Uplink Scheduling in Wireless Networks with Successive Interference Cancellation

Majid Ghaderi, *Member, IEEE,* and Mohsen Mollanoori, *Student Member, IEEE,*

Abstract—In this paper, we study the problem of uplink scheduling in wireless networks with successive interference cancellation (SIC). With SIC, concurrent transmissions, if properly scheduled, can be successfully decoded at a receiver. The scheduler decides: **i.** in which time-slot to schedule, and **ii.** in what order in a time-slot to decode each transmission in order to maximize the system utility and/or satisfy a system constraint. These two scheduling decisions effectively determine the rates allocated to concurrent transmissions, which in turn determine the throughput and fairness of the system. We consider several different scheduling problems in this context. The objective of the problems is to either maximize the throughput of the system or to obtain some kind of fairness among the users. We formulate and study each problem from the perspective of computational complexity. For each problem, we either propose a polynomial time algorithm, if any exists, or show that the problem is NP-hard.

Index Terms—Successive Interference Cancellation, Scheduling in Wireless Networks, Throughput and Fairness Maximization.

1 Introduction

Interference and noise are both limiting factors in wireless networks. However, their effects on the performance of wireless networks are not identical. In a multi-user wireless network, increasing the transmission power can reduce the noise effect. Nevertheless, an increase in the transmission power, if not carefully controlled, may even worsen the interference effect. In addition, interference is a structured signal since it is caused by other transmissions in the network, whereas noise is structureless.

Successive interference cancellation (SIC) is a multi-user detection technique that uses the structured nature of interference to decode multiple concurrent transmissions. Assume a composite signal $S = S_1 + \cdots + S_n + Z$ of n overlapping signals $S_1, \ldots, S_n$ plus the noise signal Z is received at a receiver. Employing SIC, one of the signals, say S_i, is decoded first while the rest of the signals are considered as noise. After S_i is decoded, the decoder reconstructs the corresponding analog signal and subtracts it from the original composite signal S. At this stage, the remaining signal is free from the interference of signal S_i (assuming error-free decoding and perfect removal of the signal S_i). The same technique is applied repeatedly to decode the remaining $n-1$ signals. It is worth noting that if the decoder fails to decode one of the signals, it is unlikely that it can decode the subsequent signals. Since at every stage, the remaining signals are treated as noise, the maximum rate achievable by a user depends not only on its corresponding received signal power but also on the *order* at which its signal is decoded.

While SIC does not enforce a specific decoding order, because of practical and theoretical reasons, it is common to decode the signals from the strongest to the weakest one (we discuss the issue in Sections 6 and 7). Since the amount of interference is reduced in every stage (because of subtraction of the decoded signal), late decoding of the weaker signals increases the corresponding SINR (Signal to Interference plus Noise Ratio) and consequently increases the probability of a successful decode. Later, we show that while the above order of decoding does not decrease the sum capacity of the system, it actually results in a proportionally fair rate assignment among users.

With conventional decoders, transmission *scheduling* mechanisms are commonly used to avoid multiple simultaneous transmissions because conventional receivers treated interference as random noise. However, to increase the throughput with SIC, the scheduling mechanism should allow multiple concurrent transmissions in the network while controlling the *rate* and *decoding order* of the transmissions so that the composite signal can be decoded at the receiver [1]. SIC receivers are architecturally similar to traditional non-SIC receivers in terms of hardware complexity and cost [2] as they use the same decoder to decode the composite received signal at different stages. As a result, neither a complicated decoder nor multiple antennas are required to increase the throughput of the system [1], [3]. It also makes SIC more practical than other multi-user detection techniques such as joint detection [4]. Furthermore, it is known that other multiple access techniques such as CDMA and OFDMA are no more efficient than SIC [5, Ch. 6]. As such, SIC has been recently considered in commercial wireless systems as a way to increase system throughput [6].

SIC can be employed for uplink [7] as well as down-

- *M. Ghaderi and M. Mollanoori are equal contributing authors.*
- *M. Ghaderi is with the Department of Computer Science, University of Calgary. E-mail: mghaderi@ucalgary.ca.*
- *M. Mollanoori is with the Department of Computer Science, University of Calgary. E-mail: mmollano@ucalgary.ca.*

link [8] transmissions. However, SIC can achieve a higher throughput gain on the uplink of a wireless network since the total received power is higher due to concurrent transmissions from multiple users. In Section 3, we further discuss the relation between the total received power and capacity of the system.

In this paper, we consider several uplink scheduling problems, assuming that SIC is implemented at the physical layer of the network. To the best of our knowledge, none of the problems considered in this paper and summarized as follows is studied by others in the context of SIC.

1) **Maximum Throughput Scheduling (**MTS**):** The objective of this problem is to schedule a set of users at a given number of time slots such that the throughput of the system is maximized. The throughput of the system is given by the summation of the throughput of individual users. We show that MTS is NP-hard.
2) **Constrained Maximum Throughput Scheduling (**$\mathrm{MTS-C}$**):** This problem is the same as MTS except that at most l users are scheduled at each time slot. We show that for $l = 2$, which is of practical interest, the problem is solvable in polynomial time, while for any $l \geq 3$ the problem is NP-hard.
3) **Proportional Fair Scheduling (**PFS**):** The objective of this problem is to achieve proportional fairness among users [9]. Proportional fairness among users is achieved by maximizing the summation of the logarithm of individual users' throughputs. Hence, the only difference between PFS and MTS scheduling problems is the additional $\log(.)$ function in the objective function of PFS. We show PFS can be solved in polynomial time.
4) **Scheduling with Minimum SINR (**$\mathrm{MinSINR}$**):** This problem requires that a minimum SINR for each user is provided by the scheduling algorithm. Having a minimum SINR for each user means a minimum rate guarantee for each user, which is a required property for some applications such as voice or video transmission. We show this problem is solvable in polynomial time.

In addition, we discuss variations of the above problems as corollaries, wherever possible. The approach we take in this paper is a combinatorial and complexity theoretic one. We aim to find the theoretical limits of computation of the optimal solutions to the scheduling problems with SIC. To do so, for each of the problems, we prove whether the problem can be solved in polynomial time or it does not have any polynomial time solution, unless $\mathrm{P} = \mathrm{NP}$.

From the theoretical point of view, NP-hardness of a problem means that there is no polynomial time algorithm to compute the optimal schedule (unless, $\mathrm{P} = \mathrm{NP}$). From the practical point of view, it means that there is no optimal real-time scheduler that maximizes the system utility, except for very small system sizes. Note that in real world wireless systems, scheduling is performed every few milliseconds. This makes a brute-force search to find the optimal schedule impractical (even for a few tens of users, the size of the search space becomes too large).

The rest of the paper is organized as follows: In the next section, we describe the related work. In Section 3 we describe the system model and assumptions. Problems MTS, $\mathrm{MTS-C}$, PFS, and $\mathrm{MinSINR}$ are studied in Sections 4 to 7. Section 8 presents the simulation results. Section 9 concludes the paper.

2 Related Work

Although scheduling is extensively studied in traditional wireless networks (*e.g.,* [10] and [11]), only a few works have explicitly considered interference cancellation.

There have been a number of studies in the area of SIC-aware MAC protocols. It has been shown that SIC can achieve near Shannon capacity in cellular networks in which communications are closed-loop and accurate channel estimations exist (see [2] and references therein). It is also known that SIC can improve the throughput of wireless LANs. For instance, Halperin *et al.* [12] experimentally studied the effect of SIC in unmanaged wireless networks with carrier sensing. They concluded that SIC can effectively improve the bandwidth utilization in wireless networks. In contrast, Sen *et al.* [13] reported that, without a proper scheduling mechanism, SIC may not be a promising approach to improve the throughput of wireless networks.

In [14], Gollakota *et al.* proposed a combination of interference alignment and cancellation as an effective technique to overcome the limitation of the number of antennas for improving the throughput of MIMO wireless LANs. They showed that the combined technique can be applied to scenarios where neither interference alignment nor cancellation applies alone. In a recent work, Wu *et al.* [15] utilized interference cancellation at the receiver to create a side channel that can be used to implement a multi-user scheduling algorithm without requiring any out-of-band signalling. The idea is to use the slack interference tolerance of the receiving hardware to allow simultaneous transmission of actual user data and control information. We note that interference cancellation at the physical layer is orthogonal to coding and transmission techniques that try to decode simultaneous transmissions at higher layers of the protocol stack such as ZigZag decoding [16] and Remap [17].

Yu and Giannakis proposed an algorithm called SICTA which uses SIC in a tree algorithm [18]. SICTA retains the received signal vector resulting from a collision in a time slot. The received signal vector of the subsequently decoded packets is used to cancel the interference in the collided signal and decode it. In [19], Mitran *et al.* consider a multi-rate and multi-power wireless multihop network with SIC capability. They assume that the receiver is capable of decoding up to n signals.

They formulate a joint routing and scheduling with SIC using a large linear program and compute the maximum achievable throughput of the network by solving the formulation numerically in a centralized manner. There is also a large body of research on scheduling links in wireless multihop networks with a minimum frame length. For instance, Goussevskaia *et al.* [20] show that two problems, namely scheduling and one-shot scheduling, under the geometric SINR model are NP-complete. In the first problem given a set of links, it is asked to schedule the links in the minimum number of time slots possible. The second problem, receives a weighted set of links and finds a subset of the links that can be scheduled in a single time slot so that the scheduled links have the maximum sum of weights. See [20] and references therein for more details on this topic.

