots, M_T''$. By removing all the dummy edges and vertices of G'' from these perfect matchings, we obtain T matchings $M_1', \ldots, M_T'$ of G'. In each matching $M_t', \forall t \in \mathcal{T}$, we contract back the a_i vertices of U' to the node u_i for every $i \in \mathcal{N}$ to get T bipartite graphs $M_1, \ldots, M_T$, where each vertex $u_i \in U$ has degree at most a_i and each node $v_f \in V$ has degree at most 1. Let $X_{ift} = 1$ if u_i is adjacent to v_f in M_t, and 0 otherwise. We conclude that X satisfies (5)–(7). ■

Note that the formulation of Π' has less variables and constraints than the formulation of Π; therefore, it is computationally more efficient, whether we are looking for exact solutions or using optimization software such as CPLEX [27] to find nearly optimal solutions.

Corollary 1: The TMS problem is equivalent to finding the vector Y given by the following ILP formulation:

$$\max \sum_{i=1}^{N} \sum_{f=1}^{F} U_{if} Y_{if} \tag{16}$$

s.t.

$$\sum_{f=1}^{F} Y_{if} \geq 1 \qquad \forall i \in \mathcal{N} \tag{17}$$

(13), (14), and (15).

Corollary 2: The MMFS problem is equivalent to finding the vector Y given by the following ILP formulation:

$$\max \quad Z \tag{18}$$

s.t.

$$Z \leq \left(1 - \frac{1}{\varphi}\right) R_\varphi^i + \frac{1}{\varphi} \frac{\sum_{f=1}^{F} U_{if} Y_{if}}{T} \qquad \forall i \in \mathcal{N} \tag{19}$$

(17), (13), (14), and (15).

Algorithm 1: THRMAX

Require: $\mathcal{N}, \mathcal{F}, a_i, \forall i \in \mathcal{N}$.
Ensure: Y_{if} values $\forall i \in \mathcal{N}, \forall f \in \mathcal{F}$.

1: Build an edge weighted bipartite (multi)graph $G = (U, V, E)$ as follows:
2: For each $i \in \mathcal{N}$ add a vertex u_i to U.
3: For each $f \in \mathcal{F}$ add a vertex v_f to V.
4: For each pair of vertices $u_i \in U$ and $v_f \in V$, add the edge $\{u_i, v_f\}$ to E with weight U_{if}.
5: Define the following function $\mathbf{I}$, which associates an interval of natural numbers with each vertex in G:
6: $\mathbf{I}(u_i) = [1, a_i \cdot T], \forall i \in \mathcal{N}$
7: $\mathbf{I}(v_f) = [0, T], \forall f \in \mathcal{F}$
8: Find a maximum weighted $\mathbf{I}$-*factor* M of G.
9: For all $i \in \mathcal{N}$ and $f \in \mathcal{F}$, let Y_{if} be equal to the number of edges between vertices u_i and v_f in the $\mathbf{I}$-*factor* M.

Corollary 3: The PFS problem is equivalent to finding the vector Y given by the following IP formulation:

$$\max \sum_{i=1}^{N} \log \left(\left(1 - \frac{1}{\varphi}\right) R_\varphi^i + \frac{1}{\varphi} \frac{\sum_{f=1}^{F} U_{if} Y_{if}}{T} \right) \tag{20}$$

s.t.

(17), (13), (14), and (15).

B. Algorithms for the TMS Problem

1) Polynomial-Time Algorithm:

Theorem 1: There exists a polynomial-time algorithm for the TMS problem.

Proof: We show in the following that Algorithm THRMAX is an optimal algorithm for the TMS problem. Notice here that the lower bound 1 of $\mathbf{I}(u_i)$ in line 6 is equivalent to constraint (17). Besides, the upper bound T of $\mathbf{I}(v_f)$ in line 7 is equivalent to (13), and the upper bound $a_i \cdot T$ of $\mathbf{I}(u_i)$ in line 6 is equivalent to (14).

Clearly, all the steps except line 8 of the algorithm can be performed in polynomial time. Line 8 calculates a maximum weighted $\mathbf{I}$-*factor*. This problem is solvable in polynomial time for general graphs [20]. However, as G is a bipartite graph, its incidence matrix is totally unimodular. The matrix of our linear program is obtained from the incidence matrix of the bipartite graph G by duplicating some of the rows. Therefore, our constraint matrix is totally unimodular, too. As such, the vertices of the polyhedron corresponding to the linear constraints are integral, in particular any optimal solution of it is integral. We conclude that the integrality constraints (15) are redundant, thus they can be removed. The linear programming relaxation obtained in this way can be solved in polynomial time. ■

2) Special Cases:

Case 1: Ignore constraint (17) in the TMS formulation. Recall that this constraint ensures that each SU is assigned at least one

time-slot and therefore achieves temporal fairness. Without this temporal fairness constraint in the problem formulation, some SUs with bad channel conditions may end up being unable to send any packets for a long time. Some transport-layer protocols such as TCP close the connection if no packets are received for a certain amount of time. Constraint (17) gives each SU the opportunity to send at least something and therefore avoids this undesired disconnection situation caused by transport-layer protocols. A CBS operator may prefer to ignore constraint (17) if it does not have any concern with this transport-layer behavior or temporal fairness. Ignoring constraint (17) causes the TMS scheduler to behave more opportunistically and increases the possibility to increase the total throughput at the expense of sacrificing from temporal fairness.

In this case, we can solve the following ILP and set $X_{ift} = Y_{if}, \forall i \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \mathcal{T}$:

$$\max \sum_{i=1}^{N} \sum_{f=1}^{F} U_{if} Y_{if} \tag{21}$$

s.t.

$$\sum_{i=1}^{N} Y_{if} \leq 1 \qquad \forall f \in \mathcal{F} \tag{22}$$

$$\sum_{f=1}^{F} Y_{if} \leq a_i \qquad \forall i \in \mathcal{N} \tag{23}$$

$$Y_{if} \in \{0,1\} \qquad \forall i \in \mathcal{N},\ \forall f \in \mathcal{F}. \tag{24}$$

Clearly, a method similar to the one in Algorithm THRMAX can be used. The function $\mathbf{I}$ is defined as follows: $\mathbf{I}(u_i) = [0, a_i], \forall i \in \mathcal{N}$, and $\mathbf{I}(v_f) = [0,1], \forall f \in \mathcal{F}$. However, there are other methods to solve this special case. In the following, we discuss these alternative ways and the relation of this special case with other combinatorial optimization problems such as the multiple-knapsack and terminal assignment problems.

