Distributed Call Scheduling in Wireless Networks

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

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Distributed Call Scheduling in Wireless Networks

Jean-Claude Bermond — Dorian Mazauric — Philippe Nain

N° 6763

December 2008

Thème COM

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

Distributed Call Scheduling in Wireless Networks

Jean-Claude Bermond*†, Dorian Mazauric*‡† , Philippe Nain*‡†

Thème COM — Systèmes communicants
Projets Mascotte and Maestro

Rapport de recherche n° 6763 — December 2008 — 21 pages

Abstract: This work investigates distributed call scheduling in wireless (mesh) networks in presence of random arrivals on each link. In these networks, interference phenomena prevent 'neighboring' links to be simultaneously active as otherwise transmissions will fail. For instance, in the primary interference model only links which do not have any node in common may be simultaneously activated. Time is slotted and the objective of this study is to design and evaluate the performance of call scheduling algorithms which determine which links will be activated in the current time-slot. A call scheduling algorithm takes as input the backlog on each link and produces an admissible schedule (a matching in the primary node model), namely, a set of collision-free transmissions. Since nodes do not have a global knowledge of the network, a call scheduling algorithm must be obtained in a decentralized manner through local node interactions. Furthermore it is practically important to obtain algorithms having a constant overhead during the control phase. In this paper we introduce two distributed call scheduling algorithms with a constant overhead and which work for any binary interference model. For the second algorithm we prove that in each interference set, there is at least one active edge. Their performance (throughput, stability) is investigated and compared via simulations to that of previously proposed schemes.

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

This work was partially funded by Région PACA and by 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.

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

Algorithmes distributés d'ordonnancement 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, ne peuvent être activés simultanément que des liens n'interférant pas entre eux. Par exemple dans un modèle primaire, on ne peut activer que des liens deux à deux non adjacents. Nous nous plaçons dans un contexte d'arrivée aléatoire de messages et l'objectif est d'assurer un bon comportement du réseau en particulier d'assurer la stabilité des files d'attente, en limitant le nombre moyen de messages en attente. Des algorithmes centralisés permettant de décider quels liens sont activés à chaque étape existent mais ils supposent une connaissance globale du réseau et sont peu adaptés aux applications. Il est donc nécessaire de concevoir des algorithmes distribués qui utilisent une connaissance très locale du réseau. Nous proposons dans cet article deux algorithmes distribués, valides quelque soit le modèle d'interférence binaire et avec une phase de contrôle de durée constante, améliorant les algorithmes existants vérifiant uniquement l'un ou l'autre de ces deux critères.

Mots-clés : réseau, sans-fil, ordonnancement, distribué, algorithme, interférence, stabilité.

1 Introduction

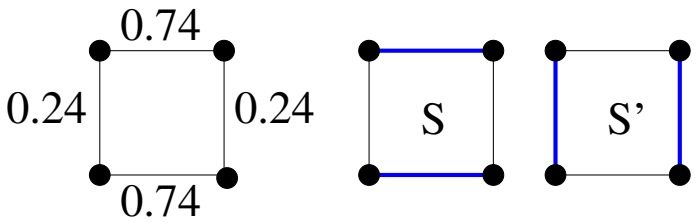
Call scheduling is a main problem in telecommunication networks. It has received a lot of attention both for wired and for wireless networks (radio, ad hoc, sensor network). In a wireless network a difficulty comes from interference problems. During a step or slot, only calls which do not interfere can be scheduled. In this paper we consider a binary interference model where two calls which are within some interference distance d from each other interfere. A particular instance is the primary node model ($d = 0$), where two calls interfere if their links intersect (one node can communicate with at most another node); therefore, a set of calls can be activated together in the same slot only if they form a matching. The traffic is single-hop and the arrival process to each link 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 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 [7] [8] and deterministic arrivals in [4].

In practice, 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 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 [1, 2, 5] distributed algorithms are described but they all lead to communication overheads which increase with the size of the network. In particular [3] 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).

