

Amrinder Arora · Fanchun Jin ·  
Gokhan Sahin · Hosam Mahmoud ·  
Hyeong-Ah Choi

# Throughput Analysis in Wireless Networks with Multiple Users and Multiple Channels

April 7, 2005

**Abstract** We consider the problem of maximizing throughput in a multi-carrier wireless network that employs predictive link adaptation. We explicitly consider the time-penalty incurred due to link adaptation. The contributions of this paper are two-fold. Firstly, several high performance algorithms (offline and online) are developed for efficient performance in multiple user and multiple channel environment under the practicable lookahead prediction of one time slot. Secondly, the presented algorithms and heuristics are shown to be competitive by deterministic and probabilistic analyses. Our results show that a modest consumption of resources for channel prediction and link adaptation may result in a significant throughput improvement.

**Keywords** Wireless networks · heuristic · competitive algorithm

## 1 Introduction

Wireless connections often experience sudden changes in the quality of transmission. Transmission occurs in units called packets, where a packet carries a certain amount of information to be transmitted in unit time. A channel, which may be a combination of frequency and code or a combination of frequency and time slot, has a certain (time-varying) capacity, that may be allotted to a user either in part or in full. When the quality of transmission degrades over a channel, the network provider may adjust the channel transmission parameters or resume the connection over a different channel. The change in the channel or the transmission parameters is called *link adaptation*. The change, however, may not be instantaneous, and the penalty for switching is the loss of the allotted capacity for a certain time, during which the protocols for the switch (authorization and acknowledgment) are communicated in lieu of data. Predicting the future channel quality may also consume a significant amount of system resources (e.g., time, bandwidth and power), since it may involve transmission of training-sequences, pilot tones, or feedback messages carrying the channel state information, see [5].

In this investigation we assume, as is practicable, that we have only one predictable lookahead slot of capacity, during which time a decision must be made as to whether lose the slot (dedicate it to switching protocols) or continue to transmit data. There is a trade-off—we may continue data transmission, losing a

---

A. Arora  
Department of Computer Science, George Washington University, Washington, DC 20052

Fanchun Jin  
Department of Computer Science, George Washington University, Washington, DC 20052

Gokhan Sahin  
Electrical and Computer Engineering, Miami University, Oxford, OH 45056

Hosam Mahmoud  
Department of Statistics, George Washington University, Washington, DC 20052

Hyeong-Ah Choi  
Department of Computer Science, George Washington University, Washington, DC 20052

suddenly improved capacity on this or another channel, or use the next time slot to prepare for switching, but ending up squandering too many slots on computing the switching protocols, and incurring other associated overhead.

Our goal is to maximize the throughput of the entire network of multiple users and multiple channels. In the absence of the complete knowledge of the future of the process, as is the practice in reality, it is desirable to find efficient online algorithms, operating under the practicable assumption of admitting the prediction of only one time slot lookahead capacity. Efficient offline algorithms provide a good basis.

Previous research work in this field has focused on maintaining system stability with by bounding the queue sizes (see, for example, [1] and [2]), and on providing fairness within throughput (see [9]). Generally, the penalty incurred by link adaptation has not been considered. We augment these investigations by considering the cost associated with the changes.

The rest of this paper is organized as follows. The system model and a precise statement of both offline and online versions of the problem are presented in Section 2, together with some background. The offline version of the problem is studied in Section 3, where we prove the inherent difficulty of the problem by showing it is NP complete in Subsection 3.1. In Subsection 3.2, we present a 2-approximation algorithm for the offline version. The algorithm and its analysis appear in three subsections therein. In Section 4 we consider the online version of the problem for the single user and single channel case (single-single henceforth), and present a competitive family of algorithms in Subsection 4.1. We also consider a probabilistic analysis of our family in Subsection 4.2. We extend the algorithm to the multiple user and multiple channel case (multiple-multiple, henceforth) in Subsection 4.3, and extend the probabilistic analysis in Subsection 4.4. Finally, our conclusions and some ideas for further research are presented in Section 5.

## 2 System Model and Problem Statement

We consider the channel scheduling problem in a wireless network that consists of one base station,  $m$  mobile stations,  $f$  channels, over a period of  $n$  (large) time slots. The maximum rate that can be used by station  $i$  on channel  $j$  in time slot  $t$  is denoted by  $c(i, j, t)$ , and the rate assigned (the *usage*) by the scheduler to user  $i$  on channel  $j$  in time slot  $t$  is denoted by  $x(i, j, t)$ , which does not exceed  $c(i, j, t)$ . For  $1 \leq i \leq m$ ,  $1 \leq j \leq f$  and  $1 \leq t \leq n$ , the matrix of capacities  $c(i, j, t)$  will be called  $\mathbf{C}$  and the schedule  $x(i, j, t)$  will be called  $\mathbf{X}$ . Regardless of the scheduling algorithm, the system operates under the following additional set of natural assumptions:

- A mobile station can receive data on at most one channel at a given time slot.
- To prevent interference, a channel can be used for transmission from one base station to at most one mobile station at a given time slot.
- While changing the usage, or the intended mobile station, the next time slot is utilized for completing the switching protocol, not for data transmission.

All these *natural constraints* symbolically translate into:

1.  $x(i, j, t) \leq c(i, j, t)$ , for each  $1 \leq i \leq m$ ,  $1 \leq j \leq f$ , and  $1 \leq t \leq n$ .
2. If  $x(i, j, t) > 0$ , then for any  $j' \neq j$ ,  $x(i, j', t) = 0$ .
3. If  $x(i, j, t) > 0$ , then for any  $i' \neq i$ ,  $x(i', j, t) = 0$ .
4. For any  $j$ , if  $x(i, j, t) > 0$  and  $x(i', j, t+1) > 0$ , then  $i = i'$  and  $x(i, j, t) = x(i, j, t+1)$ .
5. For any  $i$ , if  $x(i, j, t) > 0$  and  $x(i, j', t+1) > 0$ , then  $j = j'$  and  $x(i, j, t) = x(i, j, t+1)$ .

For example, consider  $m = f = 1$  (i.e., single-single environment),  $n = 7$ , and the channel capacity matrix  $\mathbf{C} = [1, 3, 7, 8, 7, 15, 14]$ . An algorithm may produce the feasible schedule  $\mathbf{X}_1 = [0, 0, 7, 7, 7, 0, 14]$ , with throughput 35; a better algorithm may give the feasible schedule  $\mathbf{X}_2 = [1, 0, 7, 7, 0, 14, 14]$  with the higher throughput 43. In fact  $\mathbf{X}_2$  is optimal. Note that in these feasible solutions a change of usage is always preceded by 0, as required by the switching protocols.

If a change is required in the receiving station or the usage, the next downlink transmission (in time slot  $t+1$ ) is used to notify the receiver of the new usage, which is confirmed by the recipient via an acknowledgment in the next uplink transmission. Transmission at the new usage (possibly to a new user) starts only after a delay of a full duplex transmission cycle. The earliest time a new usage can come into effect after time slot  $t$  is  $t+2$ . The same penalty is assumed to be applied when a user is changed on the same channel.

Let the throughput of user  $i$  on channel  $j$  be

$$R_{ij}(n) = \sum_{t=1}^n x(i, j, t).$$

The goal is to maximize the total network throughput, that is, maximize  $\sum_{i=1}^m \sum_{j=1}^f R_{ij}(n)$  under the natural constraints. We shall compare the network throughput against the total capacity available

$$Cap(n) = \sum_{t=1}^n \sum_{i=1}^m \sum_{j=1}^f c(i, j, t).$$

We formally state the problem as follows.