In the broader context of multi-packet reception, there are several works on scheduling and MAC protocols [21]–[23], yet none of them is applicable to SIC-enabled networks. As an example, Zhao and Tong [21], [24] developed a MAC protocol with multi-packet reception capability considering a *reception matrix* as the underlying model. A reception matrix specifies the probability of k successful receptions when there are n concurrent transmissions in the network. Hence, the model does not differentiate between the concurrent transmissions (*e.g.,* all the transmissions are supposed to have the same rate). While the reception matrix model is a clever abstraction of a general multi-packet reception network, a more sophisticated model is required to accurately model the underlying dynamics of a SIC-based network such as the selection of transmission rates or the order of decoding (see Section 3). Other works on multi-packet reception, such as [22] and [23], consider a simpler model in which a receiver is capable of receiving up to k simultaneous packets, regardless of the transmission rates and channel conditions of users.

The closest work to ours is by Kumaran and Qian [25], in which the authors consider the uplink scheduling problem with SIC. Nevertheless, their work differs in that they only consider the case of scheduling multiple transmissions in a *single* time-slot in order to maximize the network throughput. In this work, we consider scheduling multiple users in multiple time-slots in order to optimize an objective function and/or satisfy a system constraint.

3 System Model and Assumptions

We consider a network consisting of a set of wireless users communicating with a single receiver (*e.g.,* an access point in a wireless LAN or a base station in a cellular network). Time is divided into *scheduling frames*, where each scheduling frame is divided into k time slots. Scheduling is done once every scheduling frame, at the beginning of the frame (this is similar to the scheduling structure of WiMAX networks [26]).

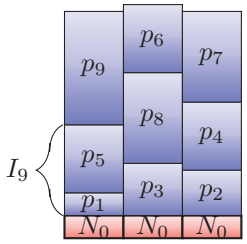


Fig. 1. Logical representation of the scheduling frame with 3 time slots and 9 powers. N_0 is the noise power. I_i specifies the sum of interference powers from other users, *e.g.*, in the above figure $I_9 = p_1 + p_5$. Note that I_i is a function of the location of u_i's signal in the scheduling frame, *i.e.*, relocation may change I_i. Figure is drawn such that in each time slot p_i's are decoded from top to bottom. Moreover, the height of the rectangles represent the signal powers.

In every scheduling frame, a set of n users denoted by $\mathcal{N} = \{u_1, u_2, \ldots, u_n\}$ is scheduled for uplink transmission. We assume that the users report their channel state information at the start of every scheduling frame and that channel fluctuations during a frame are negligible. Without loss of generality, we ignore the power control and assume that the users transmit at full power so that the scheduler is able to estimate the received power from each user. Let p_i denote the received power of the user u_i at the receiver and r_i denote the throughput of the user u_i. Given the set of received powers $p_1, p_2, \ldots, p_n$, the aim of the scheduler is to schedule the set of users $\mathcal{N}$ in the k time slots so that a *system utility function* is maximized and/or a system constraint is satisfied.

Let SINR_i be the signal to noise plus interference ratio of user u_i, *i.e.*, $\mathrm{SINR}_i = \frac{p_i}{N_0 + I_i}$, where I_i is the interference power that affects the signal of user u_i. To model the throughput of the users, we use the Shannon capacity formula as follows,

$$r_i = \log\left(1 + \mathrm{SINR}_i\right) . \tag{1}$$

Figure 1 shows the logical representation of a scheduling frame. The figure is drawn such that signals are decoded from top to bottom. For instance, in Figure 1, the signal of user u_9 is decoded just before the signal of user u_5. Therefore, the amount of interference affecting a user's signal is the sum of the received powers scheduled below that user in the same time slot.

The following well-known identity is crucial for many of the proofs throughout the paper and is formulated here for the ease of reference: The sum of the logarithms is the logarithm of the product of the terms. More formally, let $x_i > 0$ for $i = 1, \ldots, n$. We have,

$$\log(x_1 \times \cdots \times x_n) = \log(x_1) + \cdots + \log(x_n) . \tag{2}$$

We use the notation $\{\langle a_1, a_2, \ldots, a_{k_1}\rangle, \langle b_1, b_2, \ldots, b_{k_2}\rangle, \ldots\}$ to show the logical representation of the frame. That is, in one time slot a_1 is decoded first, a_2 is decoded

next, and so on until a_{k_1}. In the next time slot b_1 is decoded first, b_2 is decoded next, and so on. For instance, the frame of Figure 1 is shown as: $\{\langle p_9, p_5, p_1\rangle, \langle p_6, p_8, p_3\rangle, \langle p_7, p_4, p_2\rangle\}$. Note that, $\{\ \dots\ \}$ shows an orderless list, while $\langle\ \dots\ \rangle$ represents an ordered list. That is, the order of the time slots in a scheduling frame is not important while the order of decoding of the signals is important for us.

By this model, I_i and consequently the throughput of each user depends on the time slot in which the user is scheduled as well as the order at which the signals are decoded. However, the sum capacity of the users scheduled at a time slot is independent of the order of decoding and depends solely on the sum of the received powers $\sum_i p_i$ and the noise power N_0. Lemma 1 states this property formally.

Lemma 1. *Let $p_1, \dots, p_m$ be the received powers of the users scheduled in a single time slot. The following equality holds,*

$$\sum_{i=1}^{m} \log\left(1 + \frac{p_i}{N_0 + I_i}\right) = \log\left(1 + \frac{\sum_{i=1}^{m} p_i}{N_0}\right). \tag{3}$$

Assume without loss of generality that the decoding order is $\langle p_m, \dots, p_1\rangle$. Proof of Lemma 1 is simply by expanding the left hand side of (3) as,

$$\log(\frac{N_0 + \sum_{i=1}^{1} p_i}{N_0}) + \log(\frac{N_0 + \sum_{i=1}^{2} p_i}{N_0 + \sum_{i=1}^{1} p_i}) + \dots + \log(\frac{N_0 + \sum_{i=1}^{m-1} p_i}{N_0 + \sum_{i=1}^{m-2} p_i}) + \log(\frac{N_0 + \sum_{i=1}^{m} p_i}{N_0 + \sum_{i=1}^{m-1} p_i}), \tag{4}$$

and using (2) to simplify the expression. It is worth pointing out that Lemma 1 states two important properties of the sum capacity:

1) Sum capacity is independent of the decoding order of the signals. That is because $\sum_{i=1}^{m} p_i$ is a constant regardless of the order of decoding.
2) Increase in the sum capacity is proportional to the logarithm of the sum of the received powers.

Theoretically, the second property means: if m goes to infinity, sum capacity goes to infinity as well. Therefore, to maximize the system throughput, all the users have to be scheduled in every time slot if there is no constraint on the number of users that can be decoded using SIC [5], [25]. However, in practice, because of a number of reasons, only a few users can be scheduled at each time slot. First, since the increase in the sum capacity is logarithmic in the sum of the received powers, having a few concurrent transmissions, significantly increases the sum capacity. However, adding more users to the concurrent transmissions would be less effective as the number of users is increased. Second, due to the practical limitations, only a few overlapping signals can be decoded successfully [1]. One reason is that the decoding time in SIC increases linearly with the number of overlapping signals [5, Ch. 6]. Having many users, the linear decoding time causes practical difficulties. Another reason is that, in practice, the decoded signals cannot be perfectly removed from the composite signal. Thus, some residual interference remains after each decoding stage, and propagates during the subsequent decoding stages [6]. The accumulated residual interference results in a decoding error after a few users are decoded, limiting the number of simultaneous transmissions in a time slot.

To take into consideration the practical limitations, we consider two variations of the MTS problem (namely, MTS−C and MinSINR). In MTS−C, we assume that at most l users can be decoded in each time slot. In MinSINR, we assume that the decoder requires a minimum SINR denoted by $\mathcal{S}_{min}$ to successfully decode each signal.

We assume that the number of users is greater than the number of time slots, *i.e.*, $|\mathcal{N}| = n > k$, since the solution for the case of $n \leq k$, for all of the problems, is trivial. We assume that each node is scheduled exactly once in a scheduling frame, thus more than one node might be scheduled in a time slot (recall that scheduling simultaneous transmissions is allowed with SIC). While this restriction can be relaxed, its consideration here imposes some implicit fairness among the users. To relax the restriction, one may first schedule each of the nodes only once. Then, based on some criteria, select a subset of the nodes and add them to the scheduling frame such that none of the nodes are scheduled more than once in a time slot. In this manner, the *exactly once* constraint is relaxed to *at least once* constraint.

4 Maximum Throughput Scheduling

The objective of the maximum throughput scheduling problem is to maximize the sum of the throughput of the users in a scheduling frame. In the following we formally state the problem.

Problem 1 (Maximum Throughput Scheduling **(**MTS**)).** *Given a multiset of received powers $p_1 \leq p_2 \leq \dots \leq p_n$, and some noise power N_0, schedule the corresponding n users in k time slots so that the following objective function is maximized,*

$$\sum_{i=1}^{n} \log\left(1 + \frac{p_i}{N_0 + I_i}\right), \tag{5}$$

where I_i is the amount of interference power that affects the decoding of user u_i's signal (see Figure 1).

Theorem 1. MTS *is NP-hard.*

Proof: The proof is by reduction from the partition problem. In the partition problem, given a set S of integers, we are asked to find a subset $S' \subseteq S$ such that,

$$\Big|\sum_{x\in S'} x - \sum_{y\in(S-S')} y\Big|, \tag{6}$$

is minimized. The Partition problem is known to be NP-complete [27]. We show that the partition problem can be reduced (in polynomial time) to a special case of MTS, for $k = 2$. Since a special case of the problem is NP-hard, the general case is NP-hard as well.