Relation With Other Combinatorial Optimization Problems: This problem is a variant of the knapsack problem, where the frequencies correspond to the items and the SUs correspond to the knapsacks each with capacity a_i. The general case of (23) would be $\sum_{f=1}^{F} w_{if} Y_{if} \leq a_i$, where w_{if} values correspond to the weights of the items. Hence, constraint (23) is a special case where $w_{if} = 1, \forall i \in \mathcal{N}, \forall f \in \mathcal{F}$.

In essence, the problem in (21)–(24) is a *multiple-knapsack problem* (MKP) where the item profits (U_{if}) vary with knapsacks and all item sizes are identical. The authors of [28] state that the special case of MKP where the item profits vary with bins and all item sizes are identical is solvable in polynomial time. Besides, the work in [29] presents the case with generalized w_{if} as a variant of the *generalized assignment problem* (GAP), called LEGAP. This problem is also referred to in the operations research literature as the *loading problem*, where the items (frequencies) are loaded into containers (SUs) of different capacities (antennas) such that container capacities are not violated. In the centralized network design literature, this problem is also referred to as the *terminal assignment problem*, where terminals (frequencies) are assigned to the concentrators (SUs). Likewise, the case where all the terminal weights are identical can be solved in polynomial time by alternating chain algorithms [30].

Case 2: Ignore constraints (17) and (14) in the TMS formulation. Notice that ignoring constraint (14) is equivalent to having $a_i \geq F, \forall i \in \mathcal{N}$. Then, the optimal solution is to assign each frequency f in every time-slot to the SU that has the highest U_{if} value for frequency f breaking ties arbitrarily. In other words, $X_{ift} = 1$ if $i = \arg\max_i U_{if}$, and 0 otherwise, $\forall f \in \mathcal{F}, \forall t \in \mathcal{T}$.

Case 3: Ignore constraint (17) and assume that $a_i = 1$ $\forall i \in \mathcal{N}$. In this case, a maximum weighted bipartite matching problem between the SUs ($i \in \mathcal{N}$) and the frequencies ($f \in \mathcal{F}$) can be solved, and the result of this matching can be applied in every time-slot $t \in \mathcal{T}$. Hungarian algorithm [31] can be used to solve the maximum weighted bipartite matching problem. This technique can be extended also to the case where the a_i values are small by replacing each node u_i corresponding to SU i with a_i nodes, each of which corresponds to the antennas of SU i. In this case, the reason for a_i values to be necessarily small is to obtain a polynomial reduction.

C. Algorithms for the MMFS Problem and Its Complexity

We present in this section a graph-theoretic approach to the MMFS problem. In addition to the MMFS problem, the work in [6] also formulates the weighted MMFS problem, where constraint (19) is replaced by the following:

$$Z' \leq \frac{\left(1 - \frac{1}{\varphi}\right) R^i_{\varphi} + \frac{1}{\varphi} \frac{\sum_{f=1}^{F} U_{if} Y_{if}}{T}}{\eta_i} \qquad \forall i \in \mathcal{N} \tag{25}$$

where η_i is the relative importance of SU i. Weighted MMFS is essentially the same computational problem as the MMFS problem, where U_{if} values are replaced by $\frac{U_{if}}{\eta_i}$ and R^i_{φ} values are replaced by $\frac{R^i_{\varphi}}{\eta_i}$. All the variables except Z' and Y_{if} are again input variables. The only difference is that U_{if} values in MMFS are integers, whereas the $\frac{U_{if}}{\eta_i}$ values in the weighted MMFS are not necessarily integers. Since they appear only in the objective function, they do not affect the computational hardness of the problem. In the rest of this section, we mainly refer to the unweighted MMFS problem. However, the entire analysis is valid for the weighted MMFS problem as well.

1) Hardness Results:

Lemma 2: SANTA CLAUS $\preceq_{APX}$ MMFS.

Proof: Consider a special case of the MMFS problem where $T = 1$, $\varphi = 1$, and $a_i \geq F, \forall i \in \mathcal{N}$. The optimization in this case corresponds to distributing the frequencies to the SUs in such a way that the SU having the least throughput receives as much throughput as possible. Because $a_i \geq F$ $\forall i \in \mathcal{N}$, there is practically no upper bound on the number of frequencies that an SU can receive. This special case is exactly the SANTA CLAUS problem, where kids are replaced by SUs and presents are replaced by frequencies. ■

*Restricted Santa Claus Problem (*R-SANTA CLAUS*):* Due to the difficulty of the SANTA CLAUS problem, the attention in the theoretical computer science community has shifted toward

more special cases. One special case is the so-called R-SANTA CLAUS problem, where every present f has the same value U_f for every kid interested in that present, i.e., $U_{if} \in \{U_f, 0\}$. The authors in [32] have shown that it is NP-Hard to approximate the R-SANTA CLAUS problem within any constant factor better than 2. The proof is very similar the proof in Lenstra *et al.* [33], which proves that the problem of minimum makespan scheduling in unrelated parallel machines cannot be approximated within any constant factor better than $3/2$ unless $P = \text{NP}$.

Degree Two and Symmetric Degree Two SANTA CLAUS *Problems (*D2-SANTA CLAUS, SD2-SANTA CLAUS*):* D2-SANTA CLAUS is the variant of SANTA CLAUS problem when each present has a nonzero value for at most two kids. SD2-SANTA CLAUS is a special case of D2-SANTA CLAUS where every present has the same value for both kids interested in that present. In other words, SD2-SANTA CLAUS is a special case of the R-SANTA CLAUS problem where each present has a nonzero value for at most two kids. Clearly, we have the following observations.

Observation 1: SD2-SANTA CLAUS $\preceq_{\text{APX}}$ D2-SANTA CLAUS $\preceq_{\text{APX}}$ SANTA CLAUS.

Observation 2: SD2-SANTA CLAUS $\preceq_{\text{APX}}$ R-SANTA CLAUS $\preceq_{\text{APX}}$ SANTA CLAUS.

It is shown in [34] that SD2-SANTA CLAUS is 2-inapproximable, and a $(2+\epsilon)$-approximation algorithm for D2-SANTA CLAUS running in polynomial time for every $\epsilon > 0$ is presented.

Definition 1: For a graph $G = (V, E)$ with nonnegative edge weights U, $\beta(G, U)$ is the ratio of the maximum edge weight to the minimum nonzero edge weight, i.e., $\beta(G,U) = \frac{\max_{e \in E}\{U_e\}}{\min\{U_e \mid e \in E, U_e > 0\}}$.