**Offline Scheduling Problem (SP1)**

*Given:* A channel status matrix  $\mathbf{C} = [c(i, j, t)]$  for  $1 \leq i \leq m$ ,  $1 \leq j \leq f$ , and  $1 \leq t \leq n$ .

*Objective:* Find a schedule  $\mathbf{X} = [x(i, j, t)]$  that maximizes  $\sum_{i=1}^m \sum_{j=1}^f R_{ij}(n)$  under the natural constraints.

Next, we consider the online version of this problem. In the online version, the decision about channel allocation needs to be made at each time slot. We assume that, at the beginning of time slot  $t$ , the base station has perfect knowledge of the channel state in time slots  $t$  and  $t + 1$ . The base station decides to which station, on what channel, and at what data rate it is going to transmit during time slot  $t$ .

**Online Scheduling Problem (SP2)**

*Given:* A channel allocation matrix  $[x(i, j, t')]$ , a channel capacity matrix  $[c(i, j, t' + 1)]$  for  $1 \leq i \leq m$ ,  $1 \leq j \leq f$ , for some  $t \geq 1$ , and  $\forall t' < t$ .

*Objective:* Find the channel allocation for time slot  $t$ , i.e.,  $[x(i, j, t)]$  that maximizes  $\sum_{i=1}^m \sum_{j=1}^f R_{ij}(n)$  subject to the set of natural constraints.

Simpler single-single versions of these scheduling problems were considered in the earlier work of [3]. The results presented in this paper are an extension of some results presented there, which are summarized as:

- The single-single version of the offline problem **SP1** can be solved in  $O(n^3)$  time using a dynamic programming algorithm.
- For the online problem **SP2**, for every finite size lookahead online algorithm, there exists a sequence of capacities that makes the algorithm suboptimal. (This result is noteworthy, because the penalty is fixed to one time slot.)
- There exist two 4-competitive<sup>1</sup> 1-lookahead online algorithms for **SP2** for the single-single version: One, called WD algorithm, can be extended to the multiple-multiple environment as discussed in Section 4, and the other, called WDH algorithm, works only for the single-single environment.
- For **SP2**, for the single-single case, the WDH algorithm provides an average performance of about 96% of the optimal as noted in a simulation study.

### 3 Offline Problem Analysis

In this section, we consider the decision version of SP1, and establish its NP-completeness. In the decision version, we are given a value in addition to an instance of SP1, and we need to decide if the problem instance has a solution that has throughput matching or exceeding the specified value. Clearly, the decision version of SP1 is in NP, since an  $O(n)$  time non-deterministic algorithm can find a solution, and given a solution, we can easily verify if it is a feasible solution and if its value exceeds the specified value.

<sup>1</sup> An online algorithm is said to be  $\delta$ -competitive if the performance of the algorithm is  $\delta$ -times “worse” than the performance of an optimal offline algorithm.

### 3.1 NP-completeness of SP1

To show that SP1 is NP complete, we present a polynomial time reduction from a well-known NP-complete problem, the *3-Dimensional Matching Problem*, to SP1.

**3-Dimensional Matching Problem:** (For details see [6].)

*Given:* Disjoint sets  $T_1, T_2$  and  $T_3$  of size  $q$ , and a set  $S \subseteq T_1 \times T_2 \times T_3$ , where  $|S| = p > q$ .

*Objective:* Determine whether there exists  $S_0$ , such that  $S_0 \subseteq S$ , and  $S_0$  contains each element of  $T_1, T_2, T_3$  exactly once.

Note that if such a subset  $S_0$  exists,  $|S_0|$  must be equal to  $q$ . Consider an instance of 3DM:

$$\begin{aligned} T_1 &= \{\alpha_1, \dots, \alpha_q\}, \\ T_2 &= \{\beta_1, \dots, \beta_q\}, \\ T_3 &= \{\gamma_1, \dots, \gamma_q\}, \\ S &\subseteq T_1 \times T_2 \times T_3. \end{aligned}$$

We now construct an instance of SP1 such that 3DM is solvable if and only if total throughput for SP1 matches or exceeds a certain value. The instance of SP1 has  $m = 2q$  users,  $f = 2q$  channels and  $n = 3q - 2$  time slots. The sets of users, channels and time slots are given as follows:

$$\begin{aligned} \mathcal{U} &= \{u_1, \dots, u_q, w_1, \dots, w_q\}, \\ \mathcal{C} &= \{c_1, \dots, c_q, d_1, \dots, d_q\}, \\ \mathcal{T} &= \{t_1, t'_1, t''_1, \dots, t_{q-1}, t'_{q-1}, t''_{q-1}, t_q\}. \end{aligned}$$

Note that  $t_k = 3k - 2$ ,  $t'_k = 3k - 1$ , and  $t''_k = 3k$  for  $1 \leq k < q$ . We use the notation  $u$ -type user to indicate a  $u_i$  user, and  $w$ -type user to indicate a  $w_i$  user. Similarly,  $c$ -type channel indicates a  $c_j$  channel, and  $d$ -type channel indicates a  $d_j$  channel. For time slots, there are  $t$ -type,  $t'$ -type and  $t''$ -type timeslots. Let us introduce the notation  $I(u_i, c_j, t_k)$ .

$$I(u_i, c_j, t_k) = \begin{cases} 1, & \text{if } (\alpha_i, \beta_j, \gamma_k) \in S; \\ 0, & \text{otherwise.} \end{cases}$$

Next, we choose two constants  $L$  and  $M$  such that  $1 \ll L \ll M$  (e.g.,  $L > n$  and  $M > nL$ ). We construct capacity values for  $u$ -type and  $w$ -type users as shown in Tables 1 and 2, where the tables are shown for generic  $i$  values,  $2 \leq i \leq q - 1$ .

The capacity matrix of each  $u$ -type user is defined as:

- (i)  $m(u_i, c_j, t_k) = I(u_i, c_j, t_k)$ , for  $1 \leq j, t \leq q$ ,
- (ii)  $m(u_i, c_j, t'_k) = m(u_i, c_j, t''_k) = 0$ , for  $1 \leq j, t \leq q - 1$ ,
- (iii)  $m(u_i, d_1, t_1) = m(u_i, d_1, t'_1) = 0$ ,
- (iv)  $m(u_i, d_j, t''_{j-1}) = m(u_i, d_j, t_j) = m(u_i, d_j, t'_j) = 0$ , for  $2 \leq j \leq q - 1$ ,
- (v)  $m(u_i, d_q, t''_{q-1}) = m(u_i, d_q, t_q) = 0$ , and
- (vi) for all other  $d$ -type channels, user  $u_i$  has capacity  $L$ .

The capacity matrix of each  $w$ -type user is defined as follows:

- (i)  $m(w_1, c_j, t_1) = m(w_1, c_j, t'_1) = 0$ , for  $1 \leq j \leq q$ ,
- (ii)  $m(w_i, c_j, t''_{i-1}) = m(w_i, c_j, t_i) = m(w_i, c_j, t'_i) = 0$ , for  $1 \leq j \leq q$  and  $2 \leq i \leq q - 1$ ,
- (iii)  $m(w_q, c_j, t''_{q-1}) = m(w_q, c_j, t_q) = 0$ , for  $1 \leq j \leq q$ ,
- (iv) for all other  $c$ -type channels, user  $w_i$  has capacity  $M$ , and
- (v) for all  $d$ -type channels and all time slots, user  $w_i$  has capacity 0.