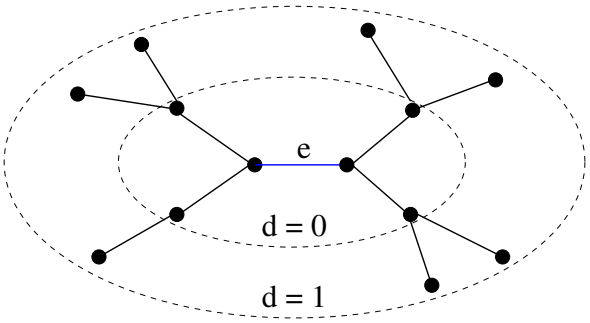
The need for distributed algorithms with a small constant overhead has been emphasized in [6], where a distributed algorithm with a constant overhead depending on the quality of the desired approximation is described; however it is only valid for the primary node model ($d = 0$). In this work we introduce two distributed algorithms which hold for any binary interference model and which generate a constant communication overhead. It is worth noting that in the general case (i.e. interference model different from the primary node 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*.

The rest of the paper is organized as follows. We first introduce the model used in our work (Section 2) and review some related works (Section 3). In Section 4, we propose two distributed link scheduling algorithms for wireless networks which meet the interference constraints and with a constant communication overhead. We also give some properties of the algorithms, in particular we prove that in each interference set, there is at least one active edge. In Section 5 we address their stability (i.e. finiteness of the backlog at each link). In Section 6 we present some simulation results for $d = 0$ and compare the performance our

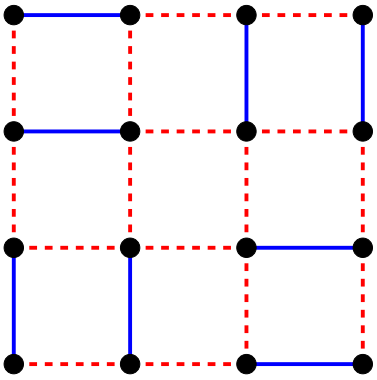
algorithms with that of the algorithm in [6]. Finally in Section 7, we present generalizations of our algorithms to take into account multi-hop traffic and queues on nodes.



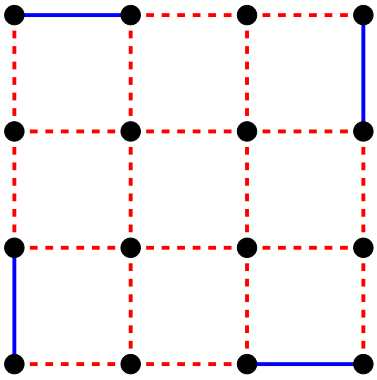
(a) Link Scheduling for a $4-cycle$ with primary node interference model ($d=0$): (S) three times over four and (S') one time over four.



(b) Examples of interference sets $\epsilon(e)$ for $d=0$ et $d=1$.



(c) Set of activated links (lines) for $d=0$.



(d) Set of activated links (lines) for $d=1$.

Figure 1: Examples of Link Scheduling (a), interference sets (b), and active links sets (c,d).

2 Modeling

We model the network by an undirected transmission graph $G = (V, E)$, where there is an edge (link) $e = uv$ between two nodes u and 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 call 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.

Interferences can be modeled in different ways. We can use SNR (Signal-to-Noise Ratio) models but that makes both the algorithms and the analysis hard. We choose to use a binary symmetric model and define the interference set of e: $\varepsilon(e)$ as the set of edges interfering with e. Our algorithms are valid for any set $\varepsilon(e)$, but in the examples and simulations, we use a model based on an interference distance d as follows. Two edges interfere if one vertex of the first edge is at distance in G at most d from a vertex of the second edge. More precisely, the interference set of an edge $e = u_1u_2$ is $\varepsilon(e) = \{e' = v_1v_2 \in E, \exists i, j \in [1, 2], d(u_i, v_j) \leq d\}$, where $d(u, v)$ is the distance in G that is the length of a shortest path between u and v.

We suppose that time is slotted and synchronized in the network. Our algorithms consist like in [3, 6] of a sequence of steps (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 set of active links and a *sending phase*. We will say that a step is valid if the edges activated during this step do not interfere. The particular case $d = 0$ is a commonly used and is known as the primary node (or node exclusive) interference model. In that case two edges interfere if they are incident and an active step is a matching of the transmission graph. In the case $d = 1$, which is more realistic, an active step is what is called an induced matching.

