HAL
open science

Distributed Call Scheduling in Wireless Networks

Jean-Claude Bermond, Dorian Mazauric, Vishal Misra, Philippe Nain

To cite this version:

Jean-Claude Bermond, Dorian Mazauric, Vishal Misra, Philippe Nain. Distributed Call Scheduling in Wireless Networks. [Research Report] RR-6763, INRIA. 2008. inria-00345669v3

HAL Id: inria-00345669

https://inria.hal.science/inria-00345669v3

Submitted on 21 Nov 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Distributed Call Scheduling in Wireless Networks

Jean-Claude Bermond — Dorian Mazauric — Vishal Misra — Philippe Nain

N° 6763

October 2009

Thème COM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--6763--FR+ENG

Distributed Call Scheduling in Wireless Networks

Jean-Claude Bermond[*†], Dorian Mazauric[*‡†] , Vishal Misra[§] , Philippe Nain[*‡†]

Thème COM — Systèmes communicants
Projets Mascotte and Maestro

Rapport de recherche n° 6763 — October 2009 — 29 pages

Abstract: This work investigates distributed transmission scheduling in wireless networks. Due to interference constraints, "neighboring links" cannot be simultaneously activated, otherwise transmissions will fail. Here, we consider any binary model of interference. We follow the model described by Bui, Sanghavi, and Srikant in [5, 21]. We suppose that time is slotted and during each slot we have two phases: one control phase which determines what links will be activated and send data during the second phase. We assume random arrivals on each link during each slot, therefore a queue is associated to each link. Since nodes do not have a global knowledge of the network, our aim (like in [5, 21]) is to design for the control phase, a distributed algorithm which determines a set of non interfering links. To be efficient the control phase should be as short as possible; this is done by exchanging control messages during a constant number of mini-slots (constant overhead).
In this article we design the first fully distributed local algorithm with the following properties: it works for any arbitrary binary interference model; it has a constant overhead (independent of the size of the network and the values of the queues); and it needs no knowledge. Indeed contrary to other existing algorithms, we do not need to know the values of the queues of the "neighboring links", which are difficult to obtain in a wireless network with interference. We prove that this algorithm gives a maximal set of active links (in each interference set, there is at least one active edge). We also give sufficient conditions for

This work has been partially supported by région PACA, ANR AGAPE and DIMAGREEN, and European project IST FET AEOLUS.

* MASCOTTE, INRIA, I3S, CNRS, Univ. Nice-Sophia Antipolis, Sophia Antipolis, France.
† firstname.lastname@sophia.inria.fr
‡ MAESTRO, INRIA, Sophia Antipolis, France.
§ Dept. of Computer Science, Columbia University.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

stability under Markovian assumptions. Finally the performance of our algorithm (throughput, stability) is investigated and compared via simulations to that of previously proposed schemes.

Key-words: wireless network, transmission scheduling, interference, distributed algorithm, stability.

Algorithmes distribués d'ordonnancement des appels dans les réseaux sans-fil

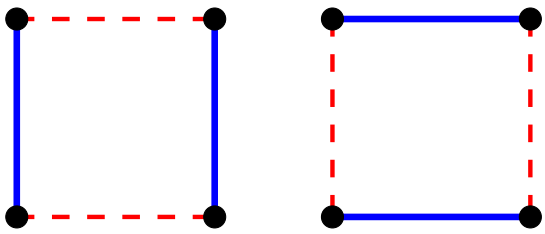
Résumé : Nous considérons dans cet article le problème d'ordonnancement distribué dans les réseaux sans-fil. En raison des interférences dans ce type de réseau, les liens "voisins" ne peuvent pas être activés simultanément, sinon les transmissions interfèrent. Nous considérons ici des modèles d'interférences binaires, comme ceux utilisés par Bui, Sanghavi et Srikant dans [5, 21]. Nous supposons que le temps est divisé en "slots" et que chaque slot comporte deux phases distinctes : une phase de contrôle qui détermine quels liens vont être activés et enverront des données durant la seconde phase. Nous supposons de plus que les arrivées de messages sur chacun des liens du réseau sont aléatoires. Comme les noeuds n'ont pas une connaissance globale du réseau, notre objectif (comme dans [5, 21]) est de concevoir pour la phase de contrôle, un algorithme distribué calculant un ensemble de liens n'interférant pas deux à deux entre eux. Pour être efficace, la phase de contrôle doit être aussi courte que possible; cela est réalisé par des échanges de messages de contrôle durant un nombre constant de "mini-slots" ("overhead" constant).
Dans cet article nous proposons le premier algorithme entièrement local vérifiant les propriétés suivantes : il est valable quelque soit le modèle d'interférence binaire utilisé; il a un overhead constant (indépendant de la taille du réseau et des valeurs des files d'attente associées aux liens du réseau); et il ne requiert pas de connaissance particulière de l'état du réseau. En effet contrairement aux algorithmes existants, nous n'avons pas besoin de connaître les valeurs des files d'attente des liens dans un "certain voisinage", une information difficile à obtenir dans un réseau sans-fil avec interférence. Nous prouvons que notre algorithme permet d'obtenir à chaque étape un ensemble maximal de liens actifs (dans chaque zone d'interférence, il y a au moins un lien activé). Nous donnons également des conditions suffisantes de stabilité sous des hypothèses markoviennes. Enfin les performances (débit, stabilité) de notre algorithme sont étudiées via des simulations, et comparées aux algorithmes existants.

Mots-clés : réseau sans-fil, ordonnancement des transmissions, interférences, algorithme distribué, stabilité.

1 Introduction

Transmission scheduling is a main problem in telecommunication networks. It has received a lot of attention both for wired and for wireless networks (radio, ad hoc, sensor network). In a wireless network a difficulty comes from interference problems. During a step or time slot, only transmissions which do not interfere can be scheduled. In this paper we consider any binary interference model. However for computations, examples, and simulations, we will use the so called d-interference model, where two transmissions which are within some interference distance d from each other interfere. In the particular case $d = 0$ (see [16]), we have the primary node interference model, where two transmissions interfere if their links intersect (one node can communicate with at most one another node); therefore, a set of transmissions can be activated together in the same time slot only if they form a matching in the corresponding undirected graph. The case $d = 1$ is known as the 802.11 interference model or distance-2 matching problem (see [1, 3, 14, 25]): here two links interfere if the first one contains a vertex adjacent to one of the second link. In that case a set of non interfering transmissions form an induced matching (see examples for $d = 0$ in Figure 3(a) and for $d = 1$ in Figure 3(b)).

The traffic is single-hop and the arrival process to each link of the network is assumed to be stochastic, with characteristics not necessarily known by the network designers. The goal is to schedule active links at each step in order to insure the stability of the system and, in particular, to activate links which are the most loaded. In the primary node interference model this corresponds to finding a maximum matching or a large matching. Centralized algorithms have been proposed to solve this problem both for random arrivals in [23, 24] and deterministic arrivals in [13]. As example, if the network is a square grid of 4 nodes (Figure 1) with the primary node interference model ($d = 0$), we can activate at one step either vertical links (Figure 1(a)) or horizontal links (Figure 1(b)). It is also possible to have a single active link in the network but we consider only the two previous sets of active links (maximal sets). A good scheduling algorithm has to insure the stability of the system (stability of the four queues associating to the four links in Figure 1). For example if the capacity of each link is 1 (if a link is active during a step, it sends 1 message), and if the average number of arrival messages per step is $A_v = \frac{2}{7}$ for the two vertical links and $A_h = \frac{4}{7}$ for the two horizontal links, we get a simple scheduling algorithm for the system. Indeed if we choose to activate vertical links 5 steps over 14 and activate horizontal links 9 steps over 14, we get trivially the stability of the queueing system because $A_h < \frac{9}{14}$ and $A_v < \frac{5}{14}$. Since arrival processes are not necessarily known a priori by network designers, thus in practice algorithms have to compute the set of active links as a function of current and past arrivals. However only distributed algorithms with limited local knowledge can be used. Indeed, centralized algorithms are based on a total knowledge of the network (i.e. link backlogs) at each step, an information which is very difficult to acquire because of the interference. Actually, acquiring this information is at least as much complicated



(a) Matching 1 (b) Matching 2

Figure 1: scheduling algorithm for grid for $d = 0$.

than solving the scheduling problem. In addition, for the sake of energy saving, decisions have to be made locally, without exchanging messages with a central station. In [4, 10, 18] distributed algorithms are described but they all lead to communication overheads which increase with the size of the network. In particular [12] presents a distributed algorithm valid for any binary interference model but at the expense of a non-constant overhead (increasing with the size of the network). The need for distributed algorithms with a small constant overhead has been emphasized in [21, 5], where a distributed algorithm with a constant overhead depending on the quality of the desired approximation is described; however it is only valid for the primary node interference model ($d = 0$). Recently, in [20] a randomized (aloha-like) contention resolution protocol was proposed that is conjectured to be throughput optimal.

It is worth noting that in the general case (i.e. interference model different from the primary node interference model) the problem of finding a set of links fulfilling the interference constraints and maximizing the sum of the weights (i.e. the total backlog on the links) is *NP-Complete*. For instance the problem of finding a maximum induced matching in a graph (case $d = 1$ in the d-interference model) is *NP-Complete* [22, 6] and it remains *NP-Complete* even for very special graphs (for 3-regular planar graphs for instance). Furthermore it is proved in [19] that the problem of finding a maximum induced matching in a graph cannot be approximated in polynomial time within a factor of $n^{\frac{1}{2}-c}$ for any constant $c > 0$ (unless $P = NP$), where n is the number of vertices in the graph (associating to the network).