We next proceed to show that there exists a subset  $S_0 \subseteq S$  solving the 3DM problem if and only if the maximum throughput of SP1 is at least  $(3q^2 - 5q + 2)(M + L) + q$ .

Suppose the 3DM is solved using  $S_0$ . We then construct channel allocation as follows:

- if  $(\alpha_i, \beta_j, \gamma_k) \in S_0$ ,  $x(u_i, c_j, t_k) = 1$ ;
- if  $(\alpha_i, \beta_j, \gamma_1) \in S_0$ ,  $x(u_i, d_j, t) = L$  and  $x(w_i, c_j, t) = M$ , for  $t''_1 \leq t \leq t_q$ ;

**Table 1** The capacity matrix for user  $u_i$ , for  $2 \leq i \leq q-1$ 

	$t_1$	$t'_1$	$t''_1$	$t_2$	$t'_2$	$t''_2$	$\dots$	$t_{q-1}$	$t'_{q-1}$	$t''_{q-1}$	$t_q$
$c_1$	$I(u_i, c_1, t_1)$	0	0	$I(u_i, c_1, t_2)$	0	0	$\dots$	$I(u_i, c_1, t_{q-1})$	0	0	$I(u_i, c_1, t_q)$
$c_2$	$I(u_i, c_2, t_1)$	0	0	$I(u_i, c_2, t_2)$	0	0	$\dots$	$I(u_i, c_2, t_{q-1})$	0	0	$I(u_i, c_2, t_q)$
$\dots$											
$c_{q-1}$	$I(u_i, c_{q-1}, t_1)$	0	0	$I(u_i, c_{q-1}, t_2)$	0	0	$\dots$	$I(u_i, c_{q-1}, t_{q-1})$	0	0	$I(u_i, c_{q-1}, t_q)$
$c_q$	$I(u_i, c_q, t_1)$	0	0	$I(u_i, c_q, t_2)$	0	0	$\dots$	$I(u_i, c_q, t_{q-1})$	0	0	$I(u_i, c_q, t_q)$
$d_1$	0	0	$L$	$L$	$L$	$L$	$\dots$	$L$	$L$	$L$	$L$
$d_2$	$L$	$L$	0	0	0	$L$	$\dots$	$L$	$L$	$L$	$L$
$\dots$											
$d_{q-1}$	$L$	$L$	$L$	$L$	$L$	$L$	$\dots$	0	0	$L$	$L$
$d_q$	$L$	$L$	$L$	$L$	$L$	$L$	$\dots$	$L$	$L$	0	0

**Table 2** The capacity matrix for user  $w_i$ , for  $2 \leq i \leq q-1$ 

	$t_1$	$t'_1$	$\dots$	$t_{i-1}$	$t'_{i-1}$	$t''_{i-1}$	$t_i$	$t'_i$	$t''_i$	$t_{i+1}$	$t'_{i+1}$	$t''_{i+1}$	$\dots$	$t_q$
$c_1$	$M$	$M$	$\dots$	$M$	$M$	0	0	0	$M$	$M$	$M$	$M$	$\dots$	$M$
$c_2$	$M$	$M$	$\dots$	$M$	$M$	0	0	0	$M$	$M$	$M$	$M$	$\dots$	$M$
$\dots$	$M$	$M$	$\dots$	$M$	$M$	0	0	0	$M$	$M$	$M$	$M$	$\dots$	$M$
$c_{q-1}$	$M$	$M$	$\dots$	$M$	$M$	0	0	0	$M$	$M$	$M$	$M$	$\dots$	$M$
$c_q$	$M$	$M$	$\dots$	$M$	$M$	0	0	0	$M$	$M$	$M$	$M$	$\dots$	$M$
$d_1$	0	0	$\dots$	0	0	0	0	0	0	0	0	0	$\dots$	0
$d_2$	0	0	$\dots$	0	0	0	0	0	0	0	0	0	$\dots$	0
$\dots$	0	0	$\dots$	0	0	0	0	0	0	0	0	0	$\dots$	0
$d_{q-1}$	0	0	$\dots$	0	0	0	0	0	0	0	0	0	$\dots$	0
$d_q$	0	0	$\dots$	0	0	0	0	0	0	0	0	0	$\dots$	0

- if  $(\alpha_i, \beta_j, \gamma_k) \in S_0$ , for  $2 \leq k \leq q-1$ ,  $x(u_i, d_j, t) = L$  and  $x(w_i, c_j, t) = M$ , for  $t_1 \leq t \leq t'_{i-1}$  and  $t''_i \leq t \leq t_q$ ;
- if  $(\alpha_i, \beta_j, \gamma_q) \in S_0$ ,  $x(u_i, d_j, t) = L$  and  $x(w_i, c_j, t) = M$  for  $t_1 \leq t \leq t'_{q-1}$ .

The total throughput obtained by the above construction can be easily verified to be  $(3q^2 - 5q + 2)(M + L) + q$ . We have thus shown that: If there exists a solution to the 3DM problem instance, then there exists an assignment for SP1 with total throughput at least  $(3q^2 - 5q + 2)(M + L) + q$ .

Conversely, suppose there exists a solution to the SP1 problem instance with the total throughput of at least  $(3q^2 - 5q + 2)(M + L) + q$ . Let us refer to this solution as  $\mathcal{S}_1$ .

**Proposition 1** In  $\mathcal{S}_1$ , the set  $\{(u_i, c_j, t_k) \mid x(u_i, c_j, t_k) > 0\}$  contains each element of  $T_1, T_2, T_3$  at most once.

*Proof:* The proof of Proposition 1 relies on technical results presented in Lemmas 1 and 2. In the sequel, the phrase user  $i$  utilizes channel  $j$  at time slot  $k$  means that  $x(i, j, k) > 0$ .

**Lemma 1** In  $\mathcal{S}_1$ , each  $w$ -type user  $w_i$  utilizes exactly one  $c$ -type channel  $c_j$  through all time slots.

*Proof:* Let  $h(w_i, \mathcal{S})$  denote the throughput for  $w_i$  in a schedule  $\mathcal{S}$ , and  $h_{\max}(w_i)$  denote the maximum possible throughput for  $w_i$  over all schedules. Then

$$h_{\max}(w_i) = \begin{cases} (n-3)M, & \text{if } 1 < i < q; \\ (n-2)M, & \text{if } i = 1 \text{ or } q. \end{cases}$$

Thus, if each  $w$ -user uses the maximum possible throughput  $h_{\max}(w_i)$ , then the total throughput for  $w$  type users is  $(q-2)(n-3)M + 2(n-2)M$ , that is,  $(3q^2 - 5q + 2)M$ . Since  $M$  is much larger than  $L$ , the solution  $\mathcal{S}_1$  must achieve the maximum throughput for  $w$ -type users, to match or exceed the specified value. Thus, in  $\mathcal{S}_1$ , each  $w$ -type user uses a  $c$ -type channel in each time slot possible.

