On the Complexity of Wireless Uplink Scheduling with Successive Interference Cancellation

Mohsen Mollanoori and Majid Ghaderi
Department of Computer Science
University of Calgary
{mmollano, mghaderi}@ucalgary.ca

***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, and (ii) in what order in a time-slot to decode each transmission in order to maximize the system utility. 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 two scheduling problems, namely the Maximum Throughput Scheduling and Proportional Fair Scheduling. We prove the first problem is NP-hard, while the second seemingly harder problem can be solved in polynomial time.**

I. 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, while 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 considering the rest of the signals 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. The same technique is applied repeatedly to decode the remaining $n-1$ signals. Since at every stage, the remaining signals are treated as noise, the maximum rate achievable by a user depends not only on its received signal power but also on the *order* at which its signal is decoded.

With conventional decoders, transmission *scheduling* mechanisms are commonly used to avoid multiple simultaneous transmissions. That is because in conventional receivers, interference is treated as random noise. However, to increase the throughput with SIC, the scheduling mechanism should allow multiple concurrent transmissions in the network while controlling their *rate* and *decoding order* 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 is required to increase the throughput of the network [1]. It also makes SIC more practical than other multi-user detection techniques such as joint detection [3]. Furthermore, it is known that other multiple access techniques such as CDMA and OFDMA are no more efficient than SIC [4, Ch. 6]. As such, SIC has been recently considered in commercial wireless systems as a way to increase system throughput [5].

SIC can be employed for uplink [6] as well as downlink [7] transmissions. However, SIC can achieve a higher throughput on the uplink of a wireless network since the total received power is higher due to concurrent transmissions from multiple users.

In this paper, we consider two uplink scheduling problems assuming that SIC is implemented at the physical layer of the network. In the first problem, we consider maximizing the throughput of the network, where the throughput of the network is given by the summation of the throughput of individual users. In the second problem, we consider proportional fair scheduling [8]. The proportional fairness among users is achieved by maximizing the summation of the logarithm of individual users throughput. Hence, the only difference between the two scheduling problems is the additional "log" function in the objective function of the second problem. We prove that the first problem is NP-hard while interestingly the second, intuitively harder, problem can be solved in polynomial time. To this end, we propose an $O(n \log n)$ algorithm for the proportional fair scheduling problem and prove its correctness.

From the theoretical point of view, NP-hardness of the maximum throughput scheduling problem means that there is no efficient polynomial time algorithm to compute the optimal schedule (unless, P = NP). From the practical point of view, it means that there is no optimal real-time scheduler that maximizes the system throughput. 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 exponentially large).

Although, scheduling is extensively studied in traditional wireless networks [9], only a few works have explicitly considered interference cancellation.

In the broader context of multi-packet reception, there are several works on scheduling and MAC protocols [10]–[12], yet non of them is applicable to SIC-enabled networks. As an example, Zhao and Tong [10] 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. While the reception matrix model is a clever abstraction of a general multi-packet reception network, it does not accurately model the underlying dynamics of a SIC-based network such as the selection of transmission rates or the order of decoding. Other works on multi-packet reception, such as [11] and [12], 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 due to Kumaran and Qian [13] 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 maximize the network throughput with and without fairness among users. In our previous work [14], we primarily focused on heuristic algorithms for fair and efficient scheduling with SIC. Our simulation results indicated that while the maximum throughput scheduling problem is NP-hard, close to optimal heuristic algorithms exist for the problem. In this paper, we mainly focus on the computational complexity of SIC scheduling.

The rest of the paper is organized as follows. In the next section, we describe our system model and assumptions. The maximum throughput scheduling and the proportional fair scheduling problems are considered in Sections III and IV respectively. Section V concludes the paper.

II. 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 [15]).

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 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.

In this work, we consider two utility functions, namely the sum of the user throughputs (*i.e.*, $\sum_i r_i$) and the sum of the logarithm of the user throughputs (*i.e.*, $\sum_i \log(r_i)$). A detailed description of these utility functions and the justification for considering them is presented in Sections III and IV.