Under these considerations, we make the following contributions to the general problem:

- We design the first, to our knowledge, distributed transmission algorithm which holds for *any* binary interference model while simultaneously having a communication overhead *independent* of the network size (Section 4).

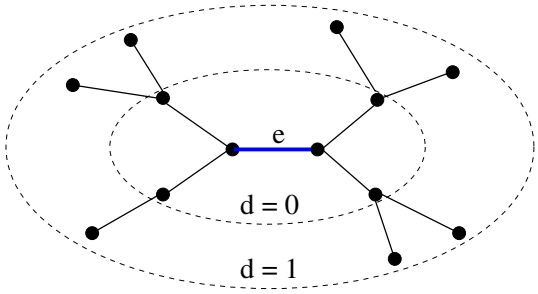


Figure 2: interference sets of e for $d = 0$ and $d = 1$.

- We formally prove properties of our algorithm and compute design tradeoffs. In particular, our overhead grows logarithmically with the *maximum degree*[1] of a node in the network and requires *no explicit knowledge* of queue size information between nodes (Section 5).
- We characterize the stability region of our algorithm under standard Markovian assumptions (Section 6) and validate our design via simulations (Section 7).
- Finally, we propose extensions to our algorithm in the multi-hop case (Section 8).

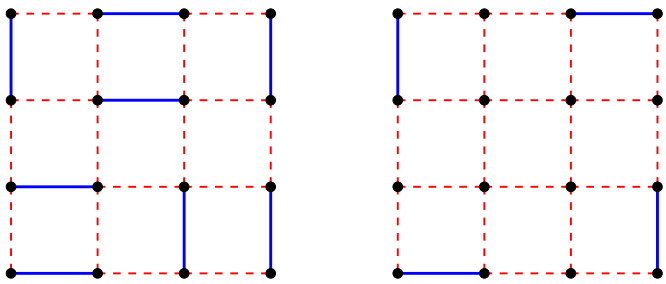
We begin by describing our model and some related work in the next two sections.

2 Modeling

We model the network by an undirected transmission graph $G = (V, E)$, where there is an edge (link) $e = uv \in E$ between two nodes $u \in V$ and $v \in V$ if they can communicate. We suppose that the relation is symmetric that is if u can transmit a packet to v, then v can transmit a packet to u and also that during a transmission both links are used as there are acknowledgment messages; so we have a symmetric directed graph; but for simplicity we use the undirected associated graph.

Interference can be modeled in different ways. We can use SINR (Signal-to-Interference-Noise Ratio) models [8] but that makes both the algorithms and the analysis hard. We choose to use a binary symmetric interference model like in [5, 12, 25]. We define the interference set of a link $e \in E$, denoted by $\varepsilon(e)$, as the set of edges interfering with e. Our algorithm is valid for any set $\varepsilon(e)$, but in the computations, examples, and simulations, we use a model based on an interference distance $d \geq 0$ as follows. Two edges interfere if one vertex of the first edge is

[1] From a practical standpoint, this presents a more attractive design tradeoff since node degree is bounded by physical layer characteristics whereas the network size can be unbounded.



(a) Matching (b) Induced matching

Figure 3: two sets of active (blue straight) links.

at distance in G at most d from a vertex of the second edge. More precisely, the interference set of an edge $e = (u_1, u_2) \in E$ is $\varepsilon(e) = \{(v_1, v_2) \in E, \exists i, j \in \{1, 2\}, d(u_i, v_j) \leq d\}$, where $d(u, v)$ is the distance between u and v in G, that is the length of a shortest path between these two nodes in G. Two examples of interference sets are represented in Figure 2 for the two interference models defined by $d = 0$ and by $d = 1$.

We suppose that time is slotted and synchronized in the network. Our algorithm consists, like in [5, 12], of a sequence of *steps* or *time slots*, all of the same size. One step contains two different phases: a *control phase* itself divided into mini-slots which consists of finding a valid set of active links for the sending (second) phase. We will say that a set of active links at step $t \geq 1$ is valid if the edges activated during this step do not interfere. The interference model defined by $d = 0$ is a particular case, commonly used, and it is known as the primary node (or node exclusive) interference model. In that case two edges interfere if they are incident and a valid set of active links, at some step, is a matching of the transmission graph G. The case $d = 1$, more realistic, is known as the 802.11 interference model. In that case two nodes can exchange data only if their neighbors are not involved in a communication. Said otherwise a valid set of active links is what is called an induced matching of G. Two examples of sets of active links are represented in Figure 3 for a square grid graph (network) of 16 nodes for interference models defined by $d = 0$ (Figure 3(a)) and defined by $d = 1$ (Figure 3(b)).

We introduce for each edge $e \in E$ of the transmission graph $G = (V, E)$ and for each step $t \geq 1$, the variable $a_t(e)$ such that $a_t(e) = 1$ if edge e is an active link, allowed to send data during step t ($a_t(e) = 0$ otherwise). The interference constraints imply that $\forall e, e' \in E$, if $a_t(e) = a_t(e') = 1$, then $e' \notin \varepsilon(e)$ (and $e \notin \varepsilon(e')$). Otherwise there are interference problems and so messages are not received correctly.

Traffic is dynamic and is assumed in this article to be single-hop, like in [5], that is to say one packet sent through a communication link leaves the network just after. However we can deal with multi-hop traffic with few changes in our algorithm (see Section 8). To each edge (link) $e \in E$ is associated a queue. We denote by $q_t(e)$ the number of messages in the queue of link $e \in E$ at the beginning of step $t \geq 1$. We will say that the weight of e at step $t \geq 1$ is $q_t(e)$. At each step, new packets arrive in the links of the network (edges of the transmission graph) according to some arrival processes. These ones are specific to each link $e \in E$ and not necessarily known by the designers of the network. Let $A_t(e)$ be the number of packets arriving in edge $e \in E$ during step $t \geq 1$. The capacity of an edge $e \in E$, denoted by $c(e)$, is the number of packets that e can serve during one time slot $t \geq 1$ if the link is active (that is if $a_t(e) = 1$). Thus we have $q_{t+1}(e) = (q_t(e) - a_t(e)c(e))^+ + A_t(e)$ with $(y)^+ = max(y, 0)$.

3 Previous Works

We briefly describe in this section two algorithms respectively proposed in [12] and [5] because we use the same modeling. They are both distributed but the algorithm in [12] does not admit a constant overhead, whereas the one described in [5] is valid only for the primary node interference model ($d = 0$).

In the algorithm described in [12], for each step $t \geq 1$ there are a control phase and a data phase, like in our modeling (Section 2). Before each control phase, composed of T mini-slots, each edge $e \in E$ is undetermined, and chooses a backoff value $t(e)$, $1 \leq t(e) \leq T$. In this context undetermined means that the edge does not know if it will be active or inactive during the data phase of the current step. If edge e receives a message (from an edge in $\varepsilon(e)$) during a mini-slot t, $1 \leq t \leq t(e) - 1$, then e is inactive. Otherwise e sends a control message during mini-slot $t(e)$ and if it does not receive a message, then e is active (e is inactive otherwise). At the end, a valid set of active links is computed, allowed to send messages during the data phase. This simple algorithm is valid for any interference set, but the choice of the backoff value $t(e)$, for any edge $e \in E$, is a function of the weights of edges located in its interference set $\varepsilon(e)$ and function of the weights of edges located in interference set of each $e' \in \varepsilon(e)$. More precisely $t(e)$ depends on the weights of edges belonging to the following set $S = \{e', e' \in \varepsilon(e)\} \cup \{e'', e'' \in \varepsilon(e'), e' \in \varepsilon(e)\}$. Thus at each time slot $t \geq 1$, each edge has to update the weights of edges located in its $2d - neighborhood$ (edges at distance $\leq 2d$ from e), if we have an interference model based on distance d. Therefore the overhead is not constant and furthermore one has to obtain this information, which, due to interference, is a problem as difficult as the transmission scheduling problem.

The algorithm described in [5], `Augmenting Paths Algorithm`, has a constant overhead but it is specific to the primary node interference model ($d = 0$). It uses the "augmenting path tool" developed for matching theory [15] and which is used to find polynomial centralized algorithms to determine maximum matchings. In that case, at each time slot $t \geq 1$,

a valid set of active links is a matching of the transmission graph $G = (V, E)$. The main idea of `Augmenting Paths Algorithm`, is to compute, at a step $t + 1 > 1$, a matching of G from the matching of G found at step t. In [5], it is proved that, if $2k + 1$ is the maximum length of the augmenting paths, the algorithm needs a constant overhead of order $4k + 2$ and achieves $\frac{k}{k+2}$ of the capacity region (we can improve via a careful analysis this value to $\frac{k}{k+1}$).

Moreover at the beginning of each time slot $t \geq 1$, each node $v \in V$ becomes *seed* with a constant probability p. A seed is a node allowed to start an alternating path. In [5], it is not described how to compute p analytically.

Finally it is necessary to precise that `Augmenting Paths Algorithm` is specific to the primary node interference model ($d = 0$), not the more realistic one. Indeed the augmenting paths technical is a property of the notion of matching in graphs. Recall that the problem of determining a maximum matching (if the interference model is defined by $d = 0$) can be done in polynomial time with a centralized algorithm, but for interference models defined by $d \geq 1$, determining a maximum valid set of active edges is an NP-complete problem. In Section 7.2.2 we compare via simulations the performance of `Augmenting Paths Algorithm` to the performance of our distributed algorithm, proposed in Section 4, for a square grid composed of 121 nodes.