Lemma 3: For any $\epsilon > 0$, the MMFS problem is $(\beta(G,U) - \epsilon)$-inapproximable, even when $T = 1, \varphi = 1$, $a_i \geq F, U_{if} \in \{U_f, 0\}$ and each frequency f is usable by at most two SUs.

Proof: This case corresponds to the SD2-SANTA CLAUS problem. The authors in [35] show that SD2-SANTA CLAUS is $(2-\epsilon)$-inapproximable for any $\epsilon > 0$. The graphs used in their reduction satisfy $\beta(G,U) = 2$. Therefore, the lemma holds. ■

Note that having $\varphi = 1$ as in Lemma 3 implies that the scheduler does not give any importance to what has happened in the recent past and focuses only on the current scheduling period. In other words, $\varphi = 1$ implies that the scheduler does not consider the historical throughput information in its current scheduling decision.

Lemma 4: The MMFS problem remains NP-Hard in the strong sense even when $T = 1, \varphi = 1, a_i = 3$, and $U_{if} = U_{jf}$ $\forall i \neq j$ and $i, j \in \mathcal{N}$.

Proof: Recall that the 3-PARTITION problem is the problem of deciding whether a given set of integers can be partitioned into triplets, all of which have the same sum. More precisely, given a multiset S of $3m$ positive integers $n_1, n_2, \ldots, n_{3m}$ such that $\sum_{i=1}^{3m} n_i = m \cdot B$, can S be partitioned into m subsets $S_1, S_2, \ldots, S_m$ such that the sum of the integers in each subset is B? This problem is well known to be NP-Hard in the strong sense. Given such an instance of the 3-PARTITION problem, we can build a complete bipartite graph $G = (U, V, E)$, where $U = \{S_1, S_2, \ldots, S_m\}, V = \{b_1, b_2, \ldots, b_{3m}\}$, and each edge $\{S_i, b_j\}$ has a weight equal to n_j, and $a_i = 3, \forall i \in \mathcal{N}$. Every 3-PARTITION corresponds to a feasible solution of this new instance, and vice versa. The value of the minimum SU i is equal to B if and only if the answer to the 3-PARTITION problem is *YES*. ■

Lemma 5: The MMFS problem remains NP-Hard even when $N = 2, T = 1, \varphi = 1, U_{if} = U_{jf}$, and $a_i \geq F, \forall i \neq j$ and $i, j \in \mathcal{N}$.

Proof: Note that this special case corresponds to the PARTITION problem, i.e., the problem of deciding whether a given set of integers can be partitioned into two subsets that have the same sum. Since the PARTITION problem is NP-Hard in the weak sense, this special case is also NP-Hard in the weak sense. ■

2) Algorithms:

Lemma 6: MMFS $\preceq_{APX}$ max-min **I**-*factor*.

Proof: Since φ and T are both constants, by multiplying both sides of (19) by $\varphi \cdot T$ and substituting $K_i = T(\varphi - 1)R^i_\varphi$ and $Z' = Z\varphi T$ we get the following constraint:

$$Z' \leq K_i + \sum_{f=1}^{F} U_{if} X_{if} \qquad \forall i \in \mathcal{N}. \tag{26}$$

Build a bipartite (multi)graph $G = (U, V, E)$ as follows.

1) For each SU $i \in \mathcal{N}$, add a vertex u_i to U.
2) For each frequency $f \in \mathcal{F}$, add a vertex v_f to V.
3) Add a dummy vertex $\overline{v}$ to V.
4) For each pair of vertices $u_i \in U$ and $v_f \in V$, add Y_{if} edges $\{u_i, v_f\}$ to E each with weight U_{if}.
5) For each vertex $v_i \in U$, add the edge $\{v_i, \overline{v}\}$ to E with weight K_i.

We claim that the MMFS problem is equivalent to the max-min **I**-*factor* problem on the (multi)graph G, where the function **I** is as follows: $\mathbf{I}(u_i) = [2, a_i \cdot T + 1]$ $\forall v_i \in X$, $\mathbf{I}(v_f) = [0, T]$ $\forall v_f \in Y$, and $\mathbf{I}(\overline{v}) = [N, N]$. For an **I**-*factor* H of G, let $Y_{if} = 1$ if and only if the edge $\{u_i, v_f\}$ is in H. $\mathbf{I}(\overline{v})$ implies that $d_H(\overline{v}) = N$. On the other hand, $d_G(\overline{v}) = N$ by the construction. Therefore, all the incident edges $\delta_G(\overline{v})$ of $\overline{v}$ are also in H. Each node $u_i \in U$ has exactly one incident edge from $\delta_G(\overline{v})$ of utility K_i, and this edge is in H. Therefore, $b(u_i) = K_i + \sum_{f=1}^{F} U_{if} \cdot Y_{if}, \forall i \in \mathcal{N}$. In particular, this equality holds for the minimum value, thus $\min_i b(u_i) = \min_i (K_i + \sum_{f=1}^{F} U_{if} \cdot Y_{if})$. We conclude that the value of the objective function is equal to the value of the **I**-*factor*. It remains to show that an **I**-*factor* H corresponds to a feasible solution Y, and vice versa. Recall that a node $u_i \in U$ has one incident edge from $\delta_G(\overline{v})$, thus $d_H(u_i) - 1$ incident edges connecting it to nodes $v_f, \forall f \in \mathcal{F}$. $\mathbf{I}(u_i)$ implies that $2 \leq d_H(u_i) \leq a_i \cdot T + 1$, thus $1 \leq d_H(u_i) - 1 \leq a_i \cdot T$. We conclude that constraints (17) and (13) are satisfied by Y. On the other hand $\mathbf{I}(v_f)$ implies that constraint (14) is satisfied. The opposite direction is shown similarly. ■

Theorem 2: There is a $\beta(G,U)$-approximation algorithm for the max-min **I**-*factor* problem.

Proof: We show in the following that Algorithm MAXMINEQ is a $\beta(G,U)$-approximation algorithm for the max-min **I**-*factor* problem. Consider an optimal max-min **I**-*matching* H^* and the vertex $u_i \in U$ with minimum degree in H^*. Then, $d_{H^*}(u_i) \leq D$ because otherwise all the nodes $u_i \in U$ have $d_{H^*}(u_i) \geq D$

Algorithm 2: MAXMINEQ

Require: $\mathcal{N}, \mathcal{F}, a_i, \forall i \in \mathcal{N}$.
Ensure: Y_{if} values $\forall i \in \mathcal{N}, \forall f \in \mathcal{F}$.