Fig. 1(a) represents the two maximal matchings S and S' which can be used as active steps for a 4-cycle. In Fig. 1(b) we give an example of interference sets for $d = 0$ and $d = 1$. In Fig. 1(c) (respectively Fig. 1(d)) we describe an example of a maximal set of active links (no edge can be added as active) for the interference model $d = 0$ (respectively $d = 1$) in a 4 by 4 grid.

We introduce for each edge of the transmission graph and for each step $t \geq 0$, the variable $a_t(e)$ such that $a_t(e) = 1$ if $e \in E$ 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 $e \notin \varepsilon(e')$.

Traffic is dynamic and is assumed in this article to be single-hop, that is to say one packet sent through a communication link leaves the network just after. To each edge 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. At each step, new packets arrive in the links according to some probability law, specific to each link and not necessary known by designers. Let $A_t(e)$ be the number of packets arriving in edge $e \in E$ during a step t. Remark that these new packets will be taken in consideration only at step $t + 1$. The capacity of a link e is the number of packets that this link can serve during one step (if the link is activated) and is denoted by $c(e)$. Thus we have $q_{t+1}(e) = (q_t(e) - a_t(e).c(e))^+ + A_t(e)$ with $[x]^+ = max(x, 0)$.

In Fig. 1(a) we indicate the average values of $A_t(e)$ for each link; furthermore $c(e) = 1$. A scheduling consists of choosing three steps over four the maximal matching S and one step over four the maximal matching S'. Doing so the size of each queue remains bounded.

3 Previous Works

We briefly describe in this section two algorithms respectively proposed in [3] and [6] as we use the same modeling. They are both distributed but the algorithm in [3] does not admit a constant overhead, whereas the one described in [6] is valid only for the primary node model ($d = 0$).

In the algorithm described in [3], each edge chooses a backoff value t_0 at the beginning of the control phase of each step, and sends, if active, a control message at the mini-slot t_0. If it receives a control message from one edge of its interference set during or before the mini-slot t_0, it will be inactive during the data phase. Otherwise it will be active. This simple algorithm is valid for any interference set, but the choice of its backoff is a function of the weights of the edges located in its interference set and function of the weights of edges in $\varepsilon(e')$ with $e' \in \varepsilon(e)$. So, at each step, an edge has to update the weights of edges located in its $2d - neighborhood$, if we have an interference distance d. Therefore the overhead is not constant and furthermore one has to obtain this information, which, due to interferences, is a problem as difficult as the call scheduling.

The algorithm described in [6] has a constant overhead but it is specific to the primary node model ($d = 0$). The goal is to find a new matching bigger than the previous one in terms of the size of the queues of active edges. Given a matching M_t at step t, during step $t + 1$, the algorithm will find paths, alternating edges in M_t and edges not in M_t. At the end, if the total weight of inactive edges of the alternating path is greater than total weight of previous matching M_t, then the matching is changed, the status of each edge is changed and we get M_{t+1}. This control phase uses a constant overhead depending on the approximation ratio of the optimal matching we want. But the algorithm is designed specifically for primary node model and uses alternating paths techniques which work only for matchings. Recall that determining a maximum matching can be done in polynomial time with a centralized algorithm, but for $d \geq 1$, determining a maximum valid set of active edges is an NP-complete problem. Furthermore the convergence of the algorithm is very slow (see Section 6).

4 Our Algorithms

We assume in the rest of the paper that $\forall t \geq 0, \forall e \in E$, $q_t(e) \in [0, K-1]$ where K depends on the size of the desired overhead. Indeed this hypothesis is not restrictive, as the general case can be reduced to this one by ranging the values of $q_t(e)$ in K classes. For example we put the values $q_t(e) \geq M$ (M a constant) in the class K and the other values in $K-1$ classes of almost same size.

4.1 Algorithm Log 1

The control phase of Algorithm Log 1 is divided into two subphases: a *regular subphase* with T mini-slots and a *random subphase*. During the control phase, an edge is active, inactive, or still undetermined. Before mini-slot 1, each edge is undetermined.