We only consider the case of $|\mathcal{N}| = n > k$, since the solution for the case of $n \leq k$ 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 users and has important practical implications where each user requires some minimum throughput in each scheduling frame (*e.g.*, voice calls in a cellular system [5])

In an ideal world, to maximize the system throughput, all the users have to be scheduled in every time slot [4], [13]. However, due to practical limitations, only a few overlapping signals can be decoded successfully [1]. One reason is that the decoding time in SIC linearly increases in the number of overlapping signals. The linear decoding time causes practical difficulties with large number of users [4]. 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 which propagates during the proceeding decoding stages [5]. 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. In our model, such limitations can be readily captured by controlling k and n.

Finally, to model the throughput of the users, we use the Shannon capacity function as follows

$$r_i = \log\left(1 + \frac{P_i}{N_i}\right), \tag{1}$$

where P_i is the received power of the user u_i and N_i is the noise plus interference power at the receiver that affects the decoding of the user u_i's signal.

III. Maximum Throughput Scheduling

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

Problem 1 (Maximum Throughput Scheduling). *Given a set of received powers $P_1 \leq P_2 \leq \cdots \leq P_n$, and some initial noise power N_0, schedule the corresponding n users in k time slots so that the following objective function is maximized,*

$$\sum_{i=1}^{n} \log\left(1 + \frac{P_i}{N_i}\right), \tag{2}$$

where N_i is the amount of noise plus interference power that affects the decoding of user u_i's signal (see Fig. 1).

Theorem 1. *The Maximum Throughput Scheduling problem 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,

$$\left| \sum_{x \in S'} x - \sum_{y \in (S - S')} y \right| \tag{3}$$

is minimized. The Partition problem is known to be NP-complete [16]. We show that the partition problem can be reduced (in polynomial time) to a special case of Problem 1 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. That is, 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 (2) as follows

$$\log\left(\left(1 + \frac{\sum_{p \in \boldsymbol{\pi}_1} p}{N_0}\right)\left(1 + \frac{\sum_{p \in \boldsymbol{\pi}_2} p}{N_0}\right)\right). \tag{4}$$

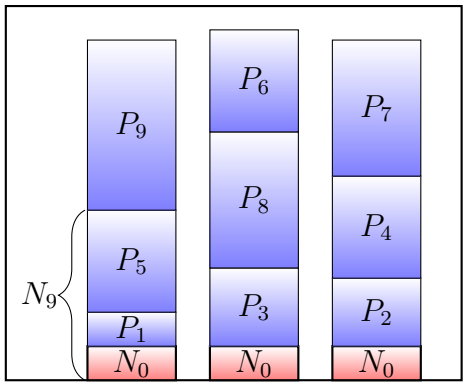


Fig. 1. Logical representation of the scheduling frame with 3 time slots and 9 powers. N_0 is the initial noise power. N_i specifies the sum of noise plus interference from other nodes, *e.g.*, in the above figure $N_9 = N_0 + P_1 + P_5$. Note that N_i is a function of the location of P_i in the scheduling frame, *i.e.*, relocation may change N_i. Figure is drawn such that in each time slot P_i's are decoded from top to bottom.

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

$$X + Y = 2 + \frac{P_1 + \ldots + P_n}{N_0} \tag{5}$$

which is constant regardless of how the set of received powers is partitioned. Note that our objective is to maximize $X \cdot Y$. Since, $X + Y$ is constant, $X \cdot Y$ 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 initial noise power N_0. The reduction is obviously polynomial. Therefore, Problem 1 is NP-hard. ■

IV. Proportional Fair Scheduling

In addition to the system throughput, fairness is an important factor in wireless networks. In this paper, we consider *proportional fairness* [8], which is widely implemented in existing wireless systems [9]. In general, the system throughput is affected by the fairness definition. However, while some fairness criteria, such as min-max fairness, may sacrifice the system throughput for fairness, the proportional fairness achieves a reasonable trade-off between fairness and throughput [9].