1: Build an edge weighted bipartite (multi)graph $G = (U, V, E)$ as follows:
2: For each SU $i \in \mathcal{N}$, add a vertex u_i to U.
3: For each frequency $f \in \mathcal{F}$, add a vertex v_f to V.
4: For each pair of vertices $u_i \in U$ and $v_f \in V$, add the edge $\{u_i, v_f\}$ to E with weight U_{if}.
5: $D \leftarrow \min_i\{a_i\} \cdot T$.
6: Execute lines 7–9 iteratively by employing a binary search on D to find the maximum possible value of D for which the below steps 7–9 return a feasible solution:
7: Find an **I**-*factor* for:
8: $\mathbf{I}(u_i) = [D, a_i \cdot T], \forall u_i \in U$.
9: $\mathbf{I}(v_f) = [0, T], \forall v_f \in V$.

$+ 1$. Then, H^* constitutes an **I**-*factor* for $\mathbf{I}(u_i) = [D + 1, a_i \cdot T], \forall u_i \in U$ and $\mathbf{I}(v_f) = [0, T], \forall v_f \in V$ contradicting the fact that D is the maximum possible value leading to a feasible solution for the steps 7–9. Therefore, $b(u_i) = \sum_{e \in \delta_{H^*}(u_i)} U_e \leq d_{H^*}(u_i) \cdot \max_{e \in E}\{U_e\} \leq D \cdot \max_{e \in E}\{U_e\}$. Then

$$O(H^*) = \min_{i \in \mathcal{N}} b(u_i) \leq D \cdot \max_{e \in E}\{U_e\}. \tag{27}$$

On the other hand, our algorithm returns an **I**-*factor* H such that $d_H(u_i) \geq D, \forall i \in \mathcal{N}$. Therefore, for every $u_i \in U$, $b(u_i) = \sum_{e \in \delta_H(u_i)} U_e \geq \min\{U_e | U_e > 0, e \in E\} \cdot d_H(u_i) \geq \min\{U_e | U_e > 0, e \in E\} \cdot D$. We conclude that our solution H has value at least $O(H) = \min_{i \in \mathcal{N}} b(u_i) \geq \min\{U_e | U_e > 0, e \in E\} \cdot D$. Combining with inequality (27), we conclude that $\frac{O(H)}{O(H^*)} \geq \beta(G, U)$. ∎

Corollary 4: If $\beta(G, U) = 1$, i.e., all nonzero U_{if} values are equal, then max-min **I**-*matching* is solvable in polynomial time.

Therefore, we have shown the following.

Theorem 3: $\beta(G, U)$ is a tight bound for the approximability of all the problems mentioned in this section, i.e., the max-min **I**-*factor*, MMFS, SANTA CLAUS, R-SANTA CLAUS, D2-SANTA CLAUS, and SD2-SANTA CLAUS problems.

We proceed with a few observations about special cases.

Lemma 7: For every $\epsilon > 0$, there is a $(2 + \epsilon)$-approximation algorithm for the special case of the MMFS problem with $T = 1, K_i = 0, a_i \geq F \ \forall i \in \mathcal{N}$ and each frequency $f \in \mathcal{F}$ has nonzero U_{if} value for at most two SUs $i \in \mathcal{N}$.

Proof: This special case corresponds to D2-SANTA CLAUS, for which there exists a $(2 + \epsilon)$-approximation algorithm [34]. Therefore, the lemma holds. ∎

Lemma 8: There exists a PTAS for the special case of the MMFS problem with $T = 1, K_i = 0, U_{if} = U_{jf}$, and $a_i \geq F, \forall i \neq j$ and $i, j \in \mathcal{N}$.

Proof: The frequencies and SUs can be regarded as jobs and machines, respectively, and this special case of the problem becomes the problem of maximizing the minimum machine completion time on identical machines [36], which is the dual problem to the well-known problem of makespan minimization. There exists a PTAS for the problem of maximizing the minimum machine completion time [36], and hence the lemma holds. ∎

3) Related Work on the SANTA CLAUS *Problem:* Bansal and Sviridenko have shown in [22] that the natural LP formulation of the SANTA CLAUS problem has a big integrality gap. Therefore, instead of the natural LP formulation, Bansal and Sviridenko have considered the so-called *configuration LP* and showed how to round it so that the resulting value is at least OPT/N, where N is the number of kids (SUs). They also showed that the integrality gap is in the order of $1/\sqrt{N}$. Asadpour and Saberi have shown in [23] that it is possible to round the configuration LP such that the objective function value is at least $OPT/\sqrt{N}\log^3 N$.

The work of Bansal and Sviridenko [22] proposes a method of rounding the configuration LP for the R-SANTA CLAUS problem such that a factor of at most $\log\log N/\log\log\log N$ is lost. Later, Asadpour *et al.* [37] have shown an integrality gap of 5 for the R-SANTA CLAUS problem, which was later improved to 4 by the same authors [38]. Note here that this is an *estimation ratio* rather than an approximation ratio. In other words, they have proved that the gap can be at most 4. However, they failed to provide a polynomial-time 4-approximation algorithm. They provided a local search heuristic that returns a solution of value at least $OPT/4$; nevertheless, their method is not known to run in polynomial time. In [34], a $(2 + \epsilon)$-approximation is given for D2-SANTA CLAUS, and it is shown that the lower bound of 2 holds even for its subproblem SD2-SANTA CLAUS.

Recall that the $\beta(G, U)$-approximation algorithm that we propose in this paper works for the max-min **I**-*factor* problem, which is a much more generalized version of the general (not restricted) SANTA CLAUS problem. Therefore, our algorithm also works for the general SANTA CLAUS problem, for which very few approximation results have been found [22], [23]. The additive approximation ratio of $\max_{if} U_{if}$ in [32] can be arbitrarily bad since it gives a very bad guarantee even when a single U_{if} value is large and the others are small. The $\sqrt{N}(log^3 N)$ approximation ratio of [23] can still be bad when the number of kids (SUs) is large. On the other hand, our $\beta(G, U)$-approximation algorithm gives a good approximation guarantee when the U_{if} values are close to each other; i.e., it works well in a fairly uniform spectral environment where the availabilities of all frequencies are similar for every SU. Even if the number SUs is very large, our algorithm gives a good approximation ratio as long as the nonzero U_{if} values are close to each other; i.e., $\beta(G, U)$ is small. In other words, it gives a better result than the one in [23] when there are many SUs but a uniform spectral environment. However, it fails to provide a good approximation guarantee for highly nonuniform spectral environments. On the other hand, when the number of kids (SUs) is small and the values of the presents to the kids (spectral environment) are highly nonuniform, the algorithm in [23] gives a better approximation ratio. To sum up, our algorithm is strong in terms of two different criteria: First, unlike [22] and [23], our algorithm works for a much more general case than the SANTA CLAUS problem. Second, it gives a better approximation guarantee than the previous

work when there are a large number of SUs and the spectral environment is fairly uniform. In other words, it provides an alternative solution so that whichever algorithm provides a better approximation guarantee (either ours or the ones in [22] and [23]) can be chosen in a practical implementation.