Say, for the purpose of contradiction, that a  $w$ -type user  $w_i$  utilizes more than one  $c$ -type channel, say channels  $c_{i1}$  and  $c_{i2}$ . Clearly, for user  $w_i$ , the change of channels cannot occur at time  $t$ , where  $t'_1 \leq t \leq t'_{i-1}$  or  $t''_i \leq t \leq t_q$ , since in that case user  $w_i$  cannot utilize time slot  $t$  because of constraint 5, and thus  $h(w_i, \mathcal{S}_1) < h_{\max}(w_i)$ .

So, let us assume that the user  $w_i$  changes channels at time slot  $t$ , for which its capacity is 0, that is,  $t \in \{t''_{i-1}, t_i, t'_i\}$ . However, in this case, we note that neither of the channels  $c_{i1}$  and  $c_{i2}$  can be used at time slot  $t$  by any other user due to constraints 4 and 5. As per the construction, all other users have a capacity

of  $M$  for all  $c$ -type channels during time slot  $t$ . Observing that there are  $q - 1$  users (besides  $w_i$ ), and  $q - 2$  available  $c$ -type channels (besides  $c_{i1}$  and  $c_{i2}$ ), we conclude that at least one other user must “miss” the time slot  $t$ , and thus  $\mathcal{S}_1$  cannot achieve a throughput of  $(3q^2 - 5q + 2)M$ . [Contradiction.]

Therefore, in  $\mathcal{S}_1$ , each  $w$ -type user  $w_i$  utilizes exactly one  $c$ -type channel  $c_j$  through all time slots.  $\square$

Lemma 1 indicates that, after assigning  $w$ -type users to  $c$ -type channels, there is exactly one time slot  $t_i$  available for  $u$ -type users to utilize on each  $c$ -type channel and for each pair of channels  $c_j$  and  $c_k$ , the available time slot is different. This implies that, in the solution of SP1, if the set of users  $\{(u_i, c_j, t_k)\}$  utilize all of the available time slots on  $c$ -type channels, then that set contains each element of  $T_2$  and  $T_3$  exactly once.

**Lemma 2** *In any solution of SP1, each  $u$ -type user  $u_i$  utilizes exactly one  $d$ -type channel  $d_j$  through all time slots.*

*Proof:* The proof is similar to that of Lemma 2.  $\square$

From Lemma 2, we know that if user  $u_i$  utilizes channel  $d_j$ , then user  $u_i$  can only utilize time slot  $t_j$  on  $c$ -type channels. This implies that each user can utilize  $c$ -type channels at no more than one time slot. The implication of Lemmas 1 and 2 proves Proposition 1. If all  $u$ -type users have a chance to utilize in the solution of SP1, then, from Proposition 1, the set  $\{(u_i, c_j, t_k) \mid x(u_i, c_j, t_k) > 0\}$  is the solution of 3DM. In this case, the throughput of SP1 should be  $(3q^2 - 5q + 2)(M + L) + q$ . The following result has been established.

**Theorem 1** *3DM is solvable if and only if the maximum throughput of SP1 is  $(3q^2 - 5q + 2)(M + L) + q$ .*

Theorem 1 indicates that 3DM is transformed to SP1 in polynomial time, which establishes the NP-completeness of the latter.

### 3.2 Offline Approximation Solution

Having proved that SP1 is NP-complete in Section 3.1, it is justified to seek an approximation algorithm for it. Before presenting it, we consider the special case when there is only one time slot. This case provides a base step that is used in the approximation algorithm, as well as in the forthcoming family of online algorithms.

#### 3.2.1 Maximum Throughput for One Time slot

If there is only one time slot, the problem is defined as follows. Given  $m$  users and  $f$  channels, with  $c(i, j, 1)$  being the capacity for the  $i$ th user on the  $j$ th channel, we wish to find an assignment that maximizes the throughput. An example capacity matrix for 6 users and 6 channels is shown in Table 3.

**Table 3** Example capacity matrix for 6 users and 6 channels for one time slot. An optimal solution is shown in boldface.

	U1	U2	U3	U4	U5	U6
C1	3	8	3	7	4	<b>8</b>
C2	<b>11</b>	7	3	9	0	2
C3	3	2	7	1	<b>9</b>	7
C4	3	<b>9</b>	3	9	8	2
C5	7	5	<b>8</b>	3	4	8
C6	8	5	3	<b>12</b>	11	4

For a single time slot, the problem of finding the maximum throughput is identical to finding the maximum weighted matching in the user-channel bipartite graph. The maximum weighted matching in a bipartite graph is a well studied problem; an optimal solution can be found in  $O((m + f)^{2.5})$  time as in [7] and [8].

Next we use this solution as a key step in the algorithm for multiple time slots.

### 3.2.2 2-Approximation Algorithm for Multiple Time slots

We propose the following algorithm *MUMC-2* for the multiple-multiple offline version of the problem. The algorithm works in two stages:

- **Step 1:** Use the maximum matching to calculate the optimal user-channel assignment for each time slot, independent of preceding and succeeding time slot. Let us store the output of this step in an array, say  $\alpha = [\alpha(1), \alpha(2), \alpha(3), \dots, \alpha(n)]$ . We call  $\alpha(t)$  the *effective capacity at time t*.
- **Step 2:** Assign *effective usage* such that the total value is maximized. This can be done using a single pass dynamic programming algorithm, that is based upon the recurrence  $\beta(k) = \max\{\beta(k-2) + \alpha(k), \beta(k-1)\}$ . The boundary conditions may be specified as  $\beta(1) = \alpha(1)$  and  $\beta(2) = \max\{\alpha(1), \alpha(2)\}$ .

**Lemma 3**  $\beta(n)$  is at least half of the sum of the sequence  $\alpha(t)$ , i.e.,  $\beta(n) \geq \frac{1}{2} \sum_{t=1}^n \alpha(i)$ .

*Proof:* Consider two specific feasible solutions:

- The feasible solution in which only odd time slots are chosen.
- The feasible solution in which only even time slots are chosen.

Since  $\beta(n)$  is optimal solution, it must be greater than both these feasible solutions. That is,

$$\begin{aligned} \beta(n) &\geq \sum_{i \text{ is odd}} \alpha(i) \text{ and } \beta(n) \geq \sum_{i \text{ is even}} \alpha(i) \\ &\Rightarrow 2\beta(n) \geq \sum_{i \text{ is odd}} \alpha(i) + \sum_{i \text{ is even}} \alpha(i) \\ &\Rightarrow \beta(n) \geq \frac{1}{2} \sum_{i=1}^n \alpha(i). \end{aligned}$$

□

**Theorem 2** Algorithm *MUMC-2* is 2-Approximation.

*Proof:* Let us call the optimum throughput for the multiple user multiple channel *OPT*. In that case, *OPT* must be less than the sum of the optimum solutions obtained for each time slot. That is,  $OPT \leq \sum_{i=1}^n \alpha(i)$ . Using Lemma 3,  $\beta(n) \geq \frac{1}{2} \sum_{i=1}^n \alpha(i) \geq \frac{1}{2} OPT$ . □

**Remarks:** We show below two simple examples, one that is the worst case for *MUMC-2*, i.e.,  $\beta(n) = \frac{1}{2} OPT$ , and one that is the best case for *MUMC-2*, i.e.,  $\beta(n) = OPT$ .

1. **An example for worst case for *MUMC-2***

Set  $c(i, j, t) = M$ . The capacity for all channels, time slots and users is the same. Any trivial solution kept constant at  $M$  for all time slots achieves the optimal result, but *MUMC-2* only achieves half of optimal throughput.