4 Our Distributed Algorithm

We propose in this section a distributed link scheduling algorithm, valid with a constant communication overhead, whereas the algorithm proposed in [12] has an overhead increasing with the size of the network, and valid for any binary interference set, whereas `Augmenting Paths Algorithm`, proposed in [5], is valid only for the primary node interference model ($d = 0$). Furthermore algorithms proposed in the literature like those in [5, 12] require exchange of weights between some edges (the acquisition of this information is, due to interference, a difficult and costly problem and needs itself a control phase). Here, in contrary, during each mini-slot of the control phase, any edge either sends a control message composed of 1 bit or nothing. The main idea of `Algorithm Log` is to use interference as information. More precisely during each mini-slot the only important thing for an edge $e \in E$ is to determine if an edge $e' \in \varepsilon(e)$ has sent a control message through this mini-slot. Based on that and on its own bit the edge will decide to be active or inactive or postpone the decision to another mini-slot. As two edges $e \in E$ and $e' \in \varepsilon(e)$ do not exchange their respective weights during the control phase, the number and size of mini-slots, and therefore the overhead, are minimized.

As previously described in Section 2, the network is modeled by an undirected transmission graph $G = (V, E)$ and time is slotted into steps of same size. At any step $t \geq 1$, a set of active links is chosen through a control phase and during the data phase, these links send their messages. We precisely describe in this section, the control phase for any time

slot $t \geq 1$. We propose a distributed algorithm, `Algorithm Log`, which will compute a valid set of active links at step t, with a sum of weights of edges belonging to this set as large as possible. Recall that the weight $q_t(e)$ of an edge $e \in E$ is the number of messages in the queue associated to this link (Section 2). Finally only edges with weight greater than their capacity will participate to `Algorithm Log`, that is if $\frac{q_t(e)}{c(e)} \geq 1$. We call this set of edges E_1.

Another idea consists in associating to each edge a virtual weight (see Section 4.1) in such a way that two edges interfering have different virtual weights (Lemma 1). Then each edge computes the binary control vector associated to the virtual weight; the bits 1 will indicate the mini-slots when the edge will send a control message if it is not yet inactive (Section 4.2). In Section 4.3 we describe precisely `Algorithm Log`, before giving two examples in Section 4.4. We then analyze `Algorithm Log` in Section 5.

4.1 Virtual Weights

The idea consists in splitting the values of the queues into a small number K of disjoint intervals (classes) $I_0, \ldots, I_{K-1}$ with the constraint that the largest class I_{K-1} contains all the values greater than some value L: $I_{K-1} =]L, \infty[$. As K will be small, many edges interfering might belong to same interval. To differentiate them we also give to each edge a color (integer) $\gamma(e)$ in such a way that two edges interfering have different colors. Let C be the number of colors. For example if G is a path, in the d-interference model, we can use only $d+2$ colors by giving to the edges values $1, 2, \ldots, d+2, 1, 2, \ldots, d+2, \ldots$ Figure 4(b) shows a coloring for a cycle of length 9 with $C = 3$ within the primary node interference model ($d = 0$). We will see in Section 5.3 how to determine C. This coloring will give a ranking between the edges in a same interval. In order to get different rankings in consecutive slots, we will permute the colors. For that we associate in the slot t the value $\gamma_t(e) = (\gamma(e) + t - 2)_{modC} + 1$. From definition of $\gamma(e)$, $\gamma_t(e) \neq \gamma_t(e')$ for each $e \in E$, $e' \in \varepsilon(e)$, $t \geq 1$. For example, with $C = 3$ an edge with color 1 will get a value 1 at slot 1, 2 at slot 2, 3 at slot 3, 1 at slot 4, 2 at slot 5 and so on.

Now we can assign to each edge $e \in E$ a virtual weight $q'_t(e)$ as follows. Let f be the mapping $f : [1, \infty[\rightarrow \{0, \ldots, K-1\}$ defined by $f(x) = i$ if $x \in I_i$ for $i = 0, 1, \ldots, K-1$. Then

$$q'_t(e) = Cf\left(\frac{q_t(e)}{c(e)}\right) + \gamma_t(e) \tag{1}$$

See Figure 4 and Figure 5 for examples of computations. Note that

$$0 < q'_t(e) \leq CK \quad \forall e \in E_1,\, t \geq 1. \tag{2}$$

The discussion of the choice of constants C, K and L is deferred to Section 5.3 and Section 5.4.

The motivation for introducing these virtual weights is threefold. First, and most importantly, virtual weights have the property that two interfering links have different virtual

weights (see Lemma 1 below). This property will be a key ingredient for ensuring that our scheduling algorithm determines a maximal set of active links in each time slot (see Section 5.1). Second, the scaling in (1) (i.e. the fact that $q_t(e)$ is divided by $c(e)$) will ensure that our algorithm will work with absolute delays in terms of time and not favor links with high capacities. Finally our algorithm will use the ranking induced by the virtual weights to determine the active edges; an edge with a high virtual weight and in particular with a high queue will have more chances to be chosen.

Lemma 1. *Let $e, e' \in E$ be two edges in E_1 such that $e' \in \varepsilon(e)$ (and $e \in \varepsilon(e')$). Then, $q'_t(e) \neq q'_t(e')$.*

Proof. If $\frac{q_t(e)}{c(e)}$ and $\frac{q_t(e')}{c(e')}$ belong to two different intervals, that is $f(\frac{q_t(e)}{c(e)}) \neq f(\frac{q_t(e')}{c(e')})$, then $|C(f(\frac{q_t(e)}{c(e)}) - f(\frac{q_t(e')}{c(e')}))| \geq C$ and as $1 \leq \gamma_t(e), \gamma_t(e') \leq C$, $q'_t(e) \neq q'_t(e')$. Otherwise, if $f(\frac{q_t(e)}{c(e)}) = f(\frac{q_t(e')}{c(e')})$, then by the choice of $\gamma(e)$ and $\gamma(e')$, $\gamma_t(e) \neq \gamma_t(e')$ and so $q'_t(e) \neq q'_t(e')$. □

4.2 Control Vector

We assign to each edge $e \in E_1$, a *control vector* $v_{t,e} = (v_{t,e}(1), \ldots, v_{t,e}(T))$ where $v_{t,e}(i)$, $1 \leq i \leq T$, corresponds to the ith bit of $q'_t(e)$. Recall that $q'_t(e) \leq CK$ for each $e \in E$, and so $T = \lceil log_2(CK) + 1 \rceil$. Remark that an edge $e \notin E_1$ (that is not participating to the control phase) can be considered as having a virtual weight $q'_t(e) = 0$ and so having a control vector $v_{t,e}$ only composed of 0, and we do not consider anymore this kind of edges in our algorithm (we could have included them with this control vector but they will never become active). In Table 1, all possible virtual weights and corresponding control vectors of an edge $e \in E_1$ are represented for each possible pair $(q_t(e), \gamma_t(e))$ according to the following parameters: $d = 0$ (primary node interference model), $C = 3$, $K = 5$, and for example $L = 4$. It corresponds to a cycle graph with an odd number of nodes. An example of such a graph $G = (V, E)$ is depicted in Figure 4(b), where values located on edges correspond to a valid assignment of $\gamma(e)$ for each $e \in E$. Indeed, any such assignment requires at least 3 different integers, and so $C \geq 3$ (see Section 5.3). Remark that for a cycle with an even number of nodes, 2 integers are sufficient to assign $\gamma(e)$ for each edge $e \in E$. We could have chosen another value for L for example $L = 100$ and take as intervals $I_4 =]100, \infty[$ and for example $I_0 = [1, 10]$; $I_1 =]10, 30]$; $I_2 =]30, 60]$; $I_3 =]60, 100]$ or any other splitting of the values of the queues. The control vector is the same as that given.

4.3 Algorithm Log

Given the following inputs : the binary interference model, K, L, and a valid integer C respecting previous constraints, `Algorithm Log` will compute a maximal set of active edges.

Table 1: $(q'_t(e), v_{t,e})$ for every possible pair $(q_t(e), \gamma_t(e))$ for `Algorithm Log` ($C = 3$, $K = 5$, $L = 4$).

			Control Vector: $v = v_{t,e}$			
$f(\frac{q_t(e)}{c(e)})$	$\gamma_t(e)$	$q'_t(e)$	$v(1)$	$v(2)$	$v(3)$	$v(4)$
1	1	1	0	0	0	1
1	2	2	0	0	1	0
1	3	3	0	0	1	1
2	1	4	0	1	0	0
2	2	5	0	1	0	1
2	3	6	0	1	1	0
3	1	7	0	1	1	1
3	2	8	1	0	0	0
3	3	9	1	0	0	1
4	1	10	1	0	1	0
4	2	11	1	0	1	1
4	3	12	1	1	0	0
≥ 5	1	13	1	1	0	1
≥ 5	2	14	1	1	1	0
≥ 5	3	15	1	1	1	1

At each mini-slot of `Algorithm Log`, an edge can be active, inactive, or still undetermined. All the edges $e \in E_1$ active at the end of `Algorithm Log` will send data after. The control phase is itself divided into α subphases. We first explain subphase 1 before motivating and describing subphases $2, \ldots, \alpha$. We will see in Section 5.1 how to determine α.

4.3.1 Subphase 1

Let us describe subphase 1 composed of T mini-slots. Before mini-slot 1, each edge is undetermined. During a mini-slot i, $1 \leq i \leq T$, $e \in E$ sends a control message if and only if e is still undetermined and $v_{t,e}(i) = 1$. If e is still undetermined at the beginning of mini-slot i, $1 \leq i \leq T$, we have the following three cases (see lines 4-14 of Algorithm 1):

(a) if e sends a control message and e does not receive one from an edge in $\varepsilon(e)$, then e becomes **active**;
(b) if e receives a control message (therefore coming from an edge in $\varepsilon(e)$) and does not send a control message, then e becomes **inactive**;
(c) otherwise e remains **undetermined**.