Let π_1 and π_2 denote the partition of the received powers for the two time slots, *i.e.,* the users corresponding to π_1 will be scheduled in time slot 1 and the users corresponding to π_2 will be scheduled in time slot 2. We can rewrite the objective function (5) as follows,

$$\log\left(\left(1+\frac{\sum_{p\in\boldsymbol{\pi}_1} p}{N_0}\right)\left(1+\frac{\sum_{p\in\boldsymbol{\pi}_2} p}{N_0}\right)\right). \tag{7}$$

Let $X = \left(1+\frac{\sum_{p\in\boldsymbol{\pi}_1} p}{N_0}\right)$ and $Y = \left(1+\frac{\sum_{p\in\boldsymbol{\pi}_2} p}{N_0}\right)$. We know that,

$$X + Y = 2 + \frac{p_1 + \ldots + p_n}{N_0}, \tag{8}$$

is constant regardless of how the set of received powers is partitioned. Note that our objective is to maximize $X\times Y$. Since, $X+Y$ is constant, $X\times Y$ is maximized when $|X - Y|$ is minimized. This is similar to the objective function of the partition problem. Therefore, to solve an instance of the partition problem for a given set S, we first sort the numbers in S in an increasing order. Then, we solve the Maximum Throughput Scheduling problem with $k = 2$ time slots for the sorted numbers as the set of powers and an arbitrary positive number as the initial noise power N_0. The reduction is obviously polynomial. Therefore, MTS is NP-hard. □

Since MTS is NP-hard, the optimal answer to the problem cannot be found in polynomial time, unless P = NP. Thus, we propose a greedy algorithm called GreedyMax which runs in polynomial time and gives a suboptimal answer to the problem. Algorithm 1 shows the pseudo code of the algorithm. The algorithm first sorts the received powers in decreasing order. Then, starting from the beginning of the list, in each step, it assigns the power to the time slot that has the minimum sum of received powers. The algorithm actually tries to make the sum of received powers assigned to the time slots as equal as possible. The rational behind Algorithm 1 is to assign the smaller powers at the end, so that it can fine tune the summation of the received powers. The same method is used in multiprocessor scheduling algorithms [28], [29].

Algorithm 1 GreedyMax: Suboptimal Solution to MTS

```
P ← list of received powers
Sort P in descending order
S ← ∅
for i ← 1 to length(P) do
    Schedule P_i at the time slot with minimum sum of powers
end for
return S
```

5 Constrained Maximum Throughput Scheduling

As described in Section 3, practical limitations prevent decoding of too many signals at a time slot. In this section, we add a constraint to the MTS problem that at most l signals can be scheduled in a time slot. Considering this constraint, we show that the problem remains NP-hard.

Problem 2 (Constrained Maximum Throughput Scheduling **(**MTS – C**)).** *Given a multiset of $n = kl$ received powers $p_1 \le p_2 \le \cdots \le p_{kl}$, and some noise power N_0, schedule the corresponding n users in k time slots so that exactly l users are assigned to each time slot and the following objective function is maximized,*

$$\sum_{i=1}^{kl} \log\left(1+\frac{p_i}{N_0+I_i}\right), \tag{9}$$

where I_i is the amount of interference power that affects the decoding of user u_i's signal.

Theorem 2. *For $l \ge 3$,* MTS – C *is NP-hard.*

Proof: We show that for $l = 3$, the problem is NP-hard. Therefore, for a general $l \ge 3$, it is NP-hard as well. Proof is by reduction from the 3-partition problem. The problem is defined as follows [27].

Definition 1 (3-Partition Problem)**.** *Given a set $\mathcal{A}$ of size $3m$, a bound $B \in Z^+$, and a size $s(a) \in Z^+$ for each $a \in \mathcal{A}$, such that each $s(a)$ satisfies $B/4 < s(a) < B/2$ and $\sum_{a\in A} s(a) = mB$. The question is: Can $\mathcal{A}$ be partitioned into m disjoint sets $S_1, \ldots, S_m$ such that for $1 \le i \le m$, $\sum_{a\in S_i} s(a) = B$?*

Note that the constraint on the item sizes implies that every S_i must contain exactly 3 elements. It is known that the 3-partition problem is NP-complete [27].

Let $\phi(p_i) = j$ denote that p_i is assigned to time slot j. We have,

$$\sum_{i=1}^{n} \log\left(1+\frac{p_i}{N_0+I_i}\right) = \log\Big(\prod_{j=1}^{k}\frac{N_0+P_j}{N_0}\Big), \tag{10}$$

where P_j is the sum of the powers assigned to time slot j, *i.e.,* $P_j = \sum_{\phi(p_i)=j} p_i$. The identity can be proved using (2), Lemma 1 and some algebraic manipulation and is omitted for the sake of brevity. We rewrite (9) using (10) as,

$$\log\Big(\prod_{i=1}^{k}\frac{N_0+\sum_{\phi(p_j)=i} p_j}{N_0}\Big). \tag{11}$$

Note that, in the 3-partition problem, $\sum_{a\in S_i} s(a) = B$ for every $1 \le i \le m$ if and only if $\prod_{i=1}^{m}\sum_{a\in S_i} s(a) = B^m$.

Given an instance of 3-partition, we construct an instance of Problem 2 such that $p_i = s(a_i)$ for every $a_i \in \mathcal{A}$, $k = m$, $l = 3$, and an arbitrary value for N_0 is chosen. Using (11), the answer to the instance of 3-partition is "Yes" if and only if the value of the objective function (9) equals $\log(\prod_{i=1}^{k} \frac{N_0+B}{N_0})$ (*i.e.*, the sum of the powers in each time slot equals B).

□

Note that for the sake of brevity, we do not propose a heuristic algorithm for MTS−C for the case $l \geq 3$. However, it is worth pointing out that GreedyMax can handle MTS−C after suitable modifications. In addition, in Section 6, we propose the GreedyFair algorithm that results in a proportionally fair schedule. GreedyFair can be used, out of the box, to obtain a suboptimal solution to MTS−C. In Section 8, through simulations, it is shown that the throughput of GreedyFair is close to the optimal throughput.

Theorem 3. *For $l \leq 2$,* MTS−C *can be solved in polynomial time.*

Proof: For $l = 1$ Theorem 3 is obvious, since there is only one way to assign the powers to the time slots. For $l = 2$, assume without loss of generality that the powers are sorted in non-decreasing order. We show that the following assignment maximizes (9):

1) For $1 \leq i \leq k$, assign p_i to time slot i.
2) For $k+1 \leq i \leq 2k$, assign p_i to time slot $2k - i + 1$.

Why the above assignment is optimal? Let i and j be two arbitrary time slots such that $i < j$. Let $\hat{p}_1, \hat{p}_2, \hat{p}_3$, and $\hat{p}_4$ be the four powers assigned to the time slots i and j. Without loss of generality, assume $\hat{p}_1 \leq \hat{p}_2 \leq \hat{p}_3 \leq \hat{p}_4$. Using (9), the throughput of any power assignment of the four powers to the two time slots is one of the following cases,

Case 1:

$$\log\left(1 + \frac{\hat{p}_1 + \hat{p}_2}{N_0}\right) + \log\left(1 + \frac{\hat{p}_3 + \hat{p}_4}{N_0}\right). \tag{12}$$

Case 2:

$$\log\left(1 + \frac{\hat{p}_1 + \hat{p}_3}{N_0}\right) + \log\left(1 + \frac{\hat{p}_2 + \hat{p}_4}{N_0}\right). \tag{13}$$

Case 3:

$$\log\left(1 + \frac{\hat{p}_1 + \hat{p}_4}{N_0}\right) + \log\left(1 + \frac{\hat{p}_2 + \hat{p}_3}{N_0}\right). \tag{14}$$

However, using algebraic manipulations, it can be shown that assuming $0 < \hat{p}_1 \leq \hat{p}_2 \leq \hat{p}_3 \leq \hat{p}_4$, Case 3 is always greater than or equal to the other two cases. It is also simple to verify that the above assignment algorithm results in the throughput of Case 3 for all i and j. Assume, the assignment of the above algorithm is not optimal and there is another assignment $\mathcal{A}$ which is optimal. Therefore, there must be at least one pair of time slots i and j in $\mathcal{A}$ such that $i < j$ and the assignments do not follow Case 3. Since, we can improve the throughput by swapping the power assignments to time slots i and j to follow Case 3, $\mathcal{A}$ is not the optimal assignment. Obviously, the assignment can be performed in polynomial time. □

Problem 3. *Given a multiset of $n \leq kl$ received powers $p_1 \leq p_2 \leq \cdots \leq p_n$, and some noise power N_0, schedule the corresponding n users in k time slots so that at most l users are assigned to each time slot and the following objective function is maximized,*

$$\sum_{i=1}^{n} \log\left(1 + \frac{p_i}{N_0 + I_i}\right), \tag{15}$$

where I_i is the amount of interference power that affects the decoding of user u_i's signal.

Corollary 1. *Problem 3 is NP-hard.*

This is obvious that if $n = kl$, Problem 3 is similar to MTS−C. In other words, Problem 2 is a special case of MTS−C. As a result, since MTS−C is NP-hard, Problem 3 is also NP-hard.

Corollary 2. *Throughput of* MTS *in each time slot scales as $\mathcal{O}(\log(n))$ where n is the number of users, while the throughput of* MTS−C *scales as $\mathcal{O}(1)$.*

Based on the definitions of the problems, one of k or l needs to be a constant (for MTS, k is fixed while for MTS−C, l is fixed). Using Lemma 1, the throughput of the schedule in each time slot is of $\mathcal{O}(\log(n/k))$. Therefore, if k is a constant, this results in $\mathcal{O}(\log(n))$ while for a constant l, it results in $\mathcal{O}(\log(l)) = \mathcal{O}(1)$.

6 Proportional Fair Scheduling

In addition to the system throughput, fairness is an important factor in wireless networks. In this paper, we consider *proportional fairness* [9], which is widely implemented in existing wireless systems [10]. In general, the system throughput is affected by the fairness definition. However, while some fairness criteria, such as max-min fairness, may sacrifice the system throughput for fairness, proportional fairness achieves a reasonable trade-off between fairness and throughput [10].

The objective of this section is to study short term fairness (*i.e.*, fairness among the users scheduled in a scheduling frame) in a wireless network with SIC at the physical layer. Long term fairness (*i.e.*, fairness among the users during a longer time than the duration of a scheduling frame) can be enforced by a higher level scheduler that selects the set of users $\{u_1, \ldots, u_n\}$ to be scheduled in each scheduling frame, which is not the focus of this paper. Before defining the proportional fair scheduling problem, we formally define proportional fairness first.