2. **An example for best case for *MUMC-2***

Set

$$c(i, j, t) = \begin{cases} M, & \text{if } t \text{ is odd;} \\ 0, & \text{if } t \text{ is even.} \end{cases}$$

### 3.2.3 Time Complexity Analysis

In the first stage, we use maximum weighted matching algorithm which runs in  $O((m+f)^{2.5})$  time, for each time slot. Thus, the total time of execution for the first stage is  $O(n(m+f)^{2.5})$ . The second stage runs in  $O(n)$ , resulting in an overall time complexity of  $O(n(m+f)^{2.5})$ .

## 4 Competitive Online Algorithm

In this section we present a family of competitive online algorithms. In [3], it was shown that no online algorithm can achieve a constant bound on the competitive ratio if no lookahead algorithm is available. However, in the following section, we show that even one lookahead information can be very useful and lead to a 4-competitive algorithm.

The algorithms presented in this section are a generalization of an earlier algorithm for the single-single case presented in [3]. It is remarkable that the competitive ratio remains the same for the much more general multiple-multiple problem.

### 4.1 Wait-Dominate Heuristic for Single User Single Channel

This family of heuristics, which we call  $WD(\gamma)$ , takes into account the usage of the previous time slot  $x(t-1)$ , the current capacity  $c(t)$ , and the next capacity  $c(t+1)$ . The parameter  $\gamma$  that defines a member of this family is assumed to exceed 1. We first present and analyze the single-single version ( $f = m = 1$ ), which will be used as a basis for the multiple-multiple case heuristic, both in construction and analysis. To make the presentation of the single-single case transparent, we use only a time-indexed notation for it. For example  $c(i, j, t)$  will be denoted simply by  $c(t)$ , while the suppressed  $i$  and  $j$  are both understood to be 1.

The intuition behind the heuristic is:

*Wait:* If the next capacity is “high” (more than  $\gamma$  of the maximum possible current usage), set the current usage to 0 (that is, dedicate the present time slot to the communication of the switching protocols so that the next (high) capacity can be used fully). We observe that the condition  $\gamma > 1$  implies that the next capacity is strictly larger than the maximum possible usage for the current time slot.

*Dominate:* Else, set the current usage to the maximum possible current capacity. Set the usage for next time slot to 0.

For the usage in the time slots  $1 \leq t < n$ , the algorithm goes according to the following precise set of rules. For boundary cases, it assumes that  $x(0) = 0$  and  $c(n+1) = 0$ . Formally, the algorithm is:

```

if  $x(t-1) > 0$  then set  $x(t) = 0$ 
else if  $c(t+1) > \gamma c(t)$  then set  $x(t) = 0$ 
else set  $x(t) = c(t)$ 

```

The second line of this algorithm corresponds to *Wait* and the third corresponds to *Dominate*. Note that if at time  $t$  a *Dominate* step is executed, the next step necessarily assigns 0 for  $x(t+1)$ .

An example of the Wait-Dominate heuristic is shown in Table 4. We note that every non-zero usage is surrounded by 0 in preceding and succeeding time slots. One might ponder on the often assignment of zeros that follow a positive usage, and the possibility of higher usage within these time slots. While this is possible in principle, and in fact it is the core idea of a Wait-Hold-Dominate algorithm that does that (see [3]), the resulting algorithms do not lend themselves easily to extension to the multiple-multiple case.

**Table 4** Example usage of Wait-Dominate heuristic.

<b>C:</b>	23	23	7	5	15	31	62	3	7	7	15	17
<b>X:</b>	23	0	7	0	0	31	0	0	7	0	15	0

**Theorem 3** (Generalization of [3]). *The Wait-Dominate heuristic  $WD(\gamma)$ , where  $\gamma > 1$ , is  $\gamma^2/(\gamma - 1)$ -competitive.*

*Proof:* Define a block  $B_i$  of length  $k_i + 1$  and starting at  $s_i$  to be:  $[c(s_i), c(s_{i+1}), \dots, c(s_i + k)]$ , such that

$$\begin{aligned} c(j) &> \gamma c(j-1), & \text{for } s_i \leq j \leq s_i + k - 1, \\ c(s_i + k) &\leq \gamma c(s_i + k - 1). \end{aligned}$$

Observe that the block size is  $k_i + 1$ , and by definition, is at least 2; the index  $i$  runs over a random number  $b$  of such blocks.

In words, a block comprises those contiguous time slots for which the capacity of each time slot is more than  $\gamma$  times that of the previous one, and one last block for which this is not the case.

The heuristic assigns the values

$$\begin{aligned} x(j) &= 0, & \forall s_i \leq j \leq s_i + k - 2, \\ x(s_i + k - 1) &= c(s_i + k - 1), \\ x(s_i + k) &= 0. \end{aligned}$$

Observe that the total capacity contained in the block  $B_i$  is

$$\begin{aligned} \sum_{j=s_i}^{s_i+k} c(j) &= c(s_i) + c(s_i + 1) + \dots + c(s_i + k - 1) + c(s_i + k) \\ &\leq \frac{c(s_i + k - 1)}{\gamma^{k-1}} + \frac{c(s_i + k - 1)}{\gamma^{k-2}} + \dots + c(s_i + k - 1) + c(s_i + k) \\ &\leq \frac{\gamma}{\gamma - 1} c(s_i + k - 1) + c(s_i + k) \\ &\leq \frac{\gamma^2}{\gamma - 1} c(s_i + k - 1). \end{aligned}$$

Thus, comparing the total usage over the block  $B_i$ ,

$$\begin{aligned} \frac{\text{Total usage for the block}}{\text{Total capacity for the block}} &= \frac{\sum_{j=s_i}^{s_i+k} x(j)}{\sum_{j=s_i}^{s_i+k} c(j)} \\ &\geq \frac{(\gamma - 1)x(s_i + k - 1)}{\gamma^2 c(s_i + k - 1)} \\ &= \frac{\gamma - 1}{\gamma^2}. \end{aligned}$$

Since the allocation for the last time slot of each block is always 0, the analysis for each block is independent, yielding (for a total of  $b$  blocks)

$$\begin{aligned} \frac{R(n)}{Cap(n)} &= \frac{\sum_{i=1}^b \text{Total Usage for } B_i}{\sum_{i=1}^b \text{Total Capacity for } B_i} \\ &\geq \frac{\sum_{i=1}^b \frac{\gamma-1}{\gamma^2} \text{Total Capacity for } B_i}{\sum_{i=1}^b \text{Total Capacity for } B_i} \\ &= \frac{\gamma - 1}{\gamma^2}. \end{aligned}$$

Since the total capacity is a weak upper bound for any offline algorithm, the result follows.  $\square$

**Optimizing  $\gamma$  for Best Competitive Ratio:** By conditioning the expression  $\frac{\gamma^2}{\gamma-1}$  over  $\gamma$ , we observe that the best competitive ratio achieved by  $WD(\gamma)$  family is for  $\gamma = 2$ , and that  $WD(2)$  is 4-competitive.