At the end of subphase 1 (after T mini-slots), we get a valid set of active links (edges), respecting interference constraints. For example if the interference model is defined by $d = 0$ (primary node interference model), we get a matching of G. If the interference model is defined by $d = 1$, we get an induced matching of G. Furthermore at the end of subphase 1, an edge $e \in E$ is either active or inactive (see Lemma 2).

Lemma 2. *At the end of subphase 1 of* `Algorithm Log`*, there is no undetermined edge.*

Proof. From Lemma 1, $q'_t(e) \neq q'_t(e')$ for any $e, e' \in E$ such that $e' \in \varepsilon(e)$. Suppose e is undetermined at the end of subphase 1 of `Algorithm Log`. During the last mini-slot where e sends a control message (mini-slot T if $q'_t(e)$ is odd, mini-slot $T - 1$ if $q'_t(e)$ is even), it stayed undetermined if another edge $e' \in \varepsilon(e)$ sends a message during this mini-slot. But as $q'_t(e) \neq q'_t(e')$ there is a preceding mini-slot where the edge with the biggest weight was sending a control message and the other not; but the edge with the smallest weight would have become inactive at this time slot, a contradiction. □

Definition 3. *An edge $e \in E_1$ is a local maximum edge at step $t \geq 1$ if $\forall e' \in \varepsilon(e)$, $q'_t(e') \leq q'_t(e)$. Remark that if an edge is maximum then we have the strict inequality from Lemma 1.*

From Lemma 2, it follows that a maximum local edge e is necessarily active at the end of subphase 1.

Corollary 4. *A local maximum edge $e \in E$ is always active at the end of subphase 1 of* `Algorithm Log`*.*

Thus edges with large weights (in a local point of view) are chosen as active and belong to the set of active links. Recall that one of the main objectives of transmission scheduling algorithms is to insure stability of queues associated to links, and so if the set of active links contains edges with large weights, queues with many messages decrease.

After subphase 1, it might happen that the set of active links was not maximal (edges can be added to this set without removing current edges in this one). For instance Figure 4 describes subphase 1 of `Algorithm Log` for a cycle composed of 9 edges with $d = 0$. For such a network it is the worst case in terms of cardinality of the set of active links (matching of G). Indeed after subphase 1, only the central link is active. One possibility is to add extra mini-slots to get more active links with a random process. Another solution is to repeat this subphase $\alpha - 1$ times, and so get α subphases in `Algorithm Log`. We describe it in Section 4.3.2.

4.3.2 Other Subphases

As described in Section 4.3.1, it might happen that the set of active edges was not maximal after subphase 1. To deal with this problem, we apply $\alpha - 1$ times the protocol described for

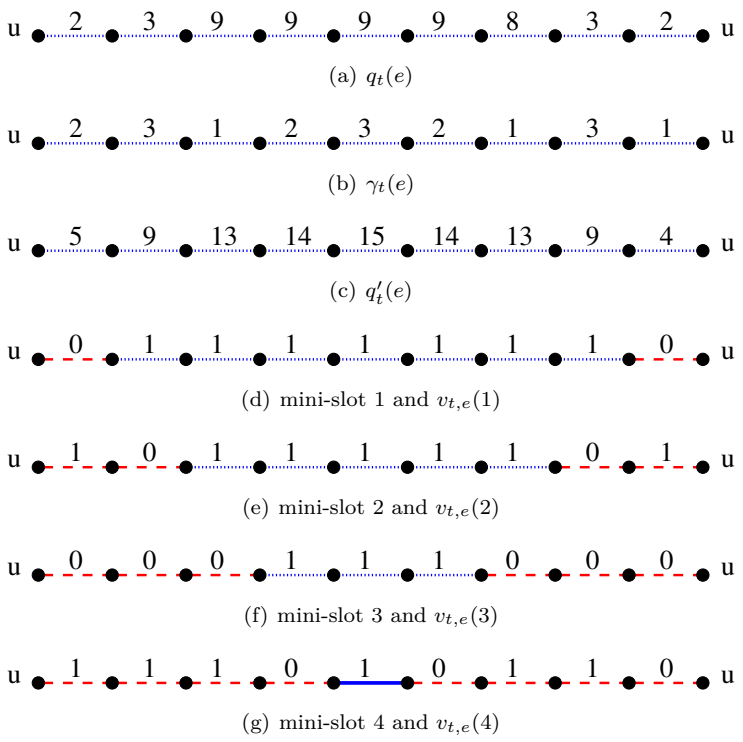


Figure 4: subphase 1 of `Algorithm Log` for a cycle of 9 edges ($d = 0$, $C = 3$, $K = 5$, $L = 4$).

subphase 1, but only with edges which can be added that is inactive edges without active links in their interference sets. After each subphase each edge is either active or inactive (Lemma 2). So, we add a re-initialization mini-slot after each subphase j, $1 \leq j \leq \alpha - 1$, where each active edge sends a control message and inactive edges become undetermined if they do not receive any message (lines 15-20 of Algorithm 1). The next subphase will involve only those undetermined edges. In Theorem 5 of Section 5.1, we will see that, if we fix the value of the parameter α to be T, we insure that at the end of each control phase, `Algorithm Log` always computes a maximal set of active edges in E_1.

Finally we define `Algorithm Log` composed of α subphases as described above. A formal algorithm for each edge $e \in E$ is described in Algorithm 1, where $s(e) = 1$ if e is active, $s(e) = 0$ if e is undetermined, and $s(e) = -1$ if e is inactive.

Algorithm 1 Distributed Algorithm of an edge $e \in E$

Require: $q_t(e)$, $\gamma_t(e)$.
Ensure: return *active* or *inactive*

```
e computes q'_t(e) and v_{t,e}
s(e) = 0
for j = 1, ..., α do
  for i = 1, ..., T do
    if s(e) = 0, and v_{t,e}(i) = 1 then
      e sends a message (to edges in ε(e))
      if e does not receive a message then
        s(e) = 1
      end if
    end if
    if s = 0, v_{t,e}(i) = 0, and e receives a message then
      s(e) = −1
    end if
  end for
  if s(e) = 1 then
    e sends a message (to edges in ε(e))
  end if
  if s(e) = −1, and e does not receive a message then
    s(e) = 0
  end if
end for
if s(e) = 1 then
  return active
else
  return inactive
end if
```

4.4 Examples

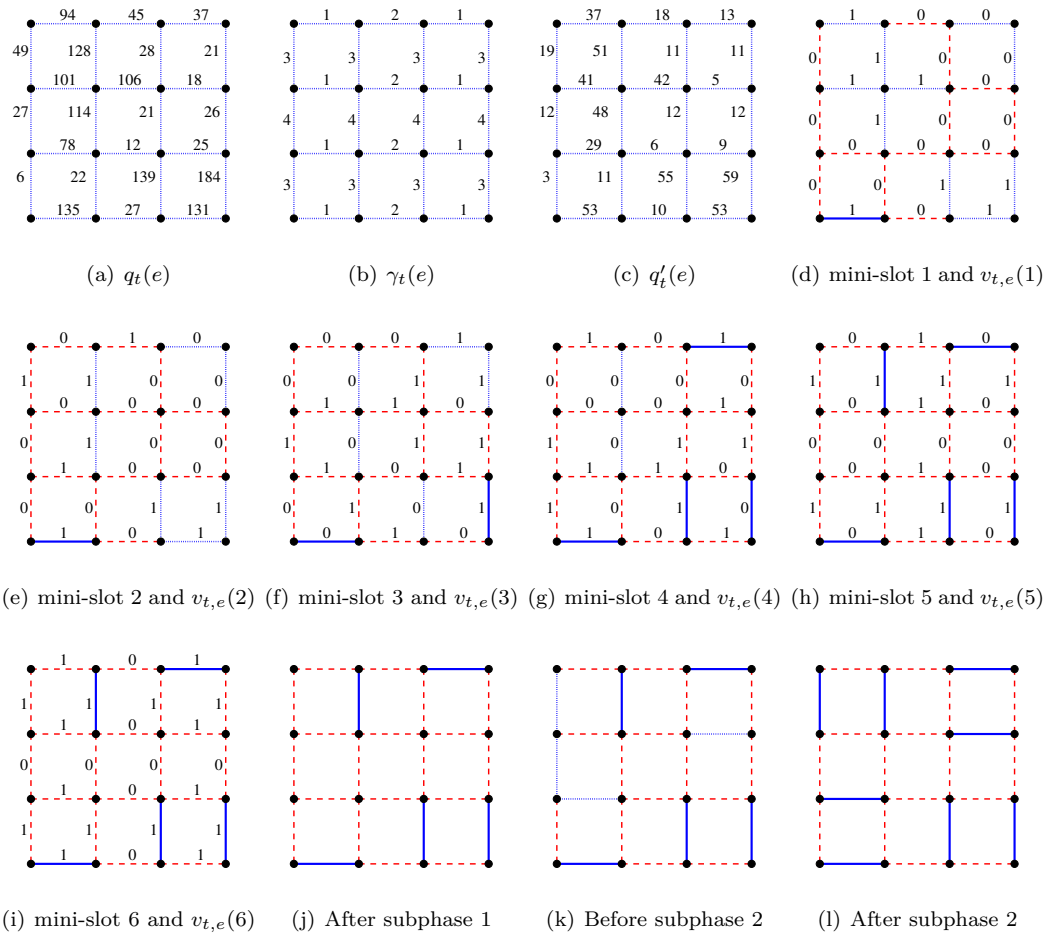
We present here two examples of applications of `Algorithm Log`. In Section 4.4.1 `Algorithm Log` is applied to a square grid graph with the interference model defined by $d = 0$ and in Section 4.4.2 `Algorithm Log` is applied to a graph (generated randomly as in Section 7.2.3) with the interference model defined by $d = 1$.