Authors in [32] have shown a method that gives an approximation guarantee of $\mathrm{OPT} - \max_{if} U_{if}$ for the general SANTA CLAUS problem. Although their method performs badly for high values of U_{if}, its performance guarantee is good for low values of U_{if}. Therefore, the approximation algorithm in [22] gives good results for the special case of the MMFS problem where U_{if} values are small, $T = 1, K_i = 0$, and $a_i \geq F\ \forall i \in \mathcal{N}$.

4) Special Cases:

Case 1: Assume that $a_i = 1\ \forall i \in \mathcal{N}$ and $T = 1$. In this case, the problem is equivalent to the max-min version of the *linear bottleneck assignment problem* (LBAP), where the workers (frequencies) are assigned to the workstations (SUs) such that the completion time of the job with the latest completion time is minimized. The authors in [39] develop threshold algorithms, a dual method, and a shortest augmenting path method for solving LBAP in polynomial time. The work in [31] also develops thresholding methods. In particular, min-max version of LBAP is equivalent to its max-min version [31], [39]. Note that having $K_i = 0\ \forall i \in \mathcal{N}$ is not necessary here. Even when $\exists i \in \mathcal{N}$ such that $K_i \neq 0$, thresholding method in [31] can still be applied. However, as in the proof of Lemma 6, we need a dummy vertex called $\overline{v}$ and dummy edges between each SU i and $\overline{v}$ with weight K_i, and then we need to implement the thresholding method in this new graph.

Case 2: Recall that if we decide to neglect the past performance, then $K_i = 0, \forall i \in \mathcal{N}$. In this case, constraint (17) can be eliminated. The reason for this is that the objective function already aims to assign every SU at least one time-slot because it tries to make the throughput of every SU as high as possible. If it is possible to assign each SU at least one time-slot, objective function will do it anyway. If it is not possible, then the only difference is that the case with constraint (17) declares that no feasible solution can be found, whereas the case without constraint (17) declares that a feasible solution has been found but the resulting objective function value is zero. Therefore, both cases are essentially the same, and hence constraint (17) can safely be eliminated in this special case. This elimination reduces the number of constraints in the problem formulation and hence enables more efficient running time if optimization software such as CPLEX [27] is used to find close to optimal solutions.

Remark 1: Assume that $a_i = 1, \forall i \in \mathcal{N}$ and ignore constraint (17). In this special case, the method of solving the problem for one time-slot and applying the same solution for all time-slots (akin to the method used in special case 3 of the TMS problem in Section III-B.2) does not work. To see this, consider the following example. There are two SUs and two frequencies. Let $U_{11} = U_{21} = 3$ and $U_{21} = U_{22} = 0$. If we solve the problem in one time-slot and apply the same solution to the other time-slots, then the result equals zero. However, we can achieve a nonzero result by assigning different frequencies to an SU in different time-slots. Assigning frequency 1 to SU 1 and frequency 2 to SU 2 in the first time-slot, and assigning frequency 1 to SU 2 and frequency 2 to SU 1 in the second time-slot achieves a nonzero throughput value for the minimum throughput.

D. Results About the PFS Problem

Theorem 4: The PFS problem remains NP-Hard in the strong sense even when $T = 1, \varphi = 1, a_i = 3$, and $U_{if} = U_{jf}\ \forall i, j \in \mathcal{N}$.

Proof: Recall that the 3-PARTITION problem is to decide whether a given set of integers can be partitioned into triplets that all have the same sum. Assume that there is a polynomial-time algorithm for the PFS problem. Then, we can use this algorithm to solve the 3-PARTITION problem as follows. Consider the special case of the PFS problem where there are m SUs, $3m$ frequencies, $T = 1$ time-slot, $U_{if} = U_{jf}\ \forall i, j \in \mathcal{N}$, and $a_i = 3\ \forall i \in \mathcal{N}$. Every 3-PARTITION corresponds to a feasible solution of this special case, and vice versa. Then, S_i equals the sum of all the U_{if} values assigned to this SU i. The values of each S_i are equal to each other if and only if the answer to the 3-PARTITION problem is *YES*. Notice that the product of a set of numbers is maximized when all numbers are equal to each other. In other words, if it is theoretically possible to make the sum of integers in each group equal to each other (if the answer to the 3-PARTITION problem is *YES*), then the polynomial-time algorithm for the PFS problem will yield a solution where the sum of integers in each group (the sum of U_{if} values assigned to each SU i) is equal to each other. If the answer is *NO*, then as a result of the PFS execution, the sum of integers in each group will not all be equal to each other; i.e., the sum in at least one group will differ from another sum (in another group). Since the 3-PARTITION problem is NP-Hard in the strong sense, PFS problem is also NP-Hard in the strong sense even when $T = 1, \varphi = 1, a_i = 3$, and $U_{if} = U_{jf}\ \forall i, j \in \mathcal{N}$. ■

*Max-Product Santa Claus Problem (*MAX-PROD-SANTA CLAUS*):* This problem is the same as the SANTA CLAUS problem except that the goal of Santa Claus is to maximize the product of the happiness values of the kids.

Lemma 9: If MAX-PROD-SANTA CLAUS cannot be approximated within any constant factor better than c, then PFS cannot be approximated within any additive constant less than $\log(c)$.

Proof: Consider a special case of the PFS problem where $T = 1$, $\varphi = 1$, and $a_i \geq F, \forall i \in \mathcal{N}$. Recall that maximizing the sum of logarithms of a set of numbers is equivalent to maximizing their product. Therefore, the optimization in this case corresponds to distributing the frequencies to the SUs in such a way that the product of the throughput values of each SU is as high as possible. Because $a_i \geq F\ \forall i \in \mathcal{N}$, there is practically no upper bound on the number of frequencies that an SU can receive. This special case is exactly the MAX-PROD-SANTA CLAUS problem where kids are replaced by SUs and presents are replaced by frequencies. Therefore, if MAX-PROD-SANTA CLAUS cannot be approximated within any constant factor better than c, then PFS cannot be approximated within any additive constant less than $\log(c)$. ■

Theorem 5: The MAX-PROD-SANTA CLAUS problem cannot be approximated within any constant factor better than $\frac{4}{3}$.