Definition 2 (Proportional Fairness). *Assume that schedule Π gives the throughput vector $\mathbf{r} = \langle r_1, \ldots, r_n \rangle$, where r_i denotes the throughput of user u_i. A schedule Π^* with throughput vector $\mathbf{r}^* = \langle r_1^*, \ldots, r_n^* \rangle$ is proportionally fair if*

Algorithm 2 GreedyFair: Proportional Fair Scheduling

Require: $p_1 \leq p_2 \leq \ldots \leq p_n$

1: **procedure** PF($\langle p_1, p_2, \ldots, p_n \rangle, N_0, k$)
2: **for** $i := 1$ to n **do**
3: $\boldsymbol{\pi}_m = \underset{\boldsymbol{\pi}}{\operatorname{argmin}} \left(\sum_{p \in \boldsymbol{\pi}} p \right)$
4: Schedule p_i on top of $\boldsymbol{\pi}_m$
5: **end for**
6: **end procedure**

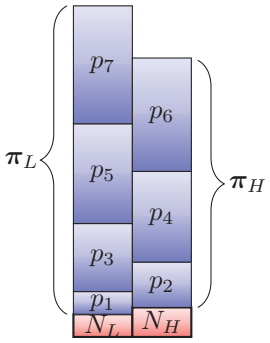


Fig. 2. The proportional fair algorithm (Algorithm 2) splits the powers into two piles $\boldsymbol{\pi}_L$ and $\boldsymbol{\pi}_H$. $\boldsymbol{\pi}_L$ contains 1^{st}, 3^{rd}, 5^{th}, ... smallest powers while $\boldsymbol{\pi}_H$ contains 2^{nd}, 4^{th}, 6^{th}, ... smallest powers. In each pile the powers are sorted from bottom to top in a non-decreasing order.

and only if the following condition holds for any other schedule Π *with throughput vector* $\mathbf{r}$*,*

$$\sum_{i=1}^{n} \frac{r_i - r_i^*}{r_i^*} \leq 0\,. \tag{16}$$

It has been shown that a proportionally fair schedule maximizes the sum of logarithm of users' utilities [9]. In this case, the utility of a users is the rate achieved by the user. Based on this property, we define the proportional fair scheduling problem as follows.

Problem 4 (Proportional Fair Scheduling **(**PFS**)).** *Given a multiset of received powers* $p_1 \leq p_2 \leq \ldots \leq p_n$*, and some noise power* N_0*, schedule the corresponding* n *users in* k *time slots so that the following objective function is maximized,*

$$\sum_{i=1}^{n} \log \left(\log \left(1 + \frac{p_i}{I_i + N_0} \right) \right), \tag{17}$$

where I_i *is the amount of interference power that affects the decoding of user* u_i*'s signal.*

Theorem 4. PFS *can be solved in polynomial time.*

We prove that the following polynomial time algorithm gives the optimal solution to PFS: Assume powers are sorted in a non-decreasing order. Starting from $i = 1$ up to n assign p_i to the time slot that minimizes I_i (see Algorithm 2). Since the assignment of the powers to the time slots takes linear time and sorting can be done in $O(n \log n)$, the time complexity of the algorithm is in $O(n \log n)$.

Assume $\phi\,(p_i)$ shows the time slot that p_i is assigned to by the above algorithm (*i.e.*, column number in Figure 2) and $\rho\,(p_i)$ shows the index of p_i in the scheduled time slot. That is, by the above algorithm, $\phi\,(p_i) = ((i-1) \mod k)+1$ and if $\rho\,(p_i) < \rho\,(p_j)$ then $p_i \leq p_j$. To complete the proof, we need the results stated in Lemma 2 and Lemma 3.

Lemma 2. *Let* $X \geq Y > 0$ *and* $A \geq B > 0$*. The following inequality holds for any* $C \geq 0$*,*

$$\log \left(\log \left(1 + \frac{X}{A+C} \right) \right) + \log \left(\log \left(1 + \frac{Y}{B} \right) \right) \geq \log \left(\log \left(1 + \frac{X}{A} \right) \right) + \log \left(\log \left(1 + \frac{Y}{B+C} \right) \right). \tag{18}$$

Proof: Rewrite (18) as follows

$$\frac{\log \left(1 + \frac{X}{A} \right)}{\log \left(1 + \frac{Y}{B} \right)} \leq \frac{\log \left(1 + \frac{X}{A+C} \right)}{\log \left(1 + \frac{Y}{B+C} \right)}. \tag{19}$$

Define function $f(u)$ as,

$$f(u) = \frac{\log \left(1 + \frac{X}{A+u} \right)}{\log \left(1 + \frac{Y}{B+u} \right)}, \tag{20}$$

and show $f(u)$ is an increasing function of $u \geq 0$. We have,

$$\frac{d}{du} f(u) = \frac{\frac{Y \log \left(1 + \frac{X}{A+u} \right)}{(B+u)(B+u+Y)} - \frac{X \log \left(1 + \frac{Y}{B+u} \right)}{(A+u)(A+u+X)}}{\log^2 \left(1 + \frac{Y}{B+u} \right)}. \tag{21}$$

To show that $f(u)$ is an increasing function of u, we need to show $\frac{d}{du} f(u) \geq 0$. Since the denominator is positive, it suffices to show that the numerator is non-negative. More formally, we need to show,

$$\frac{Y \log \left(1 + \frac{X}{A+u} \right)}{(B+u)(B+u+Y)} \geq \frac{X \log \left(1 + \frac{Y}{B+u} \right)}{(A+u)(A+u+X)}, \tag{22}$$

or, equivalently,

$$\frac{(A+u)(A+u+X) \log \left(1 + \frac{X}{A+u} \right)}{X} \geq \frac{(B+u)(B+u+Y) \log \left(1 + \frac{Y}{B+u} \right)}{Y}. \tag{23}$$

Next, define the function $g(s,t)$ as follows

$$g(s,t) = \frac{t(s+t)}{s} \log \left(1 + \frac{s}{t} \right). \tag{24}$$

To establish the Lemma, we show that the function $g(s,t)$ is non-decreasing in s and t. It can be shown that the partial derivatives of the function are non-negative. That is,

$$\frac{\partial}{\partial s} g(s,t) = \frac{t \left(s - t \log \left(1 + \frac{s}{t} \right) \right)}{s^2} \geq 0, \tag{25}$$

which follows from the fact that $z \geq \log(1+z)$. Moreover,

$$\frac{\partial}{\partial t} g(s,t) = \frac{(s+2t)\log\left(1+\frac{s}{t}\right) - s}{s} \geq 0, \tag{26}$$

which follows from the fact that $\log(1+z) \geq \frac{z}{z+2}$. □

Lemma 3. *Let $|\boldsymbol{\pi}|$ denote the number of powers in pile [1] $\boldsymbol{\pi}$ (see Figure 2) and $\mathcal{S}(\boldsymbol{\pi})$ denote the sum of the powers in pile $\boldsymbol{\pi}$ (i.e., $\mathcal{S}(\boldsymbol{\pi}) = \sum_{p_i \in \boldsymbol{\pi}} p_i$). We always have $|\boldsymbol{\pi}_L| = |\boldsymbol{\pi}_H|$ or $|\boldsymbol{\pi}_L| = |\boldsymbol{\pi}_H| + 1$. In addition,*

$$\begin{cases} \mathcal{S}(\boldsymbol{\pi}_L) \leq \mathcal{S}(\boldsymbol{\pi}_H), & \text{if } |\boldsymbol{\pi}_L| = |\boldsymbol{\pi}_H|, \\ \mathcal{S}(\boldsymbol{\pi}_L) > \mathcal{S}(\boldsymbol{\pi}_H), & \text{if } |\boldsymbol{\pi}_L| = |\boldsymbol{\pi}_H| + 1\,. \end{cases} \tag{27}$$

Proof: It is obvious that the way Algorithm 2 assigns the powers results in two piles similar to the ones shown in Figure 2. Let $\hat{p}_i$ denote the i^{th} smallest power.

For n even, $\hat{p}_1$, $\hat{p}_3$, …, $\hat{p}_{n-1}$ are assigned to $\boldsymbol{\pi}_L$ while $\hat{p}_2$, $\hat{p}_4$, …, $\hat{p}_n$ are assigned to $\boldsymbol{\pi}_H$. In this case the inequality $\mathcal{S}(\boldsymbol{\pi}_L) \leq \mathcal{S}(\boldsymbol{\pi}_H)$ is obtained by summing the pairwise inequalities: $\hat{p}_1 \leq \hat{p}_2$, $\hat{p}_3 \leq \hat{p}_4$, …, $\hat{p}_{n-1} \leq \hat{p}_n$.

For n odd, $\hat{p}_1$, $\hat{p}_3$, …, $\hat{p}_n$ are assigned to $\boldsymbol{\pi}_L$ while $\hat{p}_2$, $\hat{p}_4$, …, $\hat{p}_{n-1}$ are assigned to $\boldsymbol{\pi}_H$. In this case the inequality $\mathcal{S}(\boldsymbol{\pi}_L) > \mathcal{S}(\boldsymbol{\pi}_H)$ is obtained by summing the pairwise inequalities: $\hat{p}_3 \geq \hat{p}_2$, $\hat{p}_5 \geq \hat{p}_4$, …, $\hat{p}_n \geq \hat{p}_{n-1}$ and $\hat{p}_1 > 0$. □

Proof of Theorem 4: The proof is by induction and is divided into 4 steps. Given n received powers and k time slots, in Step 1, we prove the theorem is correct for $k = 1$ and a general n. In Step 2, the theorem is proved for $k = 2$ and $n = 2$. In Step 3, we establish the correctness of the theorem for $k = 2$ and a general n. In Step 4, we prove the theorem is correct for general k and n.