4.4.1 Square Grid with d = 0

Let $G = (V, E)$ be a square grid graph composed of $|V| = 16$ nodes and $|E| = 24$ edges (Figure 5). The interference model is defined by $d = 0$, and so a valid set of active links is a matching of G. In our example the smallest C respecting constraints described in Section 4.1 is 4 (See Figure 5(b) for an example of such optimal assignment of $\gamma(e)$ for each $e \in E$ and see Section 5.3 for details). In this example $K = 15$ and $L = 140$. Given a time slot $t \geq 1$, the weight $q_t(e)$, the value $\gamma_t(e)$, and the virtual weight $q'_t(e)$ are respectively represented in Figure 5(a), in Figure 5(b), and in Figure 5(c) for each edge $e \in E$. For example for the top-left edge $e \in E$ with weight $q_t(e) = 94$ and $\gamma_t(e) = 1$, $q'_t(e) = 37$ using Equation 1. Thus the associated control vector $v_{t,e}(e) = (1, 0, 0, 1, 0, 1)$. Let us describe subphase 1 of `Algorithm Log` composed of 6 mini-slots. At the beginning (before mini-slot 1), each link is undetermined. During mini-slot 1, each edge e such that $v_{t,e}(1) = 1$ sends a control message and:

(a) if e sends a control message and e does not receive one from an edge in $\varepsilon(e)$, then e becomes **active**;
(b) if e receives a control message (therefore coming from an edge in $\varepsilon(e)$) and does not send a control message, then e becomes **inactive**;
(c) otherwise e remains **undetermined**.

Figure 5(d) shows the state of the edges after mini-slot 1. For example the horizontal edge on the bottom-left (with $q'_t(e) = 53$) becomes active. Furthermore there are three connected components of undetermined edges, respectively composed of 5, 3, and 2 edges. Repeating the same protocol during mini-slot 2, there are three connected components of undetermined edges, respectively composed of 2, 3, and 2 edges (Figure 5(e)). After mini-slot 3, a second edge becomes active and there are three connected components of undetermined edges, respectively composed of 2, 1, and 2 edges (Figure 5(f)). After mini-slot 4, there are two additional active links, and there is one connected component of undetermined edges composed of 2 edges. After mini-slot 5, one more edge becomes active and there is no undetermined edge (Figure 5(g)). Thus after $T = 6$ mini-slots (after subphase 1 of `Algorithm Log`), a set of active links is found. But it is easy to show that this matching is not maximal. Thus at the beginning of subphase 2, inactive edges without active link in their interference sets become again undetermined (it is possible with one re-initialization mini-slot at the end of the previous subphase). We repeat the same protocol than before, and in this example we get a maximal set of active links (maximal matching) after the subphase 2. From Theorem 5 of Section 5.1, $\alpha = T$ subphases is sufficient to insure a maximal set of active links ($\forall d \geq 0$).



(a) $q_t(e)$ (b) $\gamma_t(e)$ (c) $q'_t(e)$ (d) mini-slot 1 and $v_{t,e}(1)$

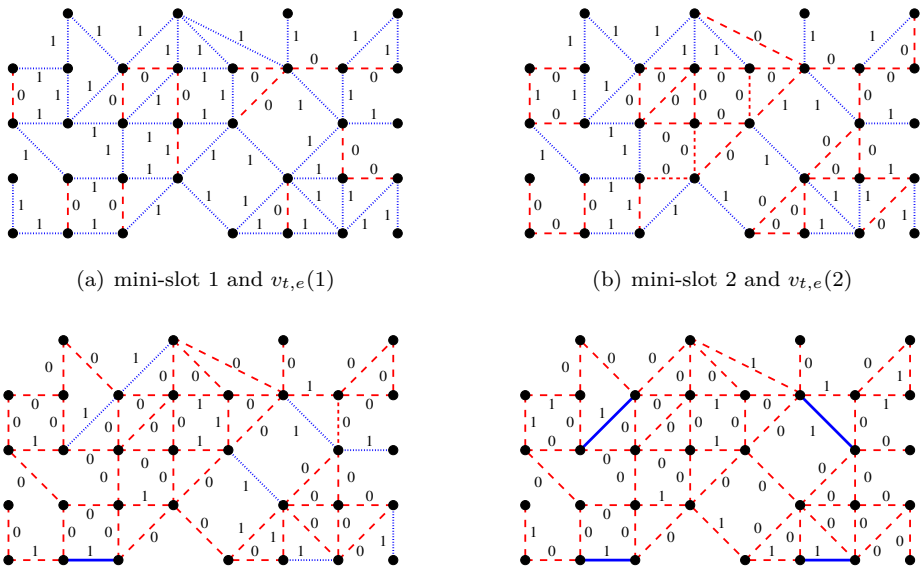
(e) mini-slot 2 and $v_{t,e}(2)$ (f) mini-slot 3 and $v_{t,e}(3)$ (g) mini-slot 4 and $v_{t,e}(4)$ (h) mini-slot 5 and $v_{t,e}(5)$

(i) mini-slot 6 and $v_{t,e}(6)$ (j) After subphase 1 (k) Before subphase 2 (l) After subphase 2

Figure 5: `Algorithm Log` for a grid ($d = 0$, $C = 4$, $K = 15$, $L = 140$).

4.4.2 Random Graph with d = 1

Let $G = (V, E)$ be a random graph composed of $|V| = 33$ nodes and $|E| = 56$ edges (Figure 6). G has been generated using method described in Section 7.2.3. The interference model is defined by $d = 1$, and so a valid set of active links is an induced matching of G. Figure 6 shows the four first mini-slots of subphase 1 of `Algorithm Log`. We get after mini-slot 4 a maximal induced matching and so the set of active links for this time slot.



(a) mini-slot 1 and $v_{t,e}(1)$

(b) mini-slot 2 and $v_{t,e}(2)$

(c) mini-slot 3 and $v_{t,e}(3)$

(d) mini-slot 4 and $v_{t,e}(4)$

Figure 6: four first mini-slots of subphase 1 of `Algorithm Log` for a random graph with $d = 1$.

5 Analysis

We analyze in this section `Algorithm Log`, fixing the value α to ensure that for any time slot $t \geq 1$, the set of active links is maximal in Section 5.1, computing the overhead (number of mini-slots of the control phase) in Section 5.2, and finally describing how to choose the different parameters C, K, L of `Algorithm Log` in Section 5.3 and in Section 5.4.

5.1 Maximality

After the control phase of `Algorithm Log` for any step $t \geq 1$, we prove that the set of active links is always maximal,if we choose $\alpha = T = \lceil log_2(CK) + 1 \rceil$.

Theorem 5. *If we choose $\alpha = T = \lceil log_2(CK) + 1 \rceil$ in `Algorithm Log`, then for all $e \in E_1$ (set of edges with positive virtual weight), there exists one edge $e' \in \varepsilon(e) \cup \{e\}$ such that e' is active at the end of `Algorithm Log`.*

Proof. Let $T = \lceil log_2(CK) + 1 \rceil$. Let e_1 be any edge of E_1. Either e_1 is active at the end of subphase 1 of `Algorithm Log`, and the theorem is proved. Or e_1 is inactive due to an edge $e_2 \in \varepsilon(e_1)$ (with $q'_t(e_2) > q'_t(e_1)$) which has sent a control message at some mini-slot t_1. Either e_2 is active and the theorem is proved or e_2 is inactive due to an edge $e_3 \in \varepsilon(e_2)$ which has sent a control message at some mini-slot $t_2 > t_1$, and so on. Let k be the largest index of a sequence of inactive edges $e_1, e_2, \ldots, e_k$ such that edge e_i $(1 \leq i \leq k)$ is inactive due to $e_{i+1} \in \varepsilon(e_i)$ which has sent a control message at mini-slot t_i and edge e_{k+1} is active. As $T \geq t_k > \ldots > t_i > t_{i-1} > \ldots > t_1 \geq 1$, we have $k \leq T$.

We will now prove by induction that, at the end of the subphase j of `Algorithm Log`, the longest sequence of edges $e_1, e_2, \ldots, e_{k_j}$ such that edge e_i $(1 \leq i \leq k_j)$ became inactive due to $e_{i+1} \in \varepsilon(e_i)$ and edge e_{k_j+1} is active, satisfies $k_j \leq T - j + 1$. As we have seen, that is true for $j = 1$. Suppose it is true till $j - 1$. Note that e_{k_j+1} was not active at the end of subphase $j - 1$, otherwise e_{k_j} would have been definitively inactive at the end of the subphase $j - 1$ and would not participate to the subphase j . As the edges $e_1, e_2, \ldots, e_{k_j}$ are also inactive at the end of subphase $j - 1$, we have a sequence of inactive edges of length $k_j + 1$ and by induction hypothesis the length of this sequence satisfies $k_j + 1 \leq T - (j - 1) + 1$ and so $k_j \leq T - j + 1$. Therefore, for some $j_0 \leq T$, $k_{j_0} \leq 1$ implying that at the end of the subphase j_0 any edge e is either active or is inactive due to an active edge $e' \in \varepsilon(e) \cup \{e\}$. □

5.2 Overhead (Complexity)

We compute here the number of mini-slots required for the control phase of `Algorithm Log` according to K and C.