Proof: This result is implied by the work of Lenstra *et al.* [33], which proves that the problem of minimum

makespan scheduling in unrelated parallel machines cannot be approximated within any constant factor better than $3/2$ unless $P = NP$. In particular, the work in [33] proves that for the minimum makespan problem on unrelated parallel machines, the question of deciding if there exists a schedule with makespan at most 2 is NP-complete. Let us consider the MAX-PROD-SANTA CLAUS problem. If an assignment where each kid has a happiness value of 2 cannot exist, then at least one kid has to have a happiness value of 3 and another kid with happiness 1. This result implies that MAX-PROD-SANTA CLAUS cannot be approximated within any constant factor better than $\frac{4}{3}$. ■

Corollary 5: Due to Lemma 9 and Theorem 5, PFS problem cannot be approximated within any additive constant less than $\log(4/3)$ even when $T = 1, \varphi = 1$, and $a_i \geq F, \forall i \in \mathcal{N}$.

IV. PRACTICAL IMPLICATIONS

Our findings in this work indicate that the MMFS problem has high computational complexity even in very special cases. Therefore, this paper shows that providing throughput fairness to the SUs is a computationally challenging task for CBS operators. To better observe the conceptual computational difficulty of providing throughput fairness, consider the following scheduling problem:

$$\max \sum_{i=1}^{N} \sum_{f=1}^{F} U_{if} Y_{if} \tag{28}$$

s.t.

$$\sum_{f=1}^{F} U_{if} Y_{if} \geq \Omega, \forall i \in \mathcal{N} \tag{29}$$

$$(17), (13), (14), \text{and } (15) \tag{30}$$

where Ω is a prespecified throughput value. In this problem, the goal is to maximize the total throughput such that each SU is guaranteed a prespecified throughput value of Ω. If we had a polynomial-time solution to this problem, we would be able to use this solution iteratively by updating the value of Ω in each iteration and therefore find the maximum value of Ω for which the above problem has a feasible solution; in other words, we would be able to solve the MMFS problem in polynomial time. Hence, the problem in (28)–(30) is also NP-Hard in the strong sense. That is to say, even checking whether each SU can be guaranteed a certain throughput value is a computationally hard problem.

In contrast, the TMS problem, which maximizes total throughput while at the same time providing temporal fairness, is solvable in polynomial time. Taking into account the fact that scheduling decisions have to be made in real time, a CBS operator may opt to provide temporal fairness (by executing the TMS formulation) instead of throughput fairness. If the CBS operator prefers to use the MMFS formulation in spite of its computational difficulties, it may check whether the special cases we have outlined in this paper are valid in that particular scheduling period and use the relatively efficient methods we discussed. For instance, we have shown in this paper that the special case of Case 1 in Section III-C.4 is solvable in polynomial time. Otherwise, the CBS operator can check the value $\beta(G, U)$ in that particular scheduling period. If $\beta(G, U)$ is low enough; i.e., if the spectral conditions are fairly uniform, the CBS operator can use our $\beta(G, U)$-approximation algorithm, with a performance guarantee of $\frac{OPT}{\beta(G,U)}$ even when there are a large number of SUs and frequencies.

V. NUMERICAL EVALUATION

We have evaluated the average case behavior of our $\beta(G, U)$-approximation algorithm (MAXMINEQ) for the MMFS problem through simulations. As in [5], we have simulated a centralized CRN cell with 600 m of radius and noise variance of $\sigma^2 = 10^{-6}$. The maximum tolerable interference power of the PUs for each frequency is $P_{IF_{\max}} = 10$ mW. Each scheduling period consists of $T = 10$ time-slots, and each time-slot has a duration of $T_s = 100$ ms. 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. Additive white Gaussian noise (AWGN) channels are assumed, and both the PUs and the SUs move within the CRN cell according to the random waypoint mobility model (RWMM) with 10 s of staying duration between the movement periods. RWMM is a very commonly used model for performance analysis of wireless networks [40]. The Markovian model described in [5] is used to model the PU spectrum utilization. Markovian models are commonly used for the spectrum occupancy modeling of cognitive radio networks [41], [42].

As in [5], we assume that all the PUs and SUs move with constant velocities of V_p and V_s, respectively. We denote the number of SUs and PUs in the cell by N and M, respectively. We take $M = 20, F = 15$, $V_p = V_s = 13$ m/s, and $a_i = 3$ $\forall i \in \mathcal{N}$. We compare the average minimum throughput performance of MAXMINEQ to the results obtained from the optimization software CPLEX [27]. In [5], a statistical method is used to calculate the number of samples to take (the number of scheduling periods to run the simulations for) so that the sample mean of all the samples are ± 0.5 of the actual mean with a 95% confidence level. We use the same statistical method to determine the number of scheduling periods for each CPLEX experiment. The number of samples we take for MAXMINEQ in all the experiments is the same as the corresponding ones obtained via CPLEX.

Fig. 1 shows the average minimum throughput of CPLEX and MAXMINEQ where the number of SUs varies between 5 and 30. As the number of SUs increases, the minimum throughput value resulting from both algorithms decreases. This behavior is natural since the resources assigned per user decrease when more users share the same amount of resources. The table in Fig. 2 provides the numerical minimum throughput values obtained from CPLEX and MAXMINEQ, which are essentially the same as the values shown in Fig. 1. The minimum throughput performance of MAXMINEQ is close to the one of CPLEX. While obtaining the CPLEX values, we have experimentally determined the appropriate gap value in order to obtain the results in a reasonable amount of time. The default gap value used by CPLEX is 0.01%. As the number of SUs increases, the simulations take much longer time. Therefore,

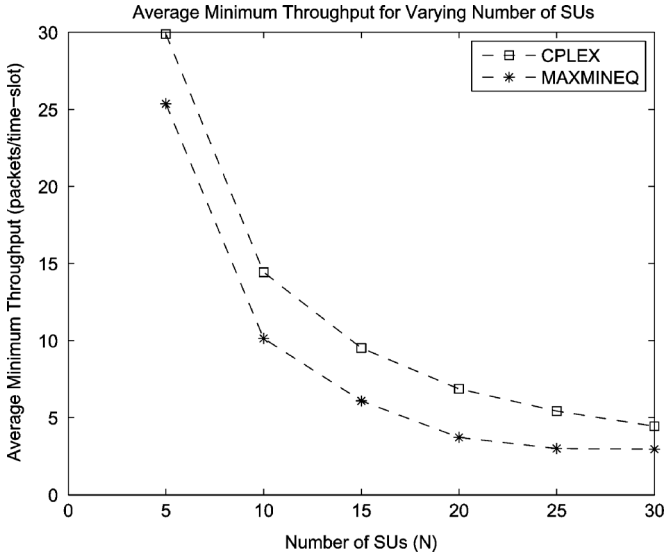

Fig. 1. Average minimum throughput and CPLEX gap values for varying number of SUs (N).