Step 1 ($k = 1$, general n): For $k = 1$, we show Algorithm 2 solves the Proportional Fair Scheduling problem. In other words, we show that for a single time slot, the optimal order of decoding is $p_n, p_{n-1}, \ldots, p_1$, *i.e.,* decode p_n first, then p_{n-1}, and so on.

The proof is by contradiction. Assume p_L and p_H are two subsequent powers in the optimal order of decoding so that $p_L < p_H$ and p_H is decoded just after p_L. We show that swapping the order of decoding of p_L and p_H will indeed increase the objective function in (17).

Since p_L and p_H are decoded successively, swapping their decoding order will only affect their individual rates, while the rates of the remaining users will remain intact. Therefore, we only need to show,

$$\log\left(\log\left(1+\frac{p_L}{N+p_H}\right)\right) + \log\left(\log\left(1+\frac{p_H}{N}\right)\right) < \log\left(\log\left(1+\frac{p_H}{N+p_L}\right)\right) + \log\left(\log\left(1+\frac{p_L}{N}\right)\right), \tag{28}$$

where N denotes the noise power N_0 plus the sum of the signal powers that are decoded after p_L and p_H. Using logarithm function properties we can rewrite (28) as,

$$\log\left(1+\frac{p_L}{N+p_H}\right)\log\left(1+\frac{p_H}{N}\right) < \log\left(1+\frac{p_H}{N+p_L}\right)\log\left(1+\frac{p_L}{N}\right). \tag{29}$$

It is clear that the sum of the two terms at each side of the inequality is constant,

$$\begin{aligned} &\log\left(1+\frac{p_L}{N+p_H}\right) + \log\left(1+\frac{p_H}{N}\right) = \\ &\log\left(1+\frac{p_H}{N+p_L}\right) + \log\left(1+\frac{p_L}{N}\right) = \\ &\log\left(1+\frac{p_L+p_H}{N}\right). \end{aligned} \tag{30}$$

In addition, it is known that the product of two terms with a constant sum is an increasing function of their difference. Thus, the product of the terms is maximized when their difference is minimized. It is then obtained that,

$$\left|\log\left(1+\frac{p_H}{N+p_L}\right) - \log\left(1+\frac{p_L}{N}\right)\right| < \left|\log\left(1+\frac{p_L}{N+p_H}\right) - \log\left(1+\frac{p_H}{N}\right)\right|, \tag{31}$$

which completes the proof.

Step 2 ($k = 2$, $n = 2$): We show for $p_H \geq p_L > 0$ and $N_H \geq N_L > 0$ we have,

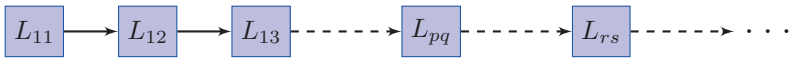
$$\log\left(\log\left(1+\frac{p_H}{N_H}\right)\right) + \log\left(\log\left(1+\frac{p_L}{N_L}\right)\right) \geq \log\left(\log\left(1+\frac{p_H}{N_L}\right)\right) + \log\left(\log\left(1+\frac{p_L}{N_H}\right)\right) \tag{32}$$

That is, scheduling the greater power on top of the higher noise and smaller power on top of the lower noise will result in a greater value for (17) in comparison to the other way around. This can simply be shown using Lemma 2 by substituting $X = p_H$, $Y = p_L$, $A = N_L$, $B = N_L$, and $C = N_H - N_L$.

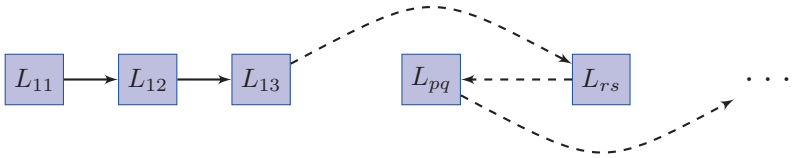
Step 3 ($k = 2$, general n): In this step, we show the correctness of the theorem for $k = 2$. Figure 2 shows the solution of the problem for $n = 7$. The algorithm splits the powers into two piles, $\boldsymbol{\pi}_L$ and $\boldsymbol{\pi}_H$. $\boldsymbol{\pi}_L$ contains 1^{st}, 3^{rd}, 5^{th}, … smallest powers, while $\boldsymbol{\pi}_H$ contains 2^{nd}, 4^{th}, 6^{th}, … smallest powers. In each pile the powers are sorted from bottom to top in a non-decreasing order. Additionally, $\boldsymbol{\pi}_L$ is always placed on top of N_L while $\boldsymbol{\pi}_H$ is placed on top of N_H.

Because of the recursive nature of the induction, we use a recursive version of Theorem 4 in this step. Algorithm 3 shows the steps of the recursive version of Algorithm 2. Given Lemma 3, it can be verified that Algorithm 3 produces the same schedule as the one given by Algorithm 2. Clearly, the theorem is correct for $n = 1$. In addition, by step 2, we know that the theorem is also correct for $n = 2$. We assume the algorithm works

1. We use the term *pile* to refer to a portion of the powers scheduled in a time slot.



(a) By Theorem 4, the sequence of L_{pq}'s shown in the figure must be a non-decreasing sequence of powers.



(b) If L_{pq}'s are not sorted in a non-decreasing order, it cannot be an optimal schedule because it contradicts the result of Step 1 and/or Step 4.

Fig. 3. In the proportional fair schedule the received powers are sorted in a non-decreasing order.

Algorithm 3 Recursive Proportional Fair Scheduling for $k = 2$

Require: $N_L + p_0 \geq N_H$

procedure RPF($\langle p_0, \ldots, p_n \rangle, N_L, N_H$)
  if $n > 0$ **then**
    Schedule p_0 on top of N_L
    RPF($\langle p_1, \ldots, p_n \rangle, N_H, N_L + p_0$)
    ▷ The requirement $p_1 + N_H \geq p_0 + N_L$ holds
  end if
end procedure

for $n = M$ powers and show the correctness of the algorithm for $n = M + 1$ powers.

Given two initial noise-plus-interference powers N_L, N_H, and $M+1$ received powers, by the result of Step 1, we know that the smallest power is scheduled either on top of N_L or on top of N_H. Let p_0 denote the smallest power while $p_1, p_2, \ldots, p_k$ denote the rest of the powers.

Having the induction assumptions, we only need to compare two cases,

1) p_0 is scheduled on top of N_H:
   In this case, since $N_H + p_0 \geq N_L$, it is obtained that $p_1, p_3, \ldots$ are scheduled on top of N_L and $p_2, p_4, \ldots$ are scheduled on top of $N_H + p_0$.
2) p_0 is scheduled on top of N_L:
   In this case, since $N_L + p_0 \geq N_H$ (using Lemma 3), it is obtained that $p_1, p_3, \ldots$ are scheduled on top of N_H and $p_2, p_4, \ldots$ are scheduled on top of $N_L + p_0$.

We show the second case is always the optimal one. To that end, we construct an inequality that shows,

$$\text{value of (17) for case 1} \leq \text{value of (17) for case 2}. \quad (33)$$

Let $\boldsymbol{\pi}_{i,<j}$ denote the sum of powers in pile i that are placed below power p_j (*i.e.*, the amount of interference but not the noise that affects p_j). For instance, in Figure 2, $\boldsymbol{\pi}_{H,<2} = 0$ and $\boldsymbol{\pi}_{L,<5} = p_3 + p_1$. Let $m = M$ if n is even, and $M - 1$ otherwise.

For n even, the optimality of the second case is shown by splitting (33) into $\frac{m}{2}$ sub-inequalities (see (34)) and showing the correctness of each sub-inequality. The correctness of (34a) is established using the result of Step 2. The correctness of (34b) to (34c) is verified by the result of Lemma 2. For instance, (34b) is proven by substitution $M = p_3$, $Y = p_2$, $A = \boldsymbol{\pi}_{H,<3} + N_L$, $B = \boldsymbol{\pi}_{L,<2} + N_L$, and $C = N_H - N_L$. The correctness of (33) can be obtained by multiplying left (right) hand side of (34a) by the left (right) hand side of (34c).

In the case that n is odd, the following extra inequality is also required,

$$\log\left(1 + \frac{p_M}{\boldsymbol{\pi}_{H,<M} + N_L}\right) \geq \log\left(1 + \frac{p_M}{\boldsymbol{\pi}_{H,<M} + N_H}\right), \quad (35)$$

which can be verified by noting that $\log(1 + \frac{p_M}{\boldsymbol{\pi}_{H,<M}+u})$ is a decreasing function of u, and $N_H \geq N_L$. This completes the proof of Step 3.

Step 4 (general k, general n): In this step, we show that the algorithm works correctly for a general $k > 2$. Note that a schedule is essentially a two-dimensional matrix of powers, where the columns of the matrix denote the time slots and the rows denote the decoding order of the received powers.

Let L_{pq} denote the power p_i that is assigned to the row p and column q of the schedule (*i.e.*, $\rho(p_i) = p$ and $\phi(p_i) = q$). By the above algorithm, L_{pq}'s are sorted in a non-decreasing order (see Figure 3 (top)). That is, if $p < r$ or $p = r$ and $q < s$ then $L_{pq} \leq L_{rs}$. Assume for some L_{pq} and L_{rs} that the condition does not hold (*i.e.*, $L_{pq} > L_{rs}$) while $p < r$ or $p = r$ and $q < s$ (see Figure 3 (bottom)). It cannot be the optimal scheduling since the fairness index in (17) can be improved by swapping L_{pq} and L_{rs} in the schedule. For $q = s$, it contradicts the result of Step 1. Otherwise, it contradicts the result of Step 3.