Theorem 6. *The number of mini-slots of the control phase of* `Algorithm Log` *is* $T_{log} = \lceil log_2(CK) + 1 \rceil^2 + \lceil log_2(CK) + 1 \rceil - 1$.

Proof. Subphase 1 of `Algorithm Log` is composed of T mini-slots where $T = \lceil log_2(CK) + 1 \rceil$. From Theorem 5, $\alpha = T$ subphases are sufficient to insure maximality. Furthermore at the end of subphase i, $1 \leq i \leq T - 1$, one re-initialization mini-slot is needed. Indeed each active edge sends a control message and some inactive edges become undetermined if they do not receive any message. Thus $T - 1$ extra mini-slots are added. Finally $T_{log} = T^2 + T - 1$. □

5.3 How to Compute the Constant C?

In order to minimize the overhead of `Algorithm Log` (number of mini-slots of the control phase), we have to minimize the constant C, insuring that $\gamma(e) \neq \gamma(e')$ and $1 \leq \gamma(e), \gamma(e') \leq C$ for all $e, e' \in E$ such that $e' \in \varepsilon(e)$ (and so $e \in \varepsilon(e')$). Recall that if $\gamma(e) \neq \gamma(e')$, then $\forall t \geq 1, \gamma_t(e) \neq \gamma_t(e')$ (Section 4.1).

Given a graph $G = (V, E)$, our problem consists in assigning an integer (a color) $\gamma(e)$ to each edge $e \in E$ such that the integers assigned to two edges $e, e' \in E$ with $e \in \varepsilon(e')$ are

different, minimizing the total number of integers (colors) used to do it. For our problem, the integer (color) assigned to an edge e corresponds to $\gamma(e)$ and the minimum total number of colors needed corresponds to the minimum C respecting previous constraints. There exists a simple greedy algorithm to do it in linear time using $2\Delta(G)^{d+1}$ different integers (colors) [9] where $\Delta(G)$ is the maximum degree in G with the interference model defined by d.

A particular case is when the binary interference model is defined by $d = 0$. In that case, the problem of minimizing C remains to compute the *Edge Chromatic Number*, denoted by $\chi'(G)$, of G (see [11]). Theorem 7 proves a better bound to the optimal number of colors needed. A proof of this theorem can be found in [11] and a constructive proof in [17]. Finally there exists a polynomial time algorithm to get a (+1)-approximation for this particular problem. More precisely this algorithm computes a valid coloring of the edges using at most $\Delta(G) + 1$ colors and in the worst case, the optimal value is $\Delta(G)$ colors.

Theorem 7 (Vizing's Theorem)**.** *Given a graph $G = (V, E)$, $\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$ where $\Delta(G)$ is the maximum degree in G.*

Furthermore, if G is bipartite, $\chi'(G) = \Delta(G)$, and in particular for the grid $\chi'(G) = 4$.

5.4 How to Choose K and L?

If the number T_{log} of mini-slots allowed is constant and fixed, it is possible to compute the maximum value of K respecting this constraint. More precisely we choose the largest K such that $\lceil log_2(CK)+1\rceil^2+\lceil log_2(CK)+1\rceil-1 \leq T_{log}$ (C is a constant previously computed by algorithms described in Section 5.3). Furthermore we choose L large enough. Recall that the number of mini-slots T_{log} of `Algorithm Log` is constant when L increases.

6 Stability

We analyze the stability of `Algorithm Log` in this section. Note that since `Algorithm Log` produces a maximal match in every timestep, it is guaranteed to achieve a fraction of the maximum throughput capacity in an arbitrary wireless network [7]. However, the capacity region for `Algorithm Log` is larger than this minimum bound, and we now proceed to calculate it.

Let $A_t(e) \in \{0, 1, \ldots\}$ be the number of arrivals on link e in slot t.

Recall that $q_t(e)$ is the number of packets waiting to be transmitted on link e (called queue e) at the beginning of slot t.

For the sake of simplicity we assume that arrivals on link $e \in E$ during slots $sC+1, sC+2, \ldots, (s+1)C$ ($s = 0, 1, \ldots$) join queue e at the end of slot $(s+1)C$.

Let $B_s(e) := \sum_{t=sC+1}^{(s+1)C} A_t(e)$ be the cumulated number of arrivals on link $e \in E$ during slots $sC+1, sC+2, \ldots, (s+1)C$. We have

$$q_{(s+1)C}(e) = q_{sC+1}(e) - x_{sC+1}(e) + B_s(e), \quad e \in E,$$

where $x_{sC}(e) \in [c(e), q_{sC+1}(e)]$ is the number of packets transmitted on link e during slots $sC+1, sC+2, \ldots, (s+1)C$. The requirement that $x_{sC}(e) \geq c(e)$ is a consequence of the property that `Algorithm Log` activates a link only if its backlog is at least equal to the link capacity (see Section 4.1).

Assumption 1. *For each $e \in E$, $\{B_s(e), s \geq 0\}$ is an independent and identically distributed (iid) sequence of random variables. Furthermore $\{B_s(e), s \geq 0\}$, $e \in E$, are mutually independent sequences.*

Assumption 1 and the definition of the virtual weights in Section 4.1 imply that $\mathbf{X} := \{((q_{sC}(e), \gamma_{sC}(e)), e \in E), s \geq 0\}$ is a Markov process with state-space $\mathbf{N}^{|E|} \times \{1, \ldots, C\}$, where C is the number of colors needed so that two interfering edges have different colors.

We will assume that this Markov chain is irreducible. Under `Algorithm Log` this property will hold, in particular, if $P(A_t(e) = k) > 0$ for all k.

The stability of `Algorithm Log` is defined as the stability of the Markov chain $\mathbf{X}$. In words, when `Algorithm Log` is stable then queues "will not build up".

Let $\{\mathbf{I}_j\}_j$ be the set of all admissible schedules. A schedule $\mathbf{I}_j = (I_j(e), e \in E)$ is a binary vector where $I_j(e) = 1$ if link e is active and $I_j(e) = 0$ otherwise. A schedule is admissible if it will not produce any interference among active links.

Define the so-called *capacity region* $\mathcal{C}$ [24] (with $\lambda_j > 0$ for all j)

$$\mathcal{C} = \left\{ \mathbf{v} = \left(\sum_j \lambda_j \mathbf{I}_j \right) \mathbf{M} \text{ with } \sum_j \lambda_j < 1 \right\}.$$

with $\mathbf{M} = \mathrm{diag}(c(e), e \in E)$.

Let $\mathbf{c} = (c(e), e \in E)$ be the link capacity vector. Let $H(e)$ be the maximum number of links that can be scheduled if link e is not scheduled and let $H = \max(1, \max_{e \in E} H(e))$.

We define the load vector as the vector $\mathbf{b} = (E[B_s(e), e \in E)$.

The following result holds:

Theorem 8. *Under Assumption 1 and if $E[B_s(e)B_s(e')]$ is bounded for all $e, e' \in E$ then `Algorithm Log` stabilizes the load vector $\mathbf{b}$ if $(CH/(C+H-1))(\mathbf{b} + \epsilon\, \mathbf{c}) \in \mathcal{C}$ for some $\epsilon > 0$.*

The proof is an adaptation of the proof of Theorem 1 in [2] (Hint: replace the definition of the set $\mathcal{L}'(s)$ in [2] by $\mathcal{L}'(s) = \{e \in E : q_{sC+1}(e) \geq LCc(e)\}$) since `Algorithm Log` behaves as a hybrid scheduling whenever $\mathcal{L}'(s) = E$ for all $s \geq 0$.

Example 9. *Consider a graph $G = (V, E)$ composed of $|E|$ links in series with an arbitrary interference distance d. In this case, $H = 2(d+1)$ since a link is in conflict with at most $2(d+1)$ other links and $C = d+2$ since $d+2$ colors are needed to color a path when the interference distance is d.*

Assume that for each e, $\{A_t(e), t \geq 0\}$ is an idd sequence with $\overline{A}(e) = E[A_t(e)]$ and that $E[A_t(e)A_t(e')]$ is bounded for $e, e' \in E$. In particular, $E[B_s(e)] = 2\overline{A}(e)$.

Theorem 8 says that if the vector $\mathbf{a} = (\overline{A}(e), \in E)$ is such that there exists $\epsilon > 0$ with $(2(d+2)/3)((d+2)\mathbf{a} + \epsilon\,\mathbf{c}) \in \mathcal{C}$ then $\mathbf{a}$ stabilizes `Algorithm Log`.

We conjecture that if

$$\sum_{e' \in \varepsilon(e) \cup \{e\}} \overline{A}(e') < \min_{e' \in \varepsilon(e) \cup \{e\}} c(e') \tag{3}$$

for all $e \in E$ then `Algorithm Log` is stable.

If this conjecture holds a natural question is to know what stability region among that defined by Theorem 8 and by conditions (3) is the largest. We provide a partial answer to that question in the setting of Example 9 when $c(e) = 1$ for all $e \in E$.

Assume that $(2(d+2)/3)((d+2)\mathbf{a} + \epsilon) \in \mathcal{C}$ for some $\epsilon > 0$ so that from Example 9 `Algorithm Log` stabilizes the load vector $\mathbf{a} = (a(1), \ldots, a(|E|))$ with $a(e) := \overline{A}(e)$.