Each edge $e \in E$ computes a *control vector* $v_{t,e} = (v_{t,e}(1), v_{t,e}(2), \ldots, v_{t,e}(T))$ where $v_{t,e}(i)$, $(i = 1, \ldots, T-1)$, corresponds to the ith bit of its weight. Furthermore $v_{t,e}(T) = 1$, if $q_t(e)$ is even, and $v_{t,e}(T) = 0$ otherwise. As explained before and w.l.o.g., weights are bounded by $K-1$, and so we can describe the control vector for each possible weight. For example if $q_t(e) = 6$ and $K = 16$ (and so $T = 5$), then we have $v_{t,e} = (0, 1, 1, 0, 1)$. In Table 1, we have every possible weight if $K = 16$. During a mini-slot i, $1 \leq i \leq T$, e sends a control message if and only if e is still undetermined and $v_{t,e}(i) = 1$.

Let us describe now the protocol for an edge $e \in E$ still undetermined during a mini-slot i, $1 \leq i \leq T$. Recall that $\varepsilon(e)$ denotes the interference set of edge e:
(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 from an edge in $\varepsilon(e)$ and does not send a control message, then e becomes inactive;
(c) otherwise e remains undetermined.

Remark that edges with odd weights send one control message at mini-slot $T-1$. To be fair, mini-slot T is for even weight edges still undetermined.

Fig. 2 represents the regular control subphase of Algorithm Log 1 for a 4 by 4 grid network, with $d = 0$ and $K = 16$ (and so $T = 5$). In this example the matching found is maximal, that is to say no edge can be added to the current matching without changing it.

Fig. 3(a-e) represents also the regular subphase of Log Algorithm 1 for a path network, with $d = 0$ and $K = 16$ (and so $T = 5$). The edge of weight 15 is the unique active edge. This example is one of the worst case.

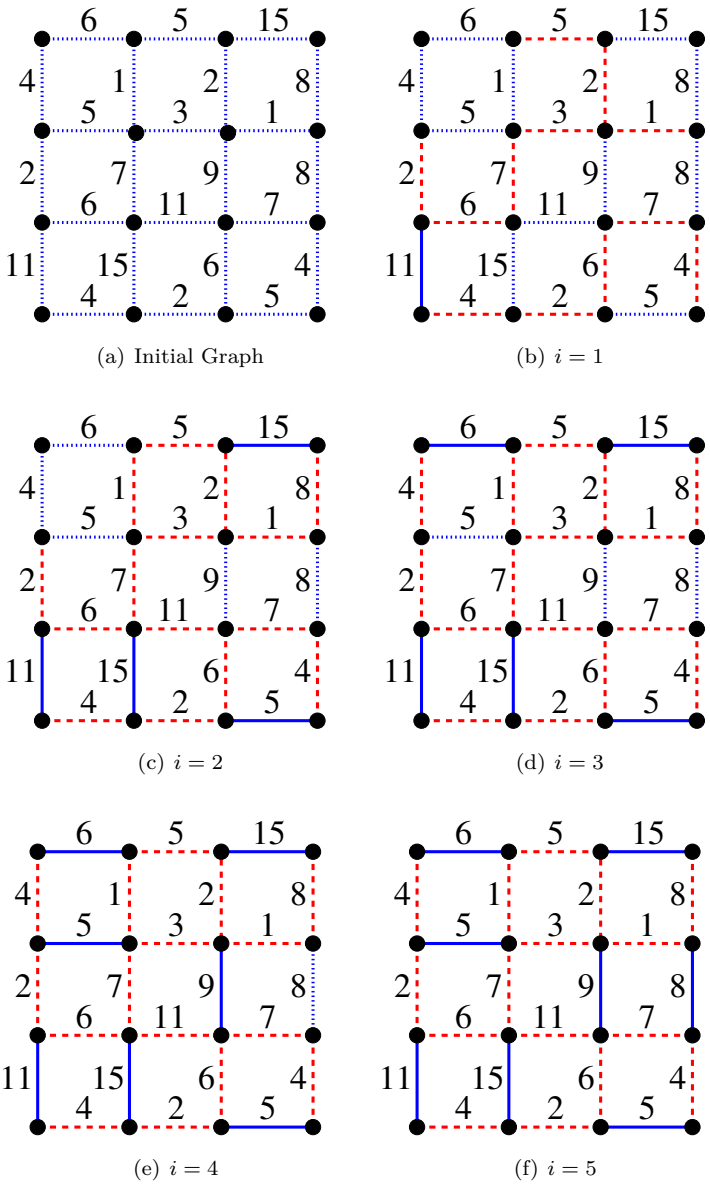
To deal with these bad cases we add a constant additional number $T' + 1$ of mini-slots (corresponding to the random subphase) in order to add more active edges. During the first mini-slot we change the status of the inactive edges which have no active edges in their interference set to undetermined. Then, each edge chooses at random a number $k \in [0, T']$. The protocol is the same as above except that an edge sends a control message at the mini-slot k, if $k > 0$, if it is still undetermined. Note that this random subphase does not modify the set of active edges computed by the regular subphase.