□

7 Scheduling with Minimum SINR Guarantee

In some applications, such as voice calls [6], a minimum SINR guarantee is required for each user, whereas in many other applications this is a desired property. A minimum SINR guarantee for each user means successful decoding of the signal with high probability plus a minimum rate guarantee. In this section, we study the problem of scheduling with minimum SINR guarantee. A formal definition of the problem follows.

$$\log\left(1+\frac{p_0}{N_L}\right)\log\left(1+\frac{p_1}{N_H}\right) \geq \log\left(1+\frac{p_0}{N_H}\right)\log\left(1+\frac{p_1}{N_L}\right) \tag{34a}$$

$$\log\left(1+\frac{p_2}{\boldsymbol{\pi}_{L,<2}+N_L}\right)\log\left(1+\frac{p_3}{\boldsymbol{\pi}_{H,<3}+N_H}\right) \geq \log\left(1+\frac{p_2}{\boldsymbol{\pi}_{L,<2}+N_H}\right)\log\left(1+\frac{p_3}{\boldsymbol{\pi}_{H,<3}+N_L}\right) \tag{34b}$$

$$\cdots \qquad \geq \qquad \cdots \tag{$\cdot$}$$

$$\log\left(1+\frac{p_{m-1}}{\boldsymbol{\pi}_{L,<m-1}+N_L}\right)\log\left(1+\frac{p_m}{\boldsymbol{\pi}_{H,<m}+N_H}\right) \geq \log\left(1+\frac{p_{m-1}}{\boldsymbol{\pi}_{L,<m-1}+N_H}\right)\log\left(1+\frac{p_m}{\boldsymbol{\pi}_{H,<m}+N_L}\right) \tag{34c}$$

Problem 5 (Scheduling with Minimum SINR Guarantee **(**MinSINR**)).** *Given a multiset of received powers $p_1 \leq p_2 \leq \cdots \leq p_n$, and some noise power N_0, schedule the corresponding n users in k time slots so that the SINR of each user is at least $\mathcal{S}_{min}$ (i.e., $\frac{p_i}{N_0+I_i} \geq \mathcal{S}_{min}, \forall i$) or output* IMPOSSIBLE *if the requirement cannot be satisfied.*

As an example, given $N_0 = 1$, $\mathcal{S}_{min} = 2$, and $\{p_1 = 2, p_2 = 3\}$ as the set of received powers, it is IMPOSSIBLE to schedule the users in $k = 1$ time slot. The reason is that, for $k = 1$, $\{\langle 2,3\rangle\}$ is the best schedule possible, which supports a minimum SINR of 1. However, for $k = 2$, the scheduling frame is $\{\langle 2\rangle, \langle 3\rangle\}$, which supports the minimum required SINR.

Proposition 1. *Let $\mathcal{F}$ be the frame that maximizes the minimum SINR of users. Also, let $\mathcal{S}_{\mathcal{F}}$ be the minimum SINR provided for the users by $\mathcal{F}$. A solution to* MinSINR *can be obtained by comparing $\mathcal{S}_{\mathcal{F}}$ and $\mathcal{S}_{min}$: if $\mathcal{S}_{\mathcal{F}} \geq \mathcal{S}_{min}$ then $\mathcal{F}$ provides the SINR requirement of the problem, otherwise, it is impossible to find such a schedule. As a result, to solve* MinSINR*, it is sufficient to find an algorithm that maximizes the minimum SINR.*

Proposition 2. *An instance of* MinSINR *might have more than one solution. For instance, given $N_0 = 1$, $\mathcal{S}_{min} = 0.1$, $k = 1$, and $\{2,3\}$ as the set of received powers, both $\{\langle 2,3\rangle\}$ and $\{\langle 3,2\rangle\}$ are correct answers. However, the answer given by Proposition 1 is always correct.*

Theorem 5. MinSINR *can be solved in polynomial time.*

Proof sketch: Algorithm 4 gives the solution to MinSINR and is based on Proposition 1. The algorithm uses a scheduling similar to Algorithm 2. Then, it computes the SINR for every user. If the SINR of any user is less than the required SINR, the algorithm outputs IMPOSSIBLE. The rest of the proof shows that the scheduling frame found by Algorithm 4, maximizes the minimum SINR.

The proof follows the four-step procedure we had for the proof of Theorem 4 with some changes in the details.
Step 1 ($k = 1$, general n): The proof is by contradiction. Assume that, in the optimal decoding order, p_L is decoded before p_H and $p_L < p_H$. It can be seen that swapping the order of decoding of p_H and p_L will either increase the minimum SINR or does not affect it. That's because

$$\min\left(\frac{p_L}{N_0+I_s}, \frac{p_H}{N_0+I_s+p_L}\right) \geq \min\left(\frac{p_H}{N_0+I_s}, \frac{p_L}{N_0+I_s+p_H}\right), \tag{36}$$

where I_s is the amount of interference which affects the signal that is decoded last.
Step 2 ($k = 2$, $n = 2$): Given $p_1 \leq p_2 \leq p_3 \leq p_4$ as the set of received powers, the optimal schedule to maximize the minimum SINR is given by $\{\langle p_1, p_3\rangle, \langle p_2, p_4\rangle\}$. This can be shown by comparing all possible scheduling frames for four users.
Step 3 ($k = 2$, general n): This step is conducted by induction and uses the same technique, which is used in Section 6. However, instead of (33) we need to show,

$$\text{minimum SINR for case 1} \leq \text{minimum SINR for case 2}\,. \tag{37}$$

Step 4 (general k, general n): This step is proven by contradiction using an approach similar to the one used in Section 6. The optimal schedule follows the one offered by Algorithm 4, otherwise, it will contradict with either Step 1 or Step 3.

Corollary 3. *The following problem can be solved in polynomial time: Given a multiset of received powers $p_1 \leq p_2 \leq \cdots \leq p_n$, some noise power N_0, and a minimum SINR $\mathcal{S}_{min}$, find the minimum number of time slots k that supports the required SINR $\mathcal{S}_{min}$.*

It follows from the fact that Problem 5 can be solved in polynomial time. The minimum k that gives a minimum SINR $\geq \mathcal{S}_{min}$ can be found using Algorithm 4 by searching over k in the range $[1, n]$. That is the minimum k for which Algorithm 4 produces a non IMPOSSIBLE output. Since Algorithm 4 runs in $\mathcal{O}(n)$, a simple linear search over k results in the overall complexity of $\mathcal{O}(n^2)$. Obviously, a binary search results in a faster running time.

Corollary 4. *The following problem is NP-hard: Given a multiset of received powers $p_1 \leq p_2 \leq \cdots \leq p_n$, and some noise power N_0, schedule the corresponding n users in k time slots so that the SINR of each user is at least $\mathcal{S}_{min}$ (i.e., $\frac{p_i}{N_0+I_i} \geq \mathcal{S}_{min}, \forall i$) and sum capacity (i.e., (5)) is maximized or output IMPOSSIBLE if the minimum SINR requirement cannot be satisfied.*

It follows from the fact that if $\mathcal{S}_{min}$ is small enough (*e.g.*, $\mathcal{S}_{min} \leq \frac{\min_i p_i}{\sum_i p_i}$), the minimum required SINR can be easily ignored, because the requirement holds for any scheduling. In this case, the problem is reduced to Problem 1, which has already been shown to be NP-hard. Note that, while the problem of Corollary 4 is generally

Algorithm 4 Scheduling with Minimum SINR Requirement

Require: $p_1 \leq p_2 \leq \ldots \leq p_n$

procedure MINSNR($\langle p_1, p_2, \ldots, p_n \rangle, N_0, k, \mathcal{S}_{min}$)
  $m = \infty$
  for $i := 1$ to k **do**
    $S_i = \emptyset$
    $N_i = N_0$
  end for
  for $i := 1$ to n **do**
    $j = 1 + (i-1) \mod k$
    $m = \min(m, \frac{p_i}{N_j})$
    $S_j = S_j \cup \{p_i\}$
    $N_j = N_j + p_i$
  end for
  if $m \geq \mathcal{S}_{min}$ **then return** $\{S_1, \ldots, S_k\}$
  else return IMPOSSIBLE
  end if
end procedure

NP-hard, some special cases of the problem can be solved in polynomial time. For instance, if Algorithm 4 returns IMPOSSIBLE for an instance of the problem of Corollary 4, the answer is definitely IMPOSSIBLE.

8 Numerical Results

In this section, we provide numerical results to show the utility and efficiency of the proposed scheduling algorithms in various simulated network scenarios.

8.1 Simulation Setup

Simulations are conducted using a custom built simulator written in Mathematica. For simulations, we consider a disk-shape network with radius $R_{\max}$ meters, where the access point is placed at the center of the disk. Node locations are drawn uniformly from inside the annulus bounded by concentric circles of radii $R_{\min}$ and $R_{\max}$.

Throughout the simulations, we set $R_{\min} = 1$ m, $R_{\max} = 1000$ m, and $N_0 = -174$ dBm. Consequently, we modify $p_{\max}$ to obtain different SNR levels.

Recall that our results and algorithms are insensitive to the specific propagation environment as we have assumed that the access point has knowledge about users' channel information (as is the case in 3G/4G networks).

8.2 Throughput Comparison by Varying n and k

The simulations conducted in this section are aimed to analyze the effect of varying the number of users, n, and the number of time slots, k, on the throughput of the network. Since the sum of received powers is proportional to n, according to Lemma 1, for a fixed k, we expect a logarithmic increase in the throughput of the network by increasing n. On the other hand, for a fixed n, by increasing k, throughput decreases because fewer users are scheduled in each time slot.

Figure 4(a) shows throughput per time slot comparison of different scheduling algorithms by varying n and k. Throughput per time slot is defined as the sum throughput of the frame divided by the number of time slots k. The plot asserts the expected behavior that for a constant number of users, decreasing k increases the throughput per time slot. The plot also asserts the logarithmic increase in the throughput by increase in sum of the received powers (Lemma 1). For $k = 1$, GreedyMax and PFS work equally well. The reason is that, in this case, all of the users are scheduled in one time slot by both of the algorithms and therefore they have the same throughput for any combination of input powers. For $n = k$ (No SIC scenario), we do not expect any increase or decrease in the average throughput with the increase of n. For $k = 2, k = 4$ and $k = 8$, GreedyMax gives a higher throughput than PFS which is expected. However, the throughput of PFS is very close to the throughput of GreedyMax which is not apparent from the formulas.