From the definition of the region $\mathcal{C}$ this means that there exist constants $\lambda_i > 0$ with $\sum_i \lambda_i < 1$ such that

$$(2(d+2)/3)\left((d+2)a(e) + \epsilon\right) = \sum_i \lambda_i \mathbf{I}_i(e), \quad e = 1, \ldots, |E|. \tag{4}$$

From (4) and the positiveness of ϵ we deduce that

$$a(e) < \frac{3}{2(d+2)^2} \sum_i \lambda_i \mathbf{I}_i(e), \quad e = 1, \ldots, |E|. \tag{5}$$

Let us show that the vector $\mathbf{a}$ strictly lies within the region defined by conditions in (3). This amounts to showing that $D(e) < 1$ for $e = 1, \ldots, |E|$ with

$$\begin{aligned} D(e) := a(e-1+d) + \cdots + a(e-1) + a(e) \\ + a(e+1) + \cdots + a(e+1+d) \end{aligned}$$

where $a(e) = 0$ if $e \notin \{1, \ldots, |E|\}$ by convention. From (5) we find

$$\begin{aligned} D(e) < \frac{3}{2(d+2)^2} \sum_i \lambda_i (\mathbf{I}_i(e-1+d) + \cdots \\ + \mathbf{I}_i(e-1) + \mathbf{I}_i(e) + \mathbf{I}_i(e+1) + \cdots + \mathbf{I}_i(e+1+d)) \end{aligned}$$

for $e = 1, \ldots, |E|$. By definition of the interference distance d we observe that

$$\begin{aligned}\mathbf{I}_i(e-1+d) + \cdots + \mathbf{I}_i(e-1) + \mathbf{I}_i(e) \\ + \mathbf{I}_i(e+1) + \cdots + \mathbf{I}_i(e+1+d) \leq 2\end{aligned}$$

for $e = 1, \ldots, |E|$ since at most 2 links can be simultaneously active among $2(d+1)+1$ consecutive links. Hence, $D(e) < 3/(d+2)^2 \leq 3/4$ for $e = 1, \ldots, |E|$ and for any $d \geq 0$. This shows that the stability region given in (3) is larger than that given in Theorem 8.

We conclude this section by observing that conditions

$$\overline{A}(e) < c(e)/C, \quad e \in E. \tag{6}$$

ensure the stability of `Algorithm Log` when $K = 1$. This is so because when $K = 1$ the virtual weights do not depend on the backlogs, which in turn implies that `Algorithm Log` becomes a deterministic algorithm that guarantees each link e to transmit at least $c(e)$ packets every C consecutive slots.

7 Simulations

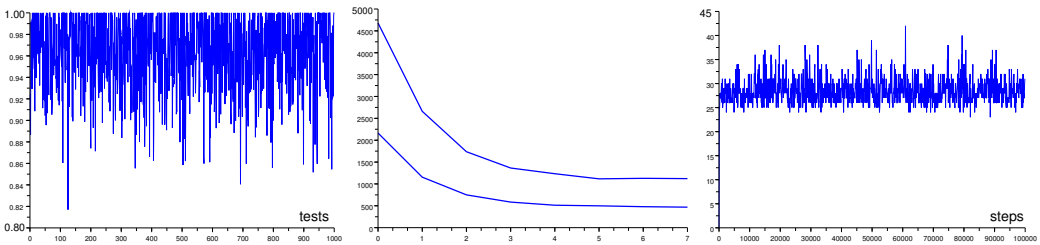
We investigate in this section performance of our distributed scheduling algorithm, `Algorithm Log`, via simulations. More precisely, we compute the total sum of active links at a given step (Section 7.1), and we study the evolution of the largest queue of the network for thousands time slots (Section 7.2). For instance in Section 7.2.2 we compare the largest and average value computed by `Algorithm Log` to the one found by `Augmenting Paths Algorithm`, proposed in [5].

7.1 Weight of the Set of Active Links

We study in this section the ratio between weights of the optimal matching and the matching computed by `Algorithm Log` for a path. Figure 7(a) shows this ratio for a path $G = (V, E)$ composed of $|E| = 50$ edges for 1000 tests. At each test, we assign a weight for each $e \in E$ as follows: $P(q_t(e) = i) = 1/101$, $0 \leq i \leq 100$. We observe that, for each test, `Algorithm Log` finds a matching of weight at least 4/5 than the optimal.

7.2 Evolution of the Largest/Average Queue

We investigate in this section the evolution of the length of the queues for a certain number of time slots applying `Algorithm Log`. The simulations have been done for path networks (Section 7.2.1), grid networks (Section 7.2.2), and random networks (Section 7.2.3). For each topology we will explain the chosen parameters and describe the evolution of the size of



(a) Ratio optimal / `Algorithm Log`. (b) Largest queue depending to K (c) Random network

Figure 7: Performance of `Algorithm Log`

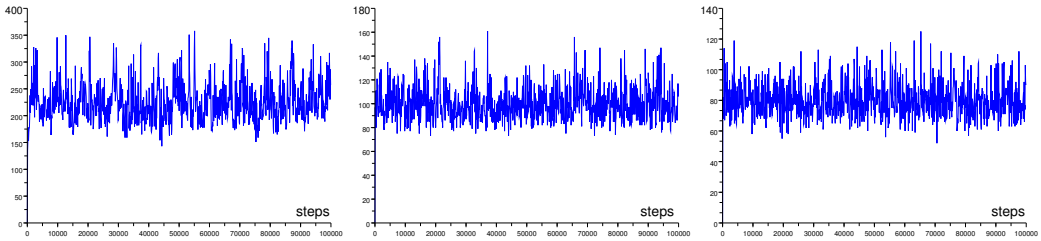
largest queue of the network, for thousands steps. Furthermore we compare in Section 7.2.2 `Algorithm Log` to `Augmenting Paths Algorithm` for the grid topology (largest and average values), using parameters chosen in [5].

7.2.1 Paths

We show here some simulations for path networks. From Section 5.3, it is sufficient to assign the sequence of integers (colors) $1, 2, 1, 2, \ldots$ on the edges of the path, considering the interference model defined by $d = 0$. Thus $C = 2$.

We first investigate `Algorithm Log` for a path $G = (V, E)$ composed of $|E| = 100$ edges during 100000 time slots. We choose $K = 1000$. Furthermore for any edge $e \in E$, the capacity is $c(e) = 18$, and the arrival process is defined as follows: n_1 messages arrive in average on e at a step $t \geq 1$ for an edge $e \in E$ with color 1 ($\gamma(e) = 1$) and n_2 message arrives in average for the others (each edge $e \in E$ such that $\gamma(e) = 2$). Figure 8(a), Figure 8(b), and Figure 8(c) represent the largest weight of edges at each time slot respectively for pairs $(n_1, n_2) = (16, 1)$, $= (12, 4)$, and $= (8, 8)$. We observe that the largest queue is always lower than respectively 400, 180, and 140.

Then we investigate the impact of the value of K on the largest and average queue of a path $G = (V, E)$ composed of 100 edges with $d = 0$. For each $e \in E$, an average of 0.75 (respectively 0.2) messages arrive at each step $t \geq 1$ if $\gamma(e) = 1$ (respectively $\gamma(e) = 2$). Figure 7(b) represents largest and average weight after 50000 time slots for different values of K: $K = 1, 2, 4, 8, \ldots, 128$. The plot illustrates the performance-overhead tradeoff of K - a higher K leads to lower average queue sizes, but most of the gains come with relatively small values of K.



(a) $(n_1, n_2) = (16, 1)$ (b) $(n_1, n_2) = (12, 4)$ (c) $(n_1, n_2) = (8, 8)$

Figure 8: evolution of the largest queue for a path network composed of 100 edges for `Algorithm Log` ($d = 0$, $K = 1000$, $L = 99$, $C = 4$).

7.2.2 Square Grids

We compare in this section our distributed algorithm, `Algorithm Log`, to the one proposed in [5], `Augmenting Paths Algorithm`, choosing the same parameters. More precisely, the simulations have been done for a square grid $G = (V, E)$ of $|V| = 121$ nodes and with $|E| = 220$ edges (Figure 9). For each edge $e \in E$, the capacity is $c(e) = 1$. Furthermore values mentioned in Figure 9 represent average arrivals divided by a parameter $\lambda \leq 1$. As described in Section 3, `Augmenting Paths Algorithm` requires parameters in inputs: k and p. In [5], simulations have been done for the following parameters: $\{k = 2, p = 0.2\}$, $\{k = 3, p = 0.2\}$, and $\{k = 3, p = 0.1\}$. In [5], the number of time slots is 48000. Recall that `Augmenting Paths Algorithm` is only valid for the primary node interference model, and so $d = 0$ for our comparisons. Let us describe the parameters chosen for `Algorithm Log`. From Section 5.3, the minimum value of C respecting constraints described in Section 4 is $\chi'(G) = 4$. See Figure 5(b) for an example of valid assignment of $\gamma(e)$. Furthermore we choose $K = 1000 = L + 1$. We compare in Figure 10 the largest and average values for each time slot $t \leq 50000$ with `Algorithm Log` and `Augmenting Paths`

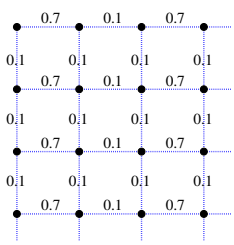
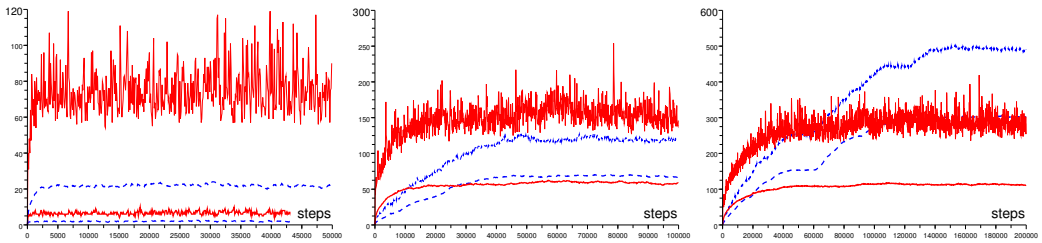


Figure 9: grid network.