**Worst Known Example for  $WD(\gamma)$ :** We show here that the lower bound  $(\gamma - 1)/\gamma^2$  is tight by exhibiting an example where it can be asymptotically achieved by a worst-case example for  $WD(\gamma)$ . The example is presented in Table 5. Let  $0 < \varepsilon < 1$ , the total capacity  $Cap(n)$  is  $(\gamma^{n+1} - 1)/(\gamma - 1) - (n + 2)\varepsilon$ , and the throughput  $R(n)$  is  $\gamma^{n-1} - \varepsilon$ . Therefore,  $\lim_{n \rightarrow \infty} \lim_{\varepsilon \downarrow 0} \frac{R(n)}{Cap(n)} = (\gamma - 1)/\gamma^2$ .

**Table 5** Worst Known Example for Wait-Dominate heuristic.

<b>C:</b>	$1 - \epsilon$	$\gamma - \epsilon$	$\gamma^2 - \epsilon$	$\gamma^3 - \epsilon$	...	$\gamma^{n-1} - \epsilon$	$\gamma^n - 3\epsilon$
<b>X:</b>	0	0	0	0	...	$\gamma^{n-1} - \epsilon$	0

#### 4.2 Probabilistic Analysis of $WD(\gamma)$ Algorithm

In this section, we analyze the probabilistic behavior of the Wait-Dominate algorithms. Our purpose is to show that such probabilistic analysis is possible, in exact form, albeit the strong dependency in the usage and capacity sequences involved. For an event  $\mathcal{E}$  let indicator  $I_{\mathcal{E}}$  be 1 if  $\mathcal{E}$  occurs, and 0 otherwise. We shall carry out our analysis under any general joint probability distribution

$$Q_r(k_1, \dots, k_{r+2}; t) := \mathbf{Prob} \{c(t-r) = k_1, \dots, c(t+1) = k_{r+2}\}.$$

In this joint probability distribution we allow the marginal distributions to be dependent or even time-varying. In spite of the combinatorial complexity of the exact moments of the usage, we shall see that an asymptotic analysis is possible in degenerate cases of independent capacities admitting some simple standard usage distribution.

For  $1 \leq t < n$ , the  $WD(\gamma)$  algorithm assigns the current usage such that  $x(t) = c(t)$  if and only if  $x(t-1) = 0$  and  $c(t+1) \leq \gamma c(t)$ . Let  $A(t)$  and  $B(t)$  respectively denote the events  $\{c(t+1) \leq \gamma c(t)\}$  and  $\{x(t-1) = 0\}$ . We then have

$$x(t) = c(t)I_{A(t)}I_{B(t)}. \quad (1)$$

**Lemma 4**  $I_{B(t)} = 1 - I_{A(t-1)}I_{B(t-1)}$

*Proof:*

$$\begin{aligned} I_{B(t)} &= 1 - I_{\{x(t-1) > 0\}} \\ &= 1 - I_{\{c(t) \leq \gamma c(t-1), x(t-2) = 0\}} \\ &= 1 - I_{A(t-1)}I_{B(t-1)}. \end{aligned}$$

□

**Proposition 2** *At time  $t$ , the  $s$ th moment of the usage of the  $WD(\gamma)$  algorithm is*

$$\mathbf{E}[x^s(t)] = \sum_{r=0}^{t-1} (-1)^r \sum_{k_{r+2} \leq \gamma k_{r+1} \leq \gamma^2 k_r \leq \dots \leq \gamma^{r+1} k_1} k_{r+1}^s Q_r(k_1, \dots, k; t).$$

*Proof:* The recursive formulation of  $B(t)$  in Lemma 4 allows us to represent the usage at time slot  $t$  in an inductive manner. By (1), we have

$$\begin{aligned} x^s(t) &= (c(t)I_{A(t)}I_{B(t)})^s \\ &= c^s(t)I_{A(t)}I_{B(t)} \\ &= c^s(t)I_{A(t)}[1 - I_{A(t-1)}I_{B(t-1)}] \\ &= c^s(t)[I_{A(t)} - I_{A(t)}I_{A(t-1)}(1 - I_{A(t-2)}I_{B(t-2)})] \\ &= c^s(t)[I_{A(t)} - I_{A(t)}I_{A(t-1)} + I_{A(t)}I_{A(t-1)}I_{A(t-2)}I_{B(t-2)}] \\ &\quad \vdots \\ &= c^s(t) \left( (-1)^{t-2} I_{A(t)} \dots I_{A(2)} I_{B(2)} + \sum_{r=0}^{t-3} (-1)^r \prod_{j=0}^r I_{A(t-j)} \right). \end{aligned}$$

Plugging in the boundary condition  $I_{B(2)} = I_{\{c(2) > \gamma c(1)\}} = 1 - I_{A(1)}$ , we obtain

$$x^s(t) = c^s(t) \sum_{r=0}^{t-1} (-1)^r \prod_{j=0}^r I_{A(t-j)}.$$

Taking expectations, we obtain

$$\begin{aligned} \mathbf{E}[x^s(t)] &= \sum_{r=0}^{t-1} (-1)^r \mathbf{E}[c^s(t) I_{A(t)} \cdots I_{A(t-r)}] \\ &= \sum_{r=0}^{t-1} (-1)^r \mathbf{E}[c^s(t) I_{A(t), A(t-1), \dots, A(t-r)}]. \end{aligned}$$

Finally,  $\mathbf{E}[c^s(t) I_{\{A(t), A(t-1), \dots, A(t-r)\}}]$  can be computed from an underlying conditional expectation:

$$\begin{aligned} &\mathbf{E}[c(t) I_{\{c(t+1) \leq \gamma c(t), \dots, c(t-r+1) \leq \gamma c(t-r)\}} \mid c(t-r) = k_1, \dots, c(t+1) = k_{r+2}] \\ &\quad \times Q_r(k_1, \dots, k_{r+2}; t) \\ &= k_{r+1}^s I_{\{k_{r+2} \leq \gamma k_{r+1} \leq \gamma^2 k_r \leq \dots \leq \gamma^{r+1} k_1\}} Q_r(k_1, \dots, k_{r+2}; t). \end{aligned}$$

□

Proposition 2 is valid for all joint distributions of capacities, and for whatever complex pattern of dependency therein. We instantiate its use by an example where the capacities are independent random variables.

Consider for instance a sequence of capacities that are independent identically distributed random variables, with distribution like that of  $KBer(p)$ , where  $K$  is a constant and  $Ber(p)$  is a Bernoulli random variable with success rate  $p$ . This example corresponds to an intended constant transmission rate  $K$ , but in practice the channel occasionally fails and becomes inaccessible. For a quick calculation of the moments, we split the sum in proposition 2 into terms corresponding to  $k_{r+1} = 0$ , and terms corresponding to  $k_{r+1} = K$ . And so, by independence we obtain the moments

$$\mathbf{E}[x^s(t)] = 0 + \sum_{r=0}^{t-1} (-1)^r \sum_{\substack{k_{r+2} \leq \gamma k_{r+1} \leq \gamma^2 k_r \leq \dots \leq \gamma^{r+1} k_1 \\ k_{r+1} = K}} K^s \prod_{j=1}^{r+2} \mathbf{Prob}(c(t-r+j-1) = k_j).$$