Figure 4(b) shows throughput per time slot per user for the same set of inputs as Figure 4(a). Throughput per time slot per user is defined as throughput per time slot divided by the number of users n. Recall that for a fixed k, throughput per time slot scales as $\mathcal{O}(\log n)$ with SIC while without SIC, it scales as $\mathcal{O}(1)$. However, since the number of users increases linearly, as $n \to \infty$, throughput per time slot per user converges to zero (although with SIC, convergence is slower). This behavior is asserted by Figure 4(b).

Figure 5 shows the throughput per time slot of the algorithms for different SNR levels and varying k. As we expect, by increasing k, throughput decreases. In addition, the plot shows that in the low SNR regime, the difference between the throughput of MTS and PFS is negligible. However, the difference increases as SNR increases.

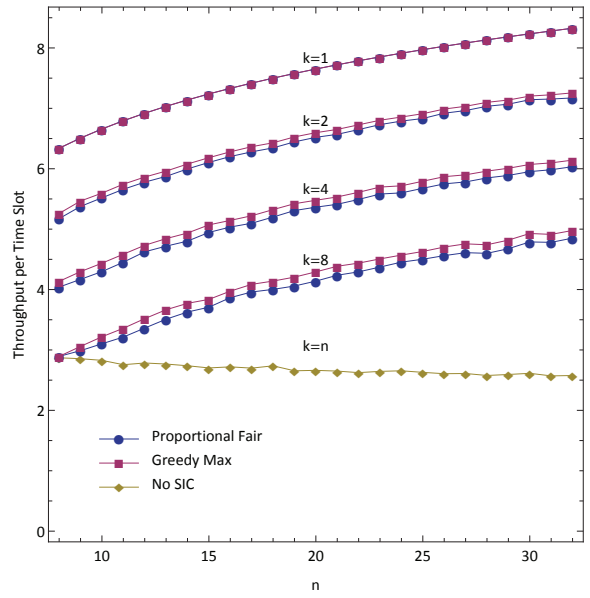
Figure 5 also shows that the performance of MTS and GreedyMax are very close in practice. Note that while in Figure 5 the performance of MTS and GreedyMax may look indistinguishable, it is easy to find cases in which they differ. For instance, consider received powers $\{9, 8, 6, 4, 3\}$ and $k = 2$. GreedyMax splits the powers into $\{9, 4, 3\}$ and $\{8, 6\}$ while the optimal schedule (given by MTS) is $\{6, 9\}$ and $\{4, 8, 3\}$.

8.3 Fairness Comparison

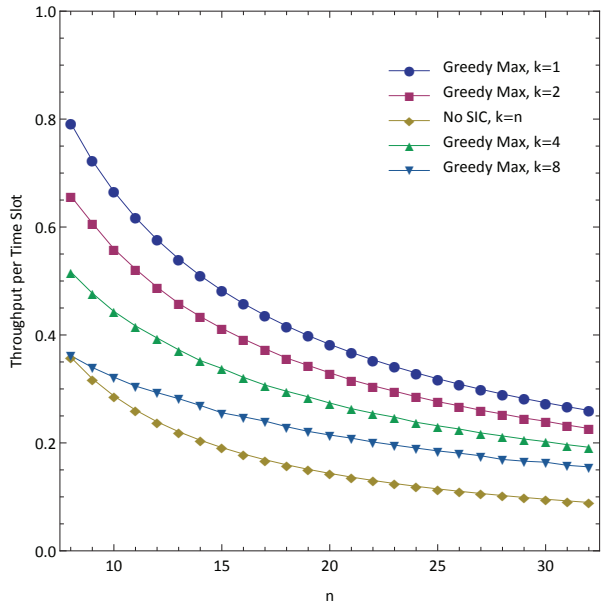
We use the Jain's fairness index, which is commonly used in the literature [30], as the measure of fairness for comparison. For a given set of throughputs $\{x_1, \ldots, x_n\}$, Jain's fairness index is defined as follows,

$$\mathcal{J}(\{x_1, \ldots, x_n\}) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2}.$$

The index ranges between $1/n$ and 1. A higher value of the index means a fairer distribution of the throughputs.



(a) Throughput per time slot



(b) Throughput per time slot per user

Fig. 4. Throughput of different SIC-enabled scheduling algorithms compared to no-SIC scheduling. Each plot shows the average of 500 runs for each case. received powers are randomly generated so that the average of SNR is 10 dB.

Figure 6 shows the result of Jain's fairness index for different algorithms and different levels of SNR by varying n. The plots show that PFS performs significantly betters in terms of fairness which is the expected behavior. Furthermore, the figure shows that the difference between the performance of the algorithms becomes more visible when SNR is high.

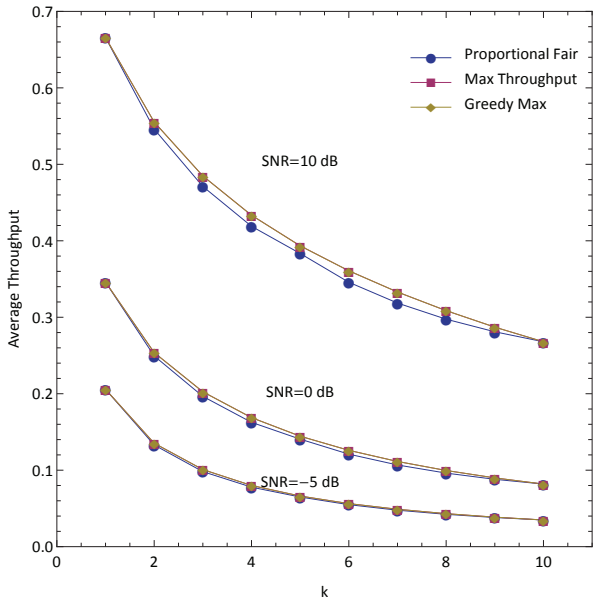


Fig. 5. Throughput per time slot per user for $n = 10$, different SNR levels, and varying k. Plot shows the average of 100 runs.

8.4 SIC Effectiveness

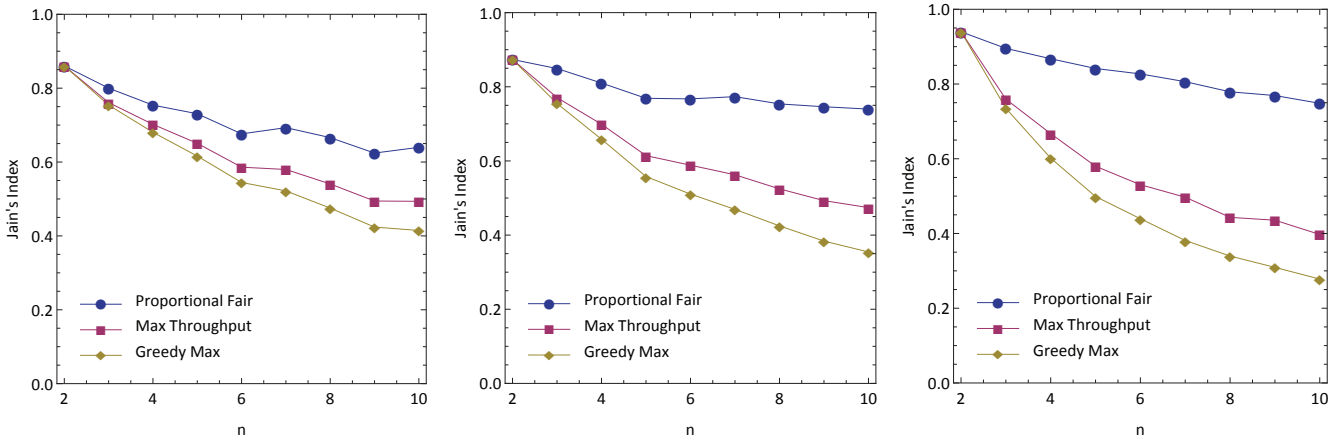
The questions considered in this section are: "How better does SIC generally perform compared to no-SIC scenario?" and "By what factors is the throughput increase affected if SIC is employed?". By no-SIC scenario, we refer to the case in which a single user is scheduled in each time slot. To answer the question we define the *percentage increase in throughput* as the percentage of increase in the throughput per time slot of GreedyMax compared to the throughput per time slot for the no-SIC scheduling (*e.g.*, 150% increase means the throughput of GreedyMax is 2.5x the throughput of no-SIC). We consider the effect of two factors: SNR and *coefficient of variation* in the simulations of this section.

Figure 7 shows the percentage of increase in throughput versus the coefficient of variation of the received powers for different SNR levels, $n = 10$ and $k = 3$. The figures also shows a linear model fitted to the points along with the 68%, 95%, and 99.7% standard deviation bands.

The figures show that there is a positive correlation between coefficient of variation and throughput increase. Additionally, the increase in throughput is much higher for lower SNR levels. Thus, we conclude, as the user channels become more diverse or the average SNR decreases, SIC becomes more effective in increasing the system throughput.