(a) $\lambda = 0.90$ (b) $\lambda = 0.95$ (c) $\lambda = 0.97$

Figure 10: evolution of the largest and average value of queues for the grid network (121 nodes and 200 edges) for `Augmenting Paths Algorithm` ($k = 2, p = 0.2$) and for `Algorithm Log` ($d = 0$, $K = 1000$, $L = 999$, $C = 4$).

`Algorithm`, for the parameter $\{k = 2, p = 0.2\}$ described in [5]. Simulation results for the two other parameters are analogous. Figure 10(a), Figure 10(b), and Figure 10(c) represent respectively largest and average value for $\lambda = 0.90, 0.95, 0.97$ (red straight lines represent `Augmenting Paths Algorithm` and blue dotted lines represent `Algorithm Log`). We note that the flexibility of `Algorithm Log` comes at a price for highly loaded networks, as `Augmenting Paths Algorithm` performs better on average (though both are stable). However, `Augmenting Paths Algorithm` is specialized for $d = 0$ where `Algorithm Log` is stable for any value of d. Figure 7.2.3 shows a simulation for the same grid network with the arrival rate in the capacity region established in Section 6. As can be seen, `Algorithm Log` stabilizes the system.

7.2.3 Random Graphs

Finally in this section we apply `Algorithm Log` to a random network. First let us describe how we have built our *random graphs* before analyzing the evolution of the largest queue.

Given a grid graph $G' = (V, E')$, we delete an edge $e \in E'$ with a certain constant probability p, before adding others edges with another constant probability q between two nodes not so *far* than a certain Euclidean distance. We get a random transmission graph $G = (V, E)$. Figure 6 shows an example of such a random graph with $|V| = 33$ nodes and $|E| = 56$ edges.

In our simulations, we investigate the stability criteria defined in Equation 3 of Section 6 with $c(e) = 20$, $\forall e \in E$.

The arrival processes for edges of the network are defined as follows: we first assign to each edge $e \in E$ an average arrival of 20 messages per step $t \geq 1$ (uniform between 0 and 40). To respect previous conditions, we decrease the average number of arrivals per step by 1 for an edge e if it is necessary, until having the desired condition. In Figure 7(c), we have

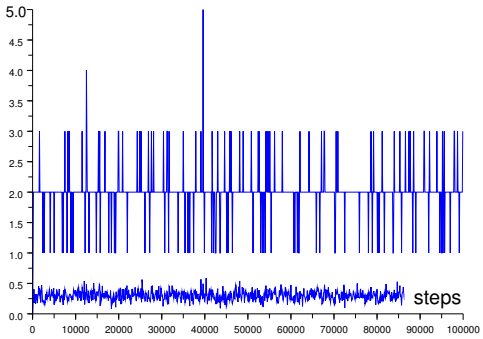


Figure 11: Largest and average queue size for a grid network with $d = 1$ interference constraint

applied `Algorithm Log` for such a graph $G = (V, E)$ with $|V| = 200$ nodes and $|E| = 424$ edges. The number of time slots is 100000, with $K = 10000$ and $C = 20$. The interference model is defined by $d = 0$. Figure 7(c) represents the largest queue for each time slot t, and it is easy to show that `Algorithm Log` stabilizes the queueing system under previous conditions.

8 Multi-Hop Traffic

We show in this section how to extend `Algorithm Log` to take into consideration multi-hop traffic when routing is fixed. The main idea is to change the vector q_t of previous weights (Section 2) and the vector q'_t of previous virtual weights (Section 4.1) into multi-hop vector of weights q_t^m and virtual multi-hop vector of weights $q_t'^m$. Let $G = (V, E)$ be the transmission graph. More precisely $\forall e \in E$, $q_t^m(e) = \sum p_t^i(e) h_t^i(e)$ where $p_t^i(e)$ is the packet numbered i in the queue of link e at step t and $h_t^i(e)$ is the number of remaining hops of the previous packet. From $q_t^m(e)$, we compute $q_t'^m(e)$ for e in the same manner than in `Algorithm Log` (Section 4.1). We assume here that the routing is pre-computed. Note that the policy of services can changed with application requirements.

9 Conclusion

We proposed in this article the first, to our knowledge, a distributed algorithm for the transmission scheduling problem in wireless networks with constant overhead and arbitrary binary interference. We proved that the set of active links at each time slot is maximal. We proposed sufficient stability conditions and investigated performance of our distributed

algorithm via simulations. We also proposed extensions to our algorithm in the multi-hop case.

Since our algorithm works with arbitrary interference constraints, and uses observed interference for its functioning, we believe it translates into a practical implementation well and that is planned as future work. It will be interesting now to prove a better stability condition and characterize the stability region of our algorithm. Although simulation results strongly suggest that the algorithm is capacity achieving, it remains to be proven analytically.

References

[1] H. Balakrishnan, C. Barrett, V. Kumar, M. Marathe, and S. Thite. The distance-2 matching problem and its relationship to the mac-layer capacity of ad hoc wireless networks. *IEEE, J. Selected Areas in Communication*, 22(6):1069–1079, 2004.

[2] V. Bhandari and N. H. Vaidya. A result on hybrid scheduling in wireless networks. Technical report, University of Illinois, Dept. Electrical and Computer Eng., March 2009.

[3] V. Bonifaci, R. Klasing, P. Korteweg, L. Stougie, and A. Marchetti-Spaccamela. *Graphs and Algorithms in Communication Networks*, chapter Data Gathering in Wireless Networks. Springer-Verlag, 2009.

[4] A. Brzezinski, G. Zussman, and E. Modiano. Enabling distributed throughput maximization in wireless mesh networks: a partitioning approach. In *MobiCom*, pages 26–37. ACM, 2006.

[5] L. X. Bui, S. Sanghavi, and R. Srikant. Distributed link scheduling with constant overhead. *IEEE/ACM Transactions on Networking*, 17(5):1467–1480, 2009.

[6] K. Cameron. Induced matchings. *Discrete Applied Mathematics*, 24(1-3):97–102, 1989.

[7] P. Chaporkar, K. Kar, X. Luo, and S. Sarkar. Throughput and fairness guarantees through maximal scheduling in wireless networks. *IEEE Transactions on Information Theory*, 54(2):572–594, 2008.

[8] H. Chen, X. Xie, and H. Wu. A queue-aware scheduling algorithm for multihop relay wireless cellular networks. *Mobile WiMAX Symposium, IEEE*, 0:63–68, 2009.

[9] R. Diestel. Graph Theory (Graduate Texts in Mathematics), 1997.

[10] A. Eryilmaz, O. Asuman, and E. Modiano. Polynomial complexity algorithms for full utilization of multi-hop wireless networks. *INFOCOM*, pages 499–507, 2007.

[11] S. Fiorini and R. J. Wilson. *Edge-colourings of graphs*. Pitman, 1977.

[12] A. Gupta, X. Lin, and R. Srikant. Low-complexity distributed scheduling algorithms for wireless networks. In *INFOCOM*, pages 1631–1639, 2007.

[13] R. Klasing, N. Morales, and S. Pérennes. On the complexity of bandwidth allocation in radio networks. *Theoretical Computer Science*, 406(3):225 – 239, 2008.

[14] V. S. A. Kumar, M. Marathe, S. Parthasarathy, and A. Srinivasan. End-to-end packet-scheduling in wireless ad-hoc networks. In *SODA '04*, pages 1021–1030. SIAM, 2004.

[15] L. Lovász and M. Plummer. *Matching Theory*, volume 29 of *Annals of Discrete Mathematics*. North-Holland, 1986.

[16] R. Mazumar, G. Sharma, and N. Shroff. Maximum weighted matching with interference constraints. *FAWN, Pisa, Italy*, 2006.

[17] J. Misra and D. Gries. A constructive proof of vizing's theorem. *Information Processing Letters*, 41, 1992.

[18] E. Modiano, D. Shah, and G. Zussman. Maximizing throughput in wireless networks via gossiping. *SIGMETRICS Perform. Eval. Rev.*, 34(1):27–38, 2006.

[19] Y. Orlovich, G. Finke, V. Gordon, and I. Zverovich. Approximability results for the maximum and minimum maximal induced matching problems. *Discrete Optimization*, 5(3):584 – 593, 2008.

[20] S. Rajagopalan, D. Shah, and J. Shin. Network adiabatic theorem: an efficient randomized protocol for contention resolution. In *SIGMETRICS '09: Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 133–144, New York, NY, USA, 2009. ACM.

[21] S. Sanghavi, L. Bui, and R. Srikant. Distributed link scheduling with constant overhead. In *SIGMETRICS*, pages 313–324, 2007.

[22] L. J. Stockmeyer and V. V. Vazirani. Np-completeness of some generalizations of the maximum matching problem. *Inf. Process. Lett.*, 15(1):14–19, 1982.

[23] L. Tassiulas. Scheduling and performance limits of networks with constantly changing topology. *IEEE Transactions on Information Theory*, 43(3):1067–1073, 1997.

[24] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Conference on Decision and Control*, pages 2130–2132 vol.4, 1990.

[25] P.-J. Wan. Multiflows in multihop wireless networks. In *MobiHoc '09*, pages 85–94. ACM, 2009.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399