Fig. 3(f) represents the first mini-slot needed to get some new undetermined edges. In Fig. 3, we indicate in line (g) the random values drawn with $T' = 2$. Then in the next two mini-slots (h) and (i) some edges with random values 1 (here the edge with weight 8) and then 2 (here the edge with weight 9) become also active. We get a matching greater than before.

Altogether the overhead for Algorithm Log 1 is $T_1 = T + 1 + T' = O(log_2(K))$. In the example $T_1 = 8$.

Table 1: Control Vector $v_{t,e}$ for an edge $e \in E$ at step $t \geq 0$ for every possible weight for Algorithm Log 1 ($K = 16$) and Algorithm Log 2 ($c_2 = 2$, $K' = 2K = 16$).

Algorithm Log 1	Algorithm Log 2			Control Vector				
$q_t(e)$	$q_t(e)$	$g(t,e)$	$q'_t(e)$	$v_{t,e}(1)$	$v_{t,e}(2)$	$v_{t,e}(3)$	$v_{t,e}(4)$	$v_{t,e}(5)$
0	0	0	0	0	0	0	0	1
1	0	1	1	0	0	0	1	0
2	1	0	2	0	0	1	0	1
3	1	1	3	0	0	1	1	0
4	2	0	4	0	1	0	0	1
5	2	1	5	0	1	0	1	0
6	3	0	6	0	1	1	0	1
7	3	1	7	0	1	1	1	0
8	4	0	8	1	0	0	0	1
9	4	1	9	1	0	0	1	0
10	5	0	10	1	0	1	0	1
11	5	1	11	1	0	1	1	0
12	6	0	12	1	1	0	0	1
13	6	1	13	1	1	0	1	0
14	7	0	14	1	1	1	0	1
15	7	1	15	1	1	1	1	0



(a) Initial Graph (b) $i = 1$

(c) $i = 2$ (d) $i = 3$

(e) $i = 4$ (f) $i = 5$

Figure 2: Regular subphase of Algorithm Log 1 for a 4 by 4 grid, $d = 0$ and $K = 16$ (5 control mini-slots). Blue edges represent active, red represent inactives and dotted-lines are still undetermined.

4.2 Algorithm Log 2

The idea of Algorithm Log 2 is to repeat the regular subphase of Algorithm Log 1. After such a phase we do a mini-slot during which we change the status of the inactive edges which have no active edges in their interference set to undetermined. Then we apply a regular subphase using only the undetermined edges and so on. We repeat this process a certain number of times.

Fig. 4(a) represents the result of the regular phase of Algorithm Log 1 on the preceding example (Fig. 3(a-e)). Using 1 control mini-slot, each inactive edge without active in its interference set, becomes undetermined (Fig. 4(b)). We repeat the regular subphase one more time and we get the matching of Fig. 4(c); another application gives a maximal matching (Fig. 4(e)).

However this procedure does not work in case of equality; indeed, if all the edges have initially equal weight they are all undetermined at the end of the subphase. To avoid this problem, we introduce new (virtual) weights $q'_t(e)$ for each edge $e \in E$ in such a way that two edges belonging to a same interference set have different virtual weights. One way to do that is to choose $q'_t(e) = c_2.q_t(e) + g(t,e)$ with $0 \leq g(t,e) < c_2$ and c_2 the largest size of $\varepsilon(e)$. More precisely $g(t,e) = g(0,e) + t \pmod{c_2}$ with $g(0,e) \neq g(0,e')$ if e' interferes with e. In Table 1, we have every possible weight if $K = 16$ and $c_2 = 2$.