Number of SUs (N)	5	10	15	20	25	30
MAXMINEQ result	25.36	10.14	6.09	3.71	2.99	2.95
CPLEX result	29.89	14.42	9.52	6.86	5.43	4.44
CPLEX gap	0.01% (default)	0.58%	0.60%	1.5%	1.5%	1.5%
$\beta(G,U)$	2.55	4.06	4.3	4.92	5.42	6.62

Fig. 2. Average minimum throughput, CPLEX gap, and $\beta(G,U)$ values.

we have increased the value of the gap parameter (*epgap*) to obtain satisfactory results in a reasonable amount of time. The gap values for each experiment are shown in the table in Fig. 2. The table also shows the average $\beta(G,U)$ values resulting from each experiment. As the number of SUs increases, the average $\beta(G,U)$ values also increase due to the increasing diversity in the network; i.e., the probability that there exists an SU with better/worse U_{if} values increases as the number of SUs increases. However, the experimental results show that the $\beta(G,U)$ values do not increase too much in practice. Furthermore, recall that $\beta(G,U)$ is the worst-case bound of the algorithm MAXMINEQ. The experimental results show that the average-case behavior of MAXMINEQ in practice is much better than its worst-case guarantee. Although the $\beta(G,U)$ value increases as the number of SUs increases, we do not observe an increase in the average performance difference of MAXMINEQ and CPLEX; in other words, the average-case performance of MAXMINEQ does not deteriorate.

VI. Conclusion

We have presented a graph-theoretic approach to throughput-maximizing, max-min fair, and proportionally fair scheduling problems for centralized cognitive radio networks, which have previously been formulated in the literature. We have proposed a polynomial-time algorithm for the throughput-maximizing scheduling problem and discussed some of its special cases. We have then proved that the max-min fair scheduling problem is NP-Hard in the strong sense and inapproximable within any constant factor less than 2 unless $P = NP$. We have also presented an approximation algorithm for this problem with approximation ratio depending on the maximum possible data rates of the secondary users. We have evaluated the average-case behavior of our approximation algorithm and demonstrated that it provides reasonable average case minimum throughput performance. Moreover, we have discussed some special cases of the MMFS problem and elaborated on their combinatorial properties. Then, we have proved that the PFS problem is also NP-Hard in the strong sense and inapproximable within any additive constant less than $\log(4/3)$. Furthermore, we have proposed more efficient integer programming formulations for all the three problems.

Our study indicates that MMFS is computationally very hard. This problem cannot be approximated within any constant factor less than 2 even in very special cases. Moreover, the theoretical computer science community has still been unable to find efficient approximation algorithms for the problem. The computational complexity of this problem together with its practical importance in cognitive radio networks call for heuristic techniques that provide efficient suboptimal solutions. On the other hand, as the objective function of PFS behaves more smoothly than the objective function of MMFS, we were unable to show the same hardness results by applying similar techniques. We conjecture that MMFS is at least as hard to approximate as PFS. To settle this conjecture is an open problem. It is not clear whether MMFS is strictly harder to approximate than PFS. To answer these questions, to design approximation algorithms for PFS, and to design efficient heuristics for both MMFS and PFS problems is subject of future work.

References

[1] F. No, "03-222 notice of proposed rule making and order," 2003.
[2] X. Liu, E. K. P. Chong, and N. B. Shroff, "Opportunistic transmission scheduling with resource-sharing constraints in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 10, pp. 2053–2064, Oct. 2001.
[3] P. Viswanath, D. N. C. Tse, and R. Laroia, "Opportunistic beamforming using dumb antennas," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1277–1294, Jun. 2002.
[4] X. Liu, E. K. Chong, and N. B. Shroff, "A framework for opportunistic scheduling in wireless networks," *Comput. Netw.*, vol. 41, no. 4, pp. 451–474, 2003.
[5] D. Gözüpek and F. Alagöz, "An interference aware throughput maximizing scheduler for centralized cognitive radio networks," in *Proc. IEEE PIMRC*, 2010, pp. 1527–1532.
[6] D. Gözüpek and F. Alagöz, "A fair scheduling model for centralized cognitive radio networks," Tech. Rep., 2013 [Online]. Available: http://arxiv.org/pdf/1309.2233v1.pdf
[7] L. Li, M. Pal, and Y. R. Yang, "Proportional fairness in multi-rate wireless LANs," in *Proc. IEEE INFOCOM*, 2008, pp. 1004–1012.
[8] T. Nandagopal, T.-E. Kim, X. Gao, and V. Bharghavan, "Achieving MAC layer fairness in wireless packet networks," in *Proc. ACM MobiCom*, 2000, pp. 87–98.
[9] D. Loguinov, J. Casas, and X. Wang, "Graph-theoretic analysis of structured peer-to-peer systems: Routing distances and fault resilience," *IEEE/ACM Trans. Netw.*, vol. 13, no. 5, pp. 1107–1120, Oct. 2005.
[10] M. Shalom and S. Zaks, "A $10/7 + \varepsilon$ approximation for minimizing the number of ADMs in SONET rings," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1593–1602, Dec. 2007.
[11] D. Blough, G. Resta, and P. Santi, "Approximation algorithms for wireless link scheduling with SINR-based interference," *IEEE/ACM Trans. Netw.*, vol. 18, no. 6, pp. 1701–1712, Dec. 2010.