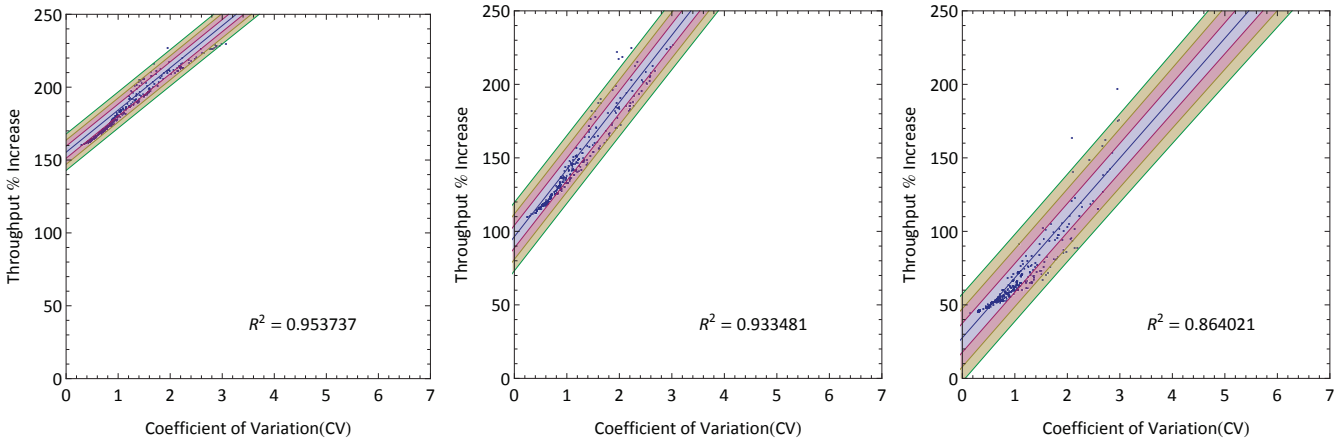
8.5 Discussion

In practice, there are several issues with SIC that prevent real systems from reaching the theoretical limits. For example, we assumed that signals are decoded in an error-free manner. However, in case an error occurs



(a) SNR $= -5$ db (b) SNR $= 0$ db (c) SNR $= +10$ db

Fig. 6. Jain's fairness index for $k = 2$ and varying n with different SNR levels. Plots show the average of 500 runs.



(a) SNR $= -5$ db (b) SNR $= 0$ db (c) SNR $= +10$ db

Fig. 7. Percentage increase in throughput using GreedyMax vs. Coefficient of Variation of received powers for $n = 10$ and $k = 3$. Each plot shows 250 runs.

while decoding a signal, it is unlikely that the signal is correctly removed from the composite signal. Thus, at later stages the signals cannot be decoded correctly [5]. Also, due to imperfect channel estimation and analog-to-digital quantization errors, the analog version of a decoded signal cannot be perfectly reconstructed. Thus, the interference removal might be imperfect, which causes lower throughput gains at later stages [31].

9 Conclusion

In this paper, we considered uplink scheduling in wireless networks supporting SIC at the physical layer. We considered four different scheduling problems with different objectives and constraints.

In the first problem (MTS) we considered maximizing the throughput of the system and showed that the problem is NP-hard. We proposed a greedy $O(n \log n)$ algorithm for the maximum throughput scheduling problem, which gives a suboptimal answer to the problem.

The second problem (MTS$-$C) is similar to the first problem except it puts a constraint on the number of decoded signals in each time slot. We showed that while the general form of the problem is NP-hard, a special case ($l \leq 2$) of the problem is solvable in polynomial time.

In the third problem (PFS) we considered proportional fair scheduling. We also proposed an $O(n \log n)$ algorithm for PFS problem and proved its correctness analytically.

The last problem considered is scheduling with some minimum SINR guarantee. We proposed an algorithm to find the optimal schedule for this problem in polynomial time.

Simulations verify the logarithmic increase in the system throughput by increase in the number of users using SIC. In addition, it is shown that while PFS performs significantly better in terms of fairness, the overall system throughput is not sacrificed by the scheduler. Since, PFS runs in polynomial time, the results suggest that PFS is

a good scheduler for the real world scenarios. Finally, simulations show that in the case that users have diverse channels or low average SNR, SIC provides a higher throughput gain compared to the non-SIC scenarios.

REFERENCES

[1] D. Halperin, J. Ammer, T. Anderson, and D. Wetherall, "Interference cancellation: better receivers for a new wireless MAC," in *ACM HotNets*, Atlanta, USA, Nov 2007.

[2] J. Andrews, "Interference cancellation for cellular systems: a contemporary overview," *IEEE Wireless Communications Magazine*, vol. 12, no. 2, 2005.

[3] J. Hou, J. Smee, H. Pfister, and S. Tomasin, "Implementing interference cancellation to increase the EV-DO Rev A reverse link capacity," *IEEE Communications Magazine*, vol. 44, no. 2, 2006.

[4] J. Blomer and N. Jindal, "Transmission capacity of wireless ad hoc networks: successive interference cancellation vs. joint detection," in *IEEE ICC*, Dresden, Germany, Jun. 2009.

[5] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[6] S. Sambhwani, W. Zhang, and W. Zeng, "Uplink interference cancellation in HSPA: principles and practice," http://www.qualcomm.co.jp/common/documents/white_papers/ul-ic-hspa.pdf, 2009.

[7] A. Zubow, M. Grauel, M. Kurth, and J. Redlich, "On uplink superposition coding and multi-user diversity for wireless mesh networks," in *IEEE Fifth International Conference on Mobile Ad-hoc and Sensor Networks*, Fujian, China, Dec. 2009.

[8] L. Li, R. Alimi, R. Ramjee, J. Shi, Y. Sun, H. Viswanathan, and Y. R. Yang, "Superposition coding for wireless mesh networks," in *ACM MobiCom*, Montréal, Canada, Sep. 2007.

[9] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, no. 1, 1997.

[10] T. Bu, L. Li, and R. Ramjee, "Generalized proportional fair scheduling in third generation wireless data networks," in *IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.

[11] R. Cruz and A. Santhanam, "Optimal routing, link scheduling and power control in multihop wireless networks," in *IEEE INFOCOM*, San Francisco, USA, Mar. 2003.

[12] D. Halperin, T. Anderson, and D. Wetherall, "Taking the sting out of carrier sense: interference cancellation for wireless LANs," in *ACM MobiCom*, San Francisco, USA, Sep. 2008.

[13] S. Sen, N. Santhapuri, R. Choudhury, and S. Nelakuditi, "Successive interference cancellation: A back-of-the-envelope perspective," in *ACM HotNets*, Monterey,USA, Oct. 2010.

[14] S. Gollakota, S. Perli, and D. Katabi, "Interference alignment and cancellation," in *ACM SIGCOMM*, Barcelona, Spain, Aug. 2009.

[15] K. Wu, H. Tan, Y. Liu, J. Zhang, Q. Zhang, and L. M. Ni, "Side channel: Bits over interference," in *ACM Mobicom*, Chicago, USA, Sep. 2010.

[16] S. Gollakota and D. Katabi, "Zigzag decoding: combating hidden terminals in wireless networks," in *ACM SIGCOMM*, Seattle, USA, Aug. 2008.

[17] L. E. Li, K. Tan, H. Viswanathan, Y. Xu, and Y. R. Yang, "Remap decoding: Simple retransmission permutation can resolve overlapping channel collisions," in *ACM Mobicom*, Chicago, USA, Sep. 2010.

[18] Y. Yu and G. Giannakis, "High-throughput random access using successive interference cancellation in a tree algorithm," *IEEE Transactions on Information Theory*, vol. 53, no. 12, 2007.

[19] P. Mitran, C. Rosenberg, and S. Shabdanov, "Throughput optimization in wireless multihop networks with successive interference cancellation," in *Wireless Telecommunications Symposium*, Apr. 2011.

[20] O. Goussevskaia, Y. A. Oswald, and R. Wattenhofer, "Complexity in geometric sinr," in *MobiHoc*, Montreal, Canada, Sep. 2007.

[21] Q. Zhao and L. Tong, "A dynamic queue protocol for multiaccess wireless networks with multipacket reception," *IEEE Transactions on Wireless Communications*, vol. 3, no. 6, 2004.

[22] M. Ghanbarinejad, C. Schlegel, and P. Gburzynski, "Adaptive probabilistic medium access in MPR-capable ad-hoc wireless networks," in *IEEE GLOBECOM*, Honolulu, USA, Dec. 2009.

[23] S. Nagaraj, D. Truhachev, and C. Schlegel, "Analysis of a random channel access scheme with multi-packet reception," in *IEEE GLOBECOM*, New Orleans, USA, Nov. 2008.

[24] Q. Zhao and L. Tong, "A multiqueue service room mac protocol for wireless networks with multipacket reception," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, 2003.

[25] K. Kumaran and L. Qian, "Scheduling on uplink of CDMA packet data network with successive interference cancellation," in *IEEE WCNC*, New Orleans, USA, Mar. 2003.

[26] S. Deb, S. Jaiswal, and K. Nagaraj, "Real-time video multicast in WiMAX networks," in *IEEE INFOCOM*, Phoenix, USA, Apr. 2008.

[27] M. Garey and D. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[28] R. Graham, "Bounds on multiprocessing timing anomalies," *SIAM Journal on Applied Mathematics*, vol. 17, no. 2, pp. 416–429, 1969.

[29] V. V. Vazirani, *Approximation Algorithms*. Springer, 2001.

[30] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *DEC Research Report TR-301*, 1984.

[31] J. Andrews and T. Meng, "Optimum power control for successive interference cancellation with imperfect channel estimation," *IEEE Transactions on Wireless Communications*, vol. 2, no. 2, 2003.

Majid Ghaderi is an Associate Professor in the Computer Science Department at the University of Calgary. Dr. Ghaderi received B.Sc. and M.Sc. degrees from Sharif University of Technology, and a Ph.D. degree from the University of Waterloo, all in Computer Science. His research interests include wireless networking and mobile computing with emphasis on modeling and performance analysis of wireless networks.

Mohsen Mollanoori is a PhD candidate in the Department of Computer Science at University of Calgary and a member of ELISA network group since 2009. He received his MSc in Software Engineering in 2007 from the Department of Electrical & Computer Engineering, Tarbiat Modares University with a first class honors. Mohsen also received his BSc in Software Engineering from Tarbiat Moallem University of Tehran with a first class honors in 2005.