Fig. 5 describes a critical example of transmission graph with almost similar weights. Using the transformation described before with $c_2 = 2$ (enough for a path in the primary node model) we obtain the new weights of Fig. 5(b). Fig. 5(c) (respectively Fig. 5(d)) shows the active links set for virtual weights $q'_t(e)$ (respectively for real weights $q_t(e)$) at the end of the Algorithm Log 2.

The size of the queues is now $K' = c_2K$. Suppose we apply α times the subphase of Algorithm Log 1, then the overhead of Algorithm Log 2 is $T_2 = \alpha(log_2(c_2K) + 2)$.

We can prove the following properties for our algorithms.

Lemma 1. *If all the weights are different in every interference set, at the end of the regular subphase of Algorithm Log 1 there is no undetermined edge.*

Proof. Suppose e is undetermined at the end of the regular subphase of Algorithm Log 1. At the last mini slot where e sends a control message ($T - 1$ if $q_t(e)$ is odd, T if $q_t(e)$ is even), it stayed undetermined if another edge e' sends a message at this 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 slot, a contradiction. □

Corollary 2. *A strict local maximum edge at some step (that is an edge with a weight greater than the other edges in its interference set) is always active at this step with Algorithm Log i (i=1,2).*

Theorem 3. *If we apply Algorithm Log 2 with $\alpha = log_2(c_2K)/2$ then for all $e \in E$, there exists one edge $e' \in \varepsilon(e) \cup \{e\}$ such that e' is active at the end of the algorithm.*

Proof. Let e_0 be any edge. Either e_0 is active at the end of the first regular subphase of Algorithm 1, we are done. Or e_0 is inactive due to an edge e_1 (with $q_t(e_1) > q_t(e_0)$) which has sent a control message at mini slot t_0. Either $e_1 \in \varepsilon(e_1)$ is active or e_1 is inactive due to an edge e_2 which has sent a control message at mini slot $t_1 > t_0$, and so on. Let k be the largest index of inactive edges $e_0, e_1, e_2, \ldots, e_k$. Edge e_i is inactive due to e_{i+1} which has been sent a control message at mini slot i and edge e_{k+1} is active. As $\alpha > t_k > \ldots > t_i > t_{i-1} \ldots > t_0$ and so $k \leq \alpha - 2$. Edge e_{k+1} is at distance $(k-1)d \leq (\alpha - 3)d$ of e_0. We will prove that at the regular subphase j, there exists an active edge at distancec at most $(\alpha - 2 - j)d$ of e_0. It is done for $j = 1$. Assume that it is not the case and let $e_0, \ldots, e_k$ is a path of inactive edges with $(k-1)d \geq (\alpha - 2 - j)d$ and let e_{k+1} be such that e_k is inactive due to e_{k+1}. e_{k+1} was not active at the end of the regular subphase $j-1$, otherwise e_k is definitively inactive at the end of this subphase and would not participate to the competition between edges. We had so a path of inactive edges $e_1, \ldots, e_{k+1}$ at the subphase $j-1$, e_{k+1} is inactive due to e_{k+2} which is inactive at the end of the subphase $j-2$ and so on. We have a path of size $k+j-1$ at the first regular subphase and so $k-1 \leq \alpha - 2 - j$. We have a contradiction. □

In summary we have an efficient algorithm with a control overhead of order $(log_2(K))^2$.

5 Stability Criteria

Assumption 1. *For each $e \in E$:*

(a) $\overline{A}(e) := E[A_t(e)] < \infty$, $\overline{A^2}(e) := E[A_t^2(e)] < \infty$.