When any of the indexes  $k_1, \dots, k_r$  is 0, the range of the multi-folded sums is empty. This leaves only terms corresponding to  $k_1 = k_2 = \dots = k_{r+1} = K$ , reducing the calculation to

$$\begin{aligned} \mathbf{E}[x^s(t)] &= \sum_{r=0}^{t-1} (-1)^r \sum_{k_{r+2} \leq \gamma K} K^s p^{r+1} \mathbf{Prob}(c(t+1) = k_{r+2}) \\ &= \sum_{r=0}^{t-1} (-1)^r K^s p^{r+1} \\ &= \frac{pK^s}{p+1} (1 + (-1)^{t+1} p^t) \\ &\rightarrow \frac{pK^s}{p+1}, \quad \text{as } t \rightarrow \infty. \end{aligned}$$

Further,  $pK^s/(p+1)$  is the  $s$ th moment of  $KBer(p/(1+p))$ , and the Bernoulli random variable is uniquely characterized by all its moments (by Carleman's condition, see a standard reference like [4]). So,

$$x(t) \rightarrow KBer\left(\frac{p}{p+1}\right), \quad (\text{in distribution}). \quad (2)$$

For  $s = 1$  we have the average usage at  $t$ , giving an average throughput over  $n$  time slots of

$$\begin{aligned}
\mathbf{E}[R(n)] &= \mathbf{E}[c(n)] + \sum_{t=1}^{n-1} \mathbf{E}[x(t)] \\
&= pK + \sum_{t=1}^{n-1} \frac{pK}{p+1} (1 + (-1)^{t+1} p^t) \\
&= pK + \frac{p}{p+1} K(n-1) + \frac{Kp^2}{(p+1)^2} (1 + (-1)^n p^{n-1}) \\
&= \frac{p}{p+1} Kn + O(1).
\end{aligned}$$

Note that  $\gamma$  does not play an important role here.

As an average measure of utilization, we compare the expected throughput against the expected total capacity  $\mathbf{E}[Cap(n)] = pKn$ , and see that, as  $n \rightarrow \infty$ ,

$$\frac{\mathbf{E}[R(n)]}{\mathbf{E}[Cap(n)]} \rightarrow \frac{1}{p+1}.$$

If  $p$  is small, this utilization goes up. Under very poor reception ( $p$  close to 0) the ratio approaches 1, because in fact many capacities are 0 anyway, achieving a very low throughput (over an equally very low total capacity). If the conditions of reception are almost perfect ( $p$  close to 1), most capacities will be full, giving rise to a solid sequence of  $K$ 's. However, after each full usage we must assign 0 (for the switching protocols) and (even optimally) at most half the solid sequence can be used; the average measure of utilization approaches  $\frac{1}{2}$ .

We also considered another example, where  $\gamma$  mattered. We took  $\gamma = 2$  (the best choice) and looked at the situation when the capacities  $c(t)$  are independent and each is distributed uniformly on the discrete set  $\{0, 1, \dots, M\}$ . In this case

$$Q(k_1, \dots, k_{r+2}) = \frac{1}{(M+1)^{r+2}}.$$

Subsequently, the average usage at time  $t$  is

$$\mathbf{E}[x(t)] = \sum_{r=0}^{t-1} \frac{(-1)^r}{(M+1)^{r+2}} \sum_{k_{r+2} \leq 2k_{r+1} \leq 4k_r \leq \dots \leq 2^{r+1}k_1} k_{r+1}^s. \quad (3)$$

We computed this multi-folded sum for  $M = 1000$ . Up to  $t = 30$  each new term introduced a correction. Each even term reduced a previous upper bound, and each lower term increased a previous lower bound. Our numerical experiment indicates the utilization ratio is 0.55988 for  $t = 30$  and higher. The simulations also show that efficiency (i.e. actual assignment versus optimal assignment) ranging between 80% and 90% is attained.

### 4.3 Adapting the WD Heuristic for Multiple Users and Multiple Channels

In this section, we use several of the ideas in the offline 2-approximate algorithm, in conjunction with a  $WD(\gamma)$  applied to the maximum matching (effective capacity) at a time slot to assign capacities in the multiple-multiple environment. We refer to the adapted heuristic as  $MUMC-ON(\gamma)$ . When applied to successive time slots, the  $MUMC-ON(\gamma)$  algorithm is competitive and gives a high network throughput.

At time slot  $t$  we have a ‘‘slice’’ in time of the continually changing capacity matrix for all the users on all the channels. For instance, with  $m = 6$ , and  $f = 6$ , at time  $t$ , the snapshot in time of available capacities might be something like the matrix in Table 3. There is an effective capacity  $\alpha(t)$ , which is the sum of the capacities chosen in the maximum matching. In the running example,  $\alpha(t) = 11 + 9 + 8 + 12 + 9 + 8 = 57$ . The idea in  $MUMC-ON(\gamma)$  is to employ this combined usage at time  $t$ , together with the prediction  $\alpha(t+1)$  in a one-dimensional manner just like the single-single  $WD(\gamma)$ : we compare two successive effective capacities  $\alpha(t)$  and  $\alpha(t+1)$ , switch to  $\alpha(t+1)$ , if  $\alpha(t+1)$  is particularly promising (more than  $\gamma$  of the current effective capacity), otherwise keep the current effective capacity. As usual, while switching all the users will lose a time slot.

The algorithm  $MUMC-ON(\gamma)$  is specified as follows:

- **Step 1:** Calculate the effective capacity (maximum matching)  $\alpha(t)$ , for each time slot  $t$  independently. This step is the same as Step 1 for the offline 2-approximation algorithm  $MUMC-2$ .
- **Step 2:** Assign the usage according to the Wait-Dominate heuristic  $WD(\gamma)$ . if  $\alpha(t+1) > \gamma\alpha(t)$ , all the users prepare to switch to their next capacity by receiving 0 usage, otherwise, each of them continues its current usage. We recall that the  $WD(\gamma)$  heuristic has the property that each non-empty allocation (of effective usage) is surrounded by 0 allocation of usage in the preceding and succeeding time slots.

Formally stated, the algorithm is just like that given in Subsection 4.1, the one difference being every  $c(t)$  is replaced by  $\alpha(t)$ , an effective capacity at time  $t$ .

#### 4.3.1 Example of $MUMC-ON(\gamma)$

In this section, we present a complete example of the  $MUMC-ON(2)$  algorithm. Our example consists of 3 users, 4 channels, and 5 time slots. The flow of information is shown in Table 6, and consists of phases as described above.

**Table 6** A complete example of  $MUMC-ON$  algorithm: (i) Input matrices, (ii) The  $\alpha$  list and application of  $WD$  algorithm, and (iii) Actual allocation.

$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
$\begin{Bmatrix} 8 & 3 & 2 \\ 4 & 6 & 9 \\ 6 & 2 & 4 \\ 7 & 7 & 1 \end{Bmatrix}$	$\begin{Bmatrix} 5 & 5 & 4 \\ 4 & 3 & 7 \\ 5 & 2 & 5 \\ 9 & 7 & 2 \end{Bmatrix}$	$\begin{Bmatrix} 2 & 3 & 2 \\ 3 & 1 & 4 \\ 3 & 2 & 4 \\ 5 & 5 & 1 \end{Bmatrix}$	$\begin{Bmatrix} 10 & 8 & 3 \\ 9 & 9 & 9 \\ 6 & 7 & 7 \\ 5 & 8 & 9 \end{Bmatrix}$	$\begin{Bmatrix} 9 & 5 & 7 \\ 7 & 8 & 8 \\ 7 & 7 & 8 \\ 8 & 8 & 6 \end{Bmatrix}$

(i) Input matrices

$\alpha:$	24	21	12	28	25
$X:$	24	0	0	28	0

(ii) The  $\alpha$  list and application of  $WD$  algorithm

$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
$\begin{Bmatrix} 8 & 0 & 0 \\ 0 & 0 & 9 \\ 0 & 0 & 0 \\ 0 & 7 & 0 \end{Bmatrix}$	$\begin{Bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{Bmatrix}$	$\begin{Bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{Bmatrix}$	$\begin{Bmatrix} 10 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 9 \end{Bmatrix}$	$\begin{Bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{Bmatrix}$

(iii) Actual allocation.