[12] P. Djukic and S. Valaee, "Delay aware link scheduling for multi-hop TDMA wireless networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 3, pp. 870–883, Jun. 2009.
[13] W. Wang and X. Liu, "List-coloring based channel allocation for open-spectrum wireless networks," in *Proc. 62nd IEEE Veh. Technol. Conf.*, 2005, pp. 690–694.
[14] F. Khozeimeh and S. Haykin, "Dynamic spectrum management for cognitive radio: an overview," *Wireless Commun. Mobile Comput.*, vol. 9, no. 11, pp. 1447–1459, 2009.
[15] 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, 2006.
[16] Y. Yuan, P. Bahl, R. Chandra, T. Moscibroda, and Y. Wu, "Allocating dynamic time-spectrum blocks in cognitive radio networks," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2007, pp. 130–139.
[17] E. Driouch and W. Ajib, "On the user scheduling in cognitive radio mimo networks," in *Proc. IEEE GLOBECOM*, 2012, pp. 1242–1247.
[18] L. Xie and X. Jia, "Qos multicast routing and transmission scheduling in multi-hop cognitive radio networks," in *Proc. IEEE GLOBECOM Workshops*, 2010, pp. 1487–1491.
[19] L. Lovász and M. Plummer, *Matching Theory*. Providence, RI, USA: AMS Chelsea, 2009.
[20] R. Pulleyblank, "Faces of matching polyhedra," Ph.D. dissertation, Dept. Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON, Canada, 1973.
[21] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*. New York, NY, USA: Springer, 2003.
[22] N. Bansal and M. Sviridenko, "The Santa Claus problem," in *Proc. ACM STOC*, 2006, p. 40.
[23] A. Asadpour and A. Saberi, "An approximation algorithm for max-min fair allocation of indivisible goods," in *Proc. ACM STOC*, 2007, pp. 114–121.
[24] D. König, "Graphok es alkalmazasuk a determinansok es a halmazok elmeletere," (in Hungarian) *Math. Termeszettudomanyi Ertesito*, vol. 34, pp. 104–119, 1916.
[25] P. Hall, "On representatives of subsets," *J. London Math. Soc.*, vol. 10, pp. 26–30, 1934.
[26] W. Tutte, "The factorization of linear graphs," *J. London Math. Soc.*, vol. 22, pp. 107–111, 1947.
[27] IBM, Armonk, NY, USA, "CPLEX," [Online]. Available: http://www.ilog.com/products/cplex
[28] C. Chekuri and S. Khanna, "A polynomial time approximation scheme for the multiple knapsack problem," *SIAM J. Comput.*, vol. 35, no. 3, pp. 713–728, 2005.
[29] S. Martello and P. Toth, *Knapsack Problems Algorithms and Computer Implementations*. New York, NY, USA: Wiley, 1990.
[30] A. Kershenbaum, *Telecommunications Network Design Algorithms*. New York, NY, USA: McGraw-Hill, 1993.
[31] E. Lawler, *Combinatorial Optimization: Networks and Matroids*. New York, NY, USA: Dover, 2001.
[32] I. Bezáková and V. Dani, "Nobody left behind: Fair allocation of indivisible goods," *ACM SIGecom Exchanges*, vol. 5, pp. 1–10, 2005.
[33] J. Lenstra, D. Shmoys, and E. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Math. Program.*, vol. 46, no. 1, pp. 259–271, 1990.
[34] D. Chakrabarty, J. Chuzhoy, and S. Khanna, "On allocating goods to maximize fairness," in *Proc. 50th Annu. IEEE FOCS*, 2009, pp. 107–116.
[35] M. Bateni, M. Charikar, and V. Guruswami, "MaxMin allocation via degree lower-bounded arborescences," in *Proc. ACM STOC*, 2009, pp. 543–552.
[36] G. Woeginger, "A polynomial-time approximation scheme for maximizing the minimum machine completion time," *Oper. Res. Lett.*, vol. 20, no. 4, pp. 149–154, 1997.
[37] A. Asadpour, U. Feige, and A. Saberi, "Santa Claus meets hypergraph matchings," in *Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques*. Berlin, Germany: Springer, 2008, pp. 10–20.
[38] A. Asadpour, U. Feige, and A. Saberi, "Santa Claus meets hypergraph matchings," *Trans. Algor.*, vol. 8, no. 3, p. 24, 2012.
[39] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. Philadelphia, PA, USA: SIAM, 2009.
[40] D. Alparslan and K. Sohraby, "Two-dimensional modeling and analysis of generalized random mobility models for wireless ad hoc networks," *IEEE/ACM Trans. Netw.*, vol. 15, no. 3, pp. 616–629, Jun. 2007.
[41] H. Lee and D. Cho, "VoIP capacity analysis in cognitive radio system," *IEEE Commun. Lett.*, vol. 13, no. 6, pp. 393–395, Jun. 2009.
[42] C. Li and M. Neely, "Exploiting channel memory for multiuser wireless scheduling without channel measurement: Capacity regions and algorithms," *Perform. Eval.*, vol. 68, no. 8, pp. 631–657, 2011.

Didem Gözüpek received the B.S. degree (high honors) in telecommunications engineering from Sabancı University, Istanbul, Turkey, in 2004, the M.S. degree in electrical engineering from the New Jersey Institute of Technology (NJIT), Newark, NJ, USA, in 2005, and the Ph.D. degree in computer engineering from Bogazici University, Istanbul, Turkey, in 2012.

She is an Assistant Professor with the Computer Engineering Department, Gebze Institute of Technology, Kocaeli, Turkey. From 2005 to 2008, she worked as an R&D Engineer in a telecommunications company in Istanbul. Her main research interests are scheduling and resource allocation in communication networks, algorithmic graph theory, and approximation algorithms.

Dr. Gözüpek received the CAREER Award from the Scientific and Technological Research Council of Turkey (TUBITAK) in 2014, the Dr. Serhat Özyar Young Scientist of the Year Honorary Award in 2013, and the Bogazici University Ph.D. Thesis Award in 2012. She was a finalist for the Google Anita Borg Memorial Scholarship in 2009.

Mordechai Shalom received the B.A. degree in architecture from Istanbul Technical University (ITU), Istanbul, Turkey, in 1981, and the M.Sc. and Ph.D. degrees in computer science from the Technion—Israel Institute of Technology, Haifa, Israel, in 1986 and 2006, respectively.

He currently serves as a Senior Lecturer with the Tel Hai Academic College, Upper Galilee, Israel, and as an Adjunct Lecturer with the Technion. His research interests are optimization problems in optical networks, distributed algorithms, approximation algorithms, and online algorithms.

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

He is a Professor with the Department of Computer Engineering, Bogazici University, Istanbul, Turkey. He has contributed to many research projects for various agencies/organizations, including the US Army Intelligence Center, Naval Research Laboratory, UAE Research Fund, Turkish Scientific Research Council, State Planning Organization of Turkey, etc. He is the Satellite Systems Advisor to the Kandilli Earthquake Research Institute, Istanbul, Turkey. He is an editor of five books and an author of more than 100 scholarly papers in selected journals and conferences. His research interests span various aspects of wireless/mobile/satellite communication networks.

Dr. Alagöz has served on several major conference technical committees and organized and chaired many technical sessions at many international conferences. He is a member of the IEEE Satellite and Space Communications Technical Committee. He has received numerous professional awards.