(b) $\sum_{e' \in \varepsilon(e) \cup \{e\}} \overline{A}(e') < c_{min}(e)$ *where* $c_{min}(e) := min_{e' \in \varepsilon(e) \cup \{e\}}(c(e'))$

Conjecture 1. *Let $\{\mathbf{q}_t, t \geq 0\}$ be a (irreducible) Markov chain (MC). Then, $\{\mathbf{q}_t, t \geq 0\}$ is stable under Assumption 1.*

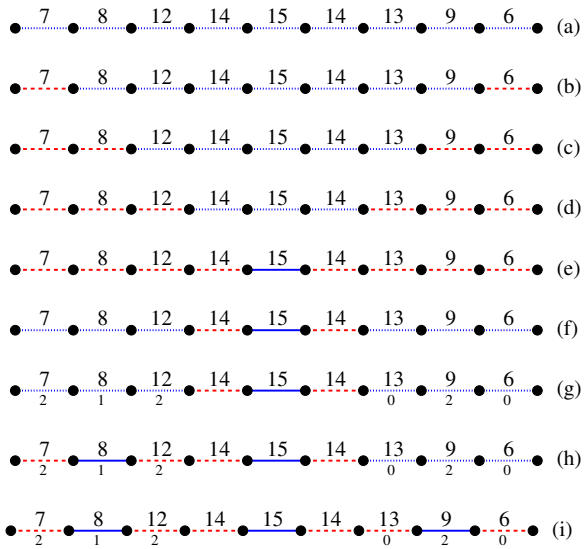


Figure 3: Algorithm Log 1 for a 10-nodes path with $d = 0$ and $K = 16$: regular subphase (a-e) and random subphase (f-i).

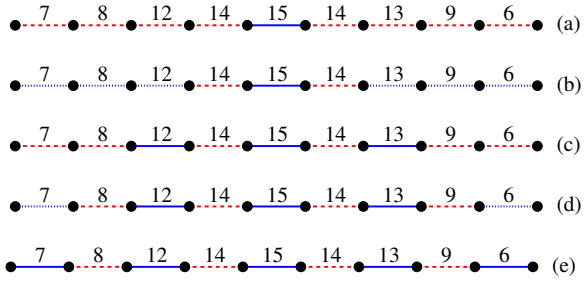


Figure 4: Algorithm Log 2 for a 10-nodes path: $d = 0$, and $K = 16$.

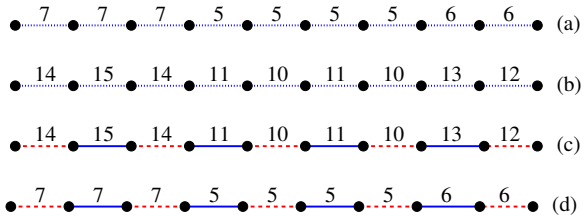


Figure 5: Algorithm Log 2 for a 10-nodes path: $d = 0$, $K = 8$, and $c_2 = 2$.

6 Simulation Results

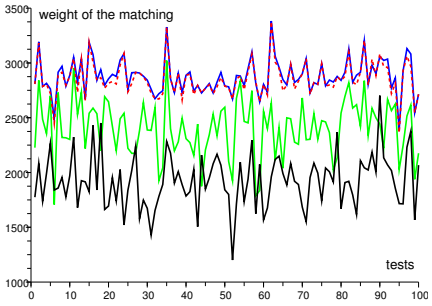
We present here some simulation results. We have implemented some classical topologies like paths and grids, and also random graphs representing real wireless networks (see an example of random transmission graph in Section 7). We have implemented centralized algorithms and some distributed like the one proposed in [6] (Section 3), Algorithm Logs described in Section 4 and one fully random algorithm. Results are presented here only for the primary node model.

6.1 One Step Efficiency

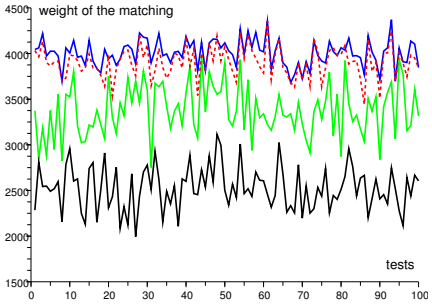
We describe here efficiencies of different algorithms for a single step (without packet arrivals) given a distribution (randomly uniform) of the weights q. We compare the sum of the weights of active edges for the different implemented algorithms. Fig. 6(a) and Fig. 6(b) represent the weight of the matching for each of the algorithm for 100 tests with random uniform weights, respectively for a 100-edges path and for a 10 by 10 grid. Algorithm Log 2 and the centralized algorithm give almost the same weight of the matching for both topologies. Algorithm Log 1 gives a total weight of active links greater than the ratio of a random algorithm matching for the grid (Fig. 6(b)) and for the path with only the regular subphase (Fig. 6(a)).