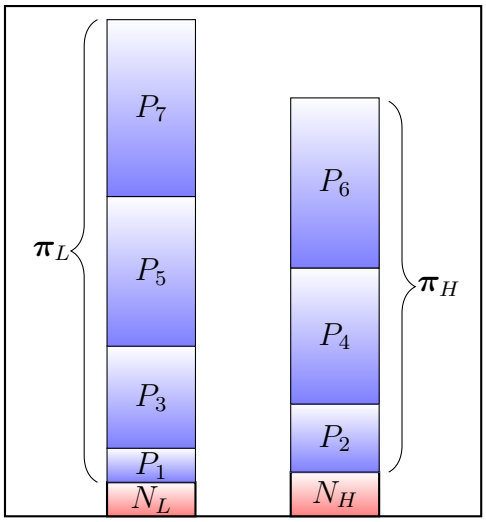


Fig. 2. The proportional fair algorithm (Algorithm 1) splits the powers into two piles $\boldsymbol{\pi}_L$ and $\boldsymbol{\pi}_H$. $\boldsymbol{\pi}_L$ contains 1^{st}, 3^{rd}, 5^{th}, ...smallest powers while $\boldsymbol{\pi}_H$ contains 2^{nd}, 4^{th}, 6^{th}, ...smallest powers. In each pile the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

Definition 1 (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 and only if the following condition holds for any other schedule* Π *with throughput vector* $\mathbf{r}$*,*

$$\sum_{i=1}^{n} \frac{r_i - r_i^*}{r_i^*} \leq 0\,. \tag{6}$$

It has been shown that a proportionally fair schedule maximizes the sum of logarithm of users' utilities [8]. Based on this property, we define the proportional fair scheduling problem as follows.

Problem 2 (Proportional Fair Scheduling). *Given a set of received powers* $P_1 \leq P_2 \leq \ldots \leq P_n$*, and some initial noise power* N_0*, schedule the corresponding* n *users in* k *time slots so that the following objective function is maximized,*

$$\sum_{i=1}^{n} \log\left(\log\left(1 + \frac{P_i}{N_i}\right)\right), \tag{7}$$

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

Theorem 2. *The Proportional Fair Scheduling problem can be solved in polynomial time.*

Algorithm 1 Proportional Fair Scheduling

Require: $P_1 \leq P_2 \leq \ldots \leq P_n$

procedure PF($\langle P_1, P_2, \ldots, P_n \rangle, N_0, k$)
  for $i := 1$ to n **do**
    $\boldsymbol{\pi}_m = \underset{\boldsymbol{\pi}}{\operatorname{argmin}} \Big(\sum_{p \in \boldsymbol{\pi}} p \Big)$
    Schedule P_i on top of $\boldsymbol{\pi}_m$
  end for
end procedure

We prove that the following polynomial algorithm gives the optimal solution to Problem 2: Starting from $i = 1$ up to n assign P_i to the time slot that minimizes N_i (see Algorithm 1). 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 $\mathcal{C}(P_i)$ shows the time slot that P_i is assigned to by the above algorithm (*i.e.*, column number in Fig. 2) and $\mathcal{R}(P_i)$ shows the index of P_i in the scheduled time slot. That is, by the above algorithm, $\mathcal{C}(P_i) = ((i-1) \mod k) + 1$ and if $\mathcal{R}(P_i) < \mathcal{R}(P_j)$ then $P_i \leq P_j$. To complete the proof, we need the results stated in Lemma 1 and Lemma 2. For the proof of Lemma 1 see Appendix 1. The proof of Lemma 2 is simple and left to the reader.

Lemma 1. *Let $X \geq Y > 0$ and $A \geq B > 0$. The following inequality holds for any $C \geq 0$,*

$$\log\left(\log\left(1+\frac{X}{A+C}\right)\right)+\log\left(\log\left(1+\frac{Y}{B}\right)\right) \geq \log\left(\log\left(1+\frac{X}{A}\right)\right)+\log\left(\log\left(1+\frac{Y}{B+C}\right)\right). \tag{8}$$

Lemma 2. *Let $|\boldsymbol{\pi}|$ denote the number of powers in pile $\boldsymbol{\pi}$ (see Fig. 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,*

$$\begin{cases} \mathcal{S}(\boldsymbol{\pi}_L) \leq \mathcal{S}(\boldsymbol{\pi}_H), & \text{if } |\boldsymbol{\pi}_L| = |\boldsymbol{\pi}_H|, \\ \mathcal{S}(\boldsymbol{\pi}_L) \geq \mathcal{S}(\boldsymbol{\pi}_H), & \text{if } |\boldsymbol{\pi}_L| = |\boldsymbol{\pi}_H| + 1\,. \end{cases} \tag{9}$$

Proof of Theorem 2: 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 1 solves the Proportional Fair Scheduling problem. In other words, we show 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 P_L and P_H will indeed increase the objective function (7).