#### 4.3.2 Analysis of $MUMC-ON$

**Theorem 4** *The algorithm  $MUMC-ON(\gamma)$  is  $\gamma^2/(\gamma-1)$ -competitive.*

*Proof:* The proof is based on the proof for  $WD$  algorithm that is used in Step 2. Let the value of an optimal offline algorithm be given by  $OPT$ . The optimum value cannot exceed the sum of optimum values obtained independently for each time slot:

$$OPT \leq \sum_{t=1}^n \alpha(t).$$

Let the throughput using the  $MUMC-ON(\gamma)$  be  $\beta$ . Employing the  $\gamma^2/(\gamma-1)$ -competitiveness of the  $WD(\gamma)$ .

$$\beta \geq \frac{\gamma-1}{\gamma^2} \sum_{t=1}^n \alpha(t) \geq \frac{\gamma-1}{\gamma^2} OPT.$$

□

#### 4.4 Analysis of Multiple User Multiple Channel Version

As already mentioned, the  $\mathcal{MUMC-ON}(\gamma)$  algorithm is just like its one dimensional counterpart  $\text{WD}(\gamma)$ , except that the effective capacity is used at each stage. Therefore, the probabilistic analysis carries over mutatis mutandis (only a change of  $c(t)$  to  $\alpha(t)$  is required).

**Proposition 3** *Let  $\alpha(t)$  be the effective capacity at time  $t$  in a network using the  $\mathcal{MUMC-ON}(\gamma)$  algorithm. The  $s$ th moment of the usage at this time slot is*

$$\mathbf{E}[x^s(t)] = \sum_{r=0}^{t-1} (-1)^r \sum_{k_{r+2} \leq \gamma k_{r+1} \leq \gamma^2 k_r \leq \dots \leq \gamma^{r+1} k_1} k_{r+1}^s \tilde{Q}_r(k_1, \dots, k; t),$$

where

$$\tilde{Q}_r(k_1, \dots, k_{r+2}; t) := \mathbf{Prob}\{\alpha(t-r) = k_1, \dots, \alpha(t+1) = k_{r+2}\}.$$

We illustrate Proposition 3 by a simple example. Suppose we have only two users, competing at each time slot for the same channel under the  $\mathcal{MUMC-ON}(2)$  algorithm. The time slice here is  $\mathbf{Y}(t) = (Y_1(t), Y_2(t))$ , a pair of independent  $\text{Ber}(p)$  random variables, and all  $\mathbf{Y}(t), t = 0, 1, \dots, n$ , are independent. Clearly, The effective capacities come across as  $K$  times one combined Bernoulli random variable, because the effective capacity is either 0, when  $Y_1(t) = Y_2(t) = 0$ , or it is  $K$  when either or both  $Y_1(t)$ , and  $Y_2(t)$  are  $K$  (in the case of both being  $K$  we still get an effective capacity  $K$ , owing to the constraint that no two users can share a channel). In other words, the maximum matching is  $\alpha(t) = \max\{Y_1(t), Y_2(t)\}$ , which is distributed like  $\text{Ber}(p')$  random variable, with  $p' = 2pq + p^2 = 1 - q^2$ . The analysis has been reduced to that of a single-single  $\text{WD}(2)$  algorithm, with the capacities  $c(t)$  being an independent Bernoulli  $\text{Ber}(p')$  sequence. For this latter system we obtained a result. Whence, for  $m = 2$ , and  $f = 1$  we have

$$\frac{\mathbf{E}[\tilde{R}(n)]}{\mathbf{E}[\tilde{C}ap(n)]} \rightarrow \frac{p'}{2p(p'+1)} = \frac{1-q^2}{2p(2-q^2)};$$

here  $\tilde{R}(n)$  is interpreted as the effective throughput for the entire network, i.e.

$$\tilde{R}(n) = \sum_{i=1}^2 \sum_{t=1}^n x(i, 1, t),$$

and  $\tilde{C}ap(n)$  is interpreted as the total available capacity, i.e.

$$\tilde{C}ap(n) = \sum_{t=1}^n \tilde{c}(t),$$

where

$$\tilde{c}(t) = \begin{cases} 0, & \text{with probability } q^2, \\ K, & \text{with probability } 2pq, \\ 2K, & \text{with probability } p^2, \end{cases}$$

and across time the variables  $\tilde{c}(t)$  are independent.

---

## 5 Conclusion and Future Work

Channel-aware scheduling and link adaptation will play an important role in achieving the high data rates required by emerging mobile applications. In this paper, we considered the problem of maximizing throughput in a multi-carrier wireless network that employs link adaptation.

We first analyzed the concept of link adaptation and based on that, we formulated a combinatorial problem to model the scheduling problem in a multi-carrier wireless network with multiple users. We considered the offline version of the problem and proved the inherent complexity of the problem by presenting a reduction from a known NP-complete problem. We presented a 2-approximation algorithm for the offline version. We also formulated the more practical online version of the problem and generalized a known algorithm for the single user single channel case. We extended the algorithm for multiple user multiple channel environment and proved that the modified algorithm is 4-competitive, and also presented an average case analysis of our algorithm.

Our results show that a modest lookahead capability of one time slot allows us to provide a 4-competitive algorithm for multiple user and multiple channel environment.

We have not considered the quality of service requirements for different users in the scheduling problem. A natural extension of this work would be to study the scheduling algorithm considering the quality of service.

## References

1. Andrews, M., Zhang, L.: Scheduling over a time-varying user-dependent channel with applications to high speed wireless data. In: Proceedings of the 43rd Symposium on Foundations of Computer Science, pp. 293–302. IEEE Computer Society (2002)
2. Andrews, M., Zhang, L.: Scheduling over non-stationary wireless channels with finite rate sets. In: IEEE INFOCOM 2004 - The Conference on Computer Communications, vol. 23:1, pp. 1695 – 1705 (2004)
3. Arora, A., Choi, H.: Channel aware scheduling for throughput maximization. Submitted. Also available at <http://www.seas.gwu.edu/~hchoi/publication/wireless.htm>
4. Billingsley, P.: Probability and Measure. Wiley, New York (1986)
5. Catreux, S., Erceg, V., Gesbert, D., Heath, R.: Adaptive modulation and MIMO coding for broadband wireless data networks. IEEE Communications Magazine **40**, 108–115 (2002)
6. Garey, M., Johnson, D.: Computers and Intractability - A Guide to the Theory of NP-completeness. Freeman (1979)
7. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum flow problem. J. Assoc. Comput. Mach. **35**, 921–940 (1988)
8. Hopcroft, J.E., Karp, R.M.: An  $n^{5/2}$  algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing **2**(4), 225–231 (1973)
9. Tsibonis, V., Georgiadis, L., Tassiulas, L.: Exploiting wireless channel state information for throughput maximization. In: Proceedings of IEEE Infocom '03, pp. 301–310 (2003)