Since P_L and P_H are decoded consequently, 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,

$$\log\left(\log\left(1+\frac{P_L}{N+P_H}\right)\right)+\log\left(\log\left(1+\frac{P_H}{N}\right)\right) < \log\left(\log\left(1+\frac{P_H}{N+P_L}\right)\right)+\log\left(\log\left(1+\frac{P_L}{N}\right)\right), \tag{10}$$

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 (10) as,

$$\log\left(1+\frac{P_L}{N+P_H}\right)\log\left(1+\frac{P_H}{N}\right) < \log\left(1+\frac{P_H}{N+P_L}\right)\log\left(1+\frac{P_L}{N}\right). \tag{11}$$

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

$$\begin{aligned} &\log\left(1+\frac{P_L}{N+P_H}\right)+\log\left(1+\frac{P_H}{N}\right) \\ &= \log\left(1+\frac{P_H}{N+P_L}\right)+\log\left(1+\frac{P_L}{N}\right) \\ &= \log\left(1+\frac{P_L+P_H}{N}\right). \end{aligned} \tag{12}$$

In addition, we know 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 can be shown that,

$$\left|\log\left(1+\frac{P_H}{N+P_L}\right)-\log\left(1+\frac{P_L}{N}\right)\right| < \left|\log\left(1+\frac{P_L}{N+P_H}\right)-\log\left(1+\frac{P_H}{N}\right)\right|, \tag{13}$$

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,

$$\log\left(\log\left(1+\frac{P_H}{N_H}\right)\right)+\log\left(\log\left(1+\frac{P_L}{N_L}\right)\right) \geq \log\left(\log\left(1+\frac{P_H}{N_L}\right)\right)+\log\left(\log\left(1+\frac{P_L}{N_H}\right)\right) \tag{15}$$

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 (7) in compare to the other way around. This can simply be shown using Lemma 1 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 $k = 2$. Fig. 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 2 in this step. Algorithm 2 shows the steps of the recursive version of Algorithm 1. Given Lemma 2, it can be verified that Algorithm 2 produces the same schedule as the one given by Algorithm 1. 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 for $n = M$ powers and show the correctness of the algorithm for $n = M+1$ powers.

Algorithm 2 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

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,

I. 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 $N_H + P_0$.

II. P_0 is scheduled on top of N_L:
In this case, since $N_L + P_0 \geq N_H$ (using Lemma 2), 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,

$$\text{value of (7) for case I} \leq \text{value of (7) for case II}. \tag{16}$$

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 Fig. 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.

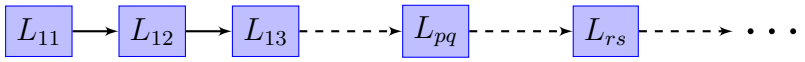
For n even, we obtain the optimality of the second case by splitting (16) into $\frac{m}{2}$ smaller inequalities (see (16)). We show the correctness of each small inequality. Then we combine the inequalities to construct the main inequality. The correctness of (16a) is established using the result of Step 2. The

$$\log\left(1+\frac{P_0}{N_L}\right)\log\left(1+\frac{P_1}{N_H}\right) \geq \log\left(1+\frac{P_0}{N_H}\right)\log\left(1+\frac{P_1}{N_L}\right) \tag{16a}$$

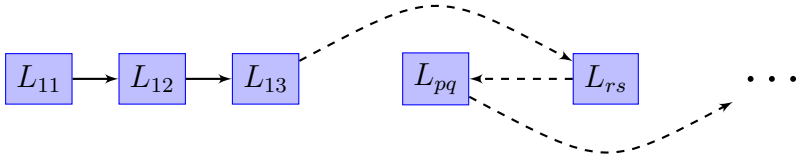
$$\log\left(1+\frac{P_2}{\boldsymbol{\pi}_{L,<2}+N_L}\right)\log\left(1+\frac{P_3}{\boldsymbol{\pi}_{H,<3}+N_H}\right) \geq \log\left(1+\frac{P_2}{\boldsymbol{\pi}_{L,<2}+N_H}\right)\log\left(1+\frac{P_3}{\boldsymbol{\pi}_{H,<3}+N_L}\right) \tag{16b}$$

$$\cdots \qquad \geq \qquad \cdots \qquad (\vdots)$$

$$\log\left(1+\frac{P_{m-1}}{\boldsymbol{\pi}_{L,<m-1}+N_L}\right)\log\left(1+\frac{P_m}{\boldsymbol{\pi}_{H,<m}+N_H}\right) \geq \log\left(1+\frac{P_{m-1}}{\boldsymbol{\pi}_{L,<m-1}+N_H}\right)\log\left(1+\frac{P_m}{\boldsymbol{\pi}_{H,<m}+N_L}\right) \tag{16c}$$



(a) By Lemma 2, 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. The optimal scheduling sorts the received powers in a non-decreasing order.

correctness of (16b) to (16c) is verified by the result of Lemma 1. For instance, (16b) 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 (16) can be obtained by multiplying the sides of (16a) to (16c).

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

$$\log\left(1+\frac{P_M}{\boldsymbol{\pi}_{H,<M}+N_L}\right) \geq \log\left(1+\frac{P_M}{\boldsymbol{\pi}_{H,<M}+N_H}\right), \tag{17}$$

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.*, $\mathcal{R}(P_i) = p$ and $\mathcal{C}(P_i) = q$. By the above algorithm, L_{pq}'s are sorted in a non-decreasing order (see Fig. 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} the condition does not hold, *i.e.*, $L_{pq} > L_{rs}$ while $p < r$ or $p = r$ and $q < s$ (see Fig. 3 (bottom)). It cannot be the optimal scheduling since the fairness index (7) 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.

■

V. Conclusion

In this paper, we considered the problem of uplink scheduling in wireless networks supporting SIC at the physical layer. We proved that the maximum throughput scheduling problem is NP-hard, while the proportional fair scheduling problem can be solved in polynomial time. We pro-

posed an $O(n \log n)$ algorithm for the proportional fair scheduling problem and proved its correctness mathematically.

APPENDIX 1

Proof of Lemma 1: Rewrite (8) as follows

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

Define function $f(u)$ as,

$$f(u) = \frac{\log\left(1+\frac{X}{A+u}\right)}{\log\left(1+\frac{Y}{B+u}\right)}, \quad (19)$$

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

$$\frac{d}{du}f(u) = \frac{\frac{Y\log\left(1+\frac{X}{A+u}\right)}{(B+u)(B+u+Y)} - \frac{X\log\left(1+\frac{Y}{B+u}\right)}{(A+u)(A+u+X)}}{\log^2\left(1+\frac{Y}{B+u}\right)}. \quad (20)$$

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,

$$\frac{Y\log\left(1+\frac{X}{A+u}\right)}{(B+u)(B+u+Y)} \geq \frac{X\log\left(1+\frac{Y}{B+u}\right)}{(A+u)(A+u+X)}, \quad (21)$$

or, equivalently,

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

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

$$g(s,t) = \frac{t(s+t)}{s}\log\left(1+\frac{s}{t}\right). \quad (23)$$

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,

$$\frac{\partial}{\partial s}g(s,t) = \frac{t\left(s - t\log\left(1+\frac{s}{t}\right)\right)}{s^2} \geq 0, \quad (24)$$

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

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

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

REFERENCES

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

[2] J. Andrews, "Interference cancellation for cellular systems: A contemporary overview," vol. 12, no. 2, Apr. 2005.

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

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

[5] S. Sambhwani, W. Zhang, W. Zeng *et al.*, "Uplink interference cancellation in hspa: Principles and practice," *QUALCOMM Inc*.

[6] A. Zubow, M. Grauel, M. Kurth, and J. Redlich, "On uplink superposition coding and multi-user diversity for wireless mesh networks," in *Proc. Mobile Ad-hoc and Sensor Networks*, China, Dec. 2009.

[7] L. Li *et al.*, "Superposition coding for wireless mesh networks," in *Proc. ACM MobiCom*, Montréal, Canada, Sep. 2007.

[8] F. Kelly, "Charging and rate control for elastic traffic," *European Trans. on Telecommunications*, vol. 8, no. 1, Jan-Feb 1997.

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

[10] Q. Zhao and L. Tong, "A dynamic queue protocol for multi-access wireless networks with multipacket reception," vol. 3, no. 6, Nov. 2004.

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

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

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

[14] M. Mollanoori and M. Ghaderi, "Fair and efficient scheduling in wireless networks with successive interference cancellation," in *Proc. IEEE WCNC*, Cancun, Mexico, Mar 2011.

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

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