6.2 Stability Results

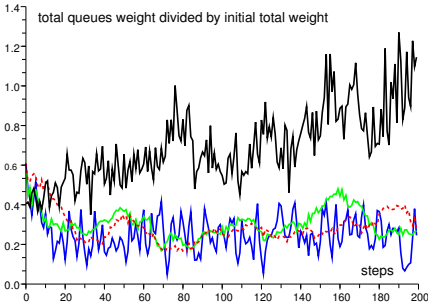
We study here the stability of the queueing system of the network simuling consecutive steps with packet arrivals. Fig. 6(c) shows the stability of our proposed algorithms and the centralized algorithm for a 100-edges path with a capacity $c(e) = 30$ for each link $e \in E$, an average arrival number equals to 12 (uniformily between 0 and 24) and 200 steps. The random algorithm is not stable and the ratio (total current weight over initial) is greater than 1.5 for the distributed algorithm proposed in [6]. Its convergence is very slow.



(a) Weight of the matching at one step for a 100-edges path, random weights, $d = 0$ et 100 tests.



(b) Weight of the matching at one step for a 10 by 10 grid, uniform weights, $d = 0$ et 100 tests.



(c) $\sum_{e\in E} q_t(e)/\sum_{e\in E} q_0(e)$ for a 100-edges path, $c(e) = 30$, 200 steps ($\forall e \in E$, $\overline{A}(e) = 12$).

Figure 6: Simulation results: centralized algorithm (blue), Algorithm Log 1 (green), Algorithm Log 2 (dotted red) and one random algorithm (black).

7 Discussion

In this section we describe some possible generalizations and futur works such that dealing with multi-hop traffic and queues located on nodes.

7.1 Multi-hop traffic

We can adapt Algorithms Log to take into consideration multi-hop traffic. From Algorithm Log 1 (respectively Algorithm Log 2) we can change previous weights (respectively virtual weights) into multi-hop weights $q_t^m(e)$ (respectively virtual multi-hop weights $q_t^{m\prime}(e)$). More precisely $q_t^m(e) = \sum p_t^i(e).h_t^i(e)$ where $p_t^i(e)$ is the packet numbered i in the queue of link $e \in E$ at step t and $h_t^i(e)$ is the number of remaining hops of the previous packet. Furthermore $q_t^{m\prime}(e) = c_2.q_t^m(e) + g(t,e)$. We assumed here that the routing is pre-computed. The policy of services can changed with application requirements.

7.2 Queues on nodes

We can also adapt Algorithm Log to weights on nodes instead of weights on links (edges). To do it, it is possible to get a similar algorithm with an overhead $T' = 3.T_2$ (recall that T_2 is the number of control time-slots of Algorithm Log 2). Thus the overhead remains constant. We can explain easily the new version of Algorithm Log with weights on nodes using an example where $d = 0$. Figure 7 represents a *9-nodes path* where almost each node has 2 different weights corresponding to its 2 queues (except of course the left and right node). One version of new Algorithm Log is that each node chooses the maximum queue length (see Figure 7(b)). After that each node computes its vector (the same computation than the previous Algorithm Log). At step (c), each node sends a message and so all nodes remains undetermined. At step (d), the nodes of weight 6 or 7 send a message. The right node of weight 6 has sent a message to 5 and 4 with the identification number of the node of weight 4. This one sends a message to 6 to confirm its request and 6 sends again a message to confirm. Thus node 6 actives the link (Figure7(e)). For other nodes, there is a problem of interference and so only inactive some nodes (of weight 4 and 5). In the same way, two other links are activated (Figure 7(f,g,h,i)). As for Algorithm Log 2, this protocol is repeated and at the end there are 4 activated links (Figure 7(j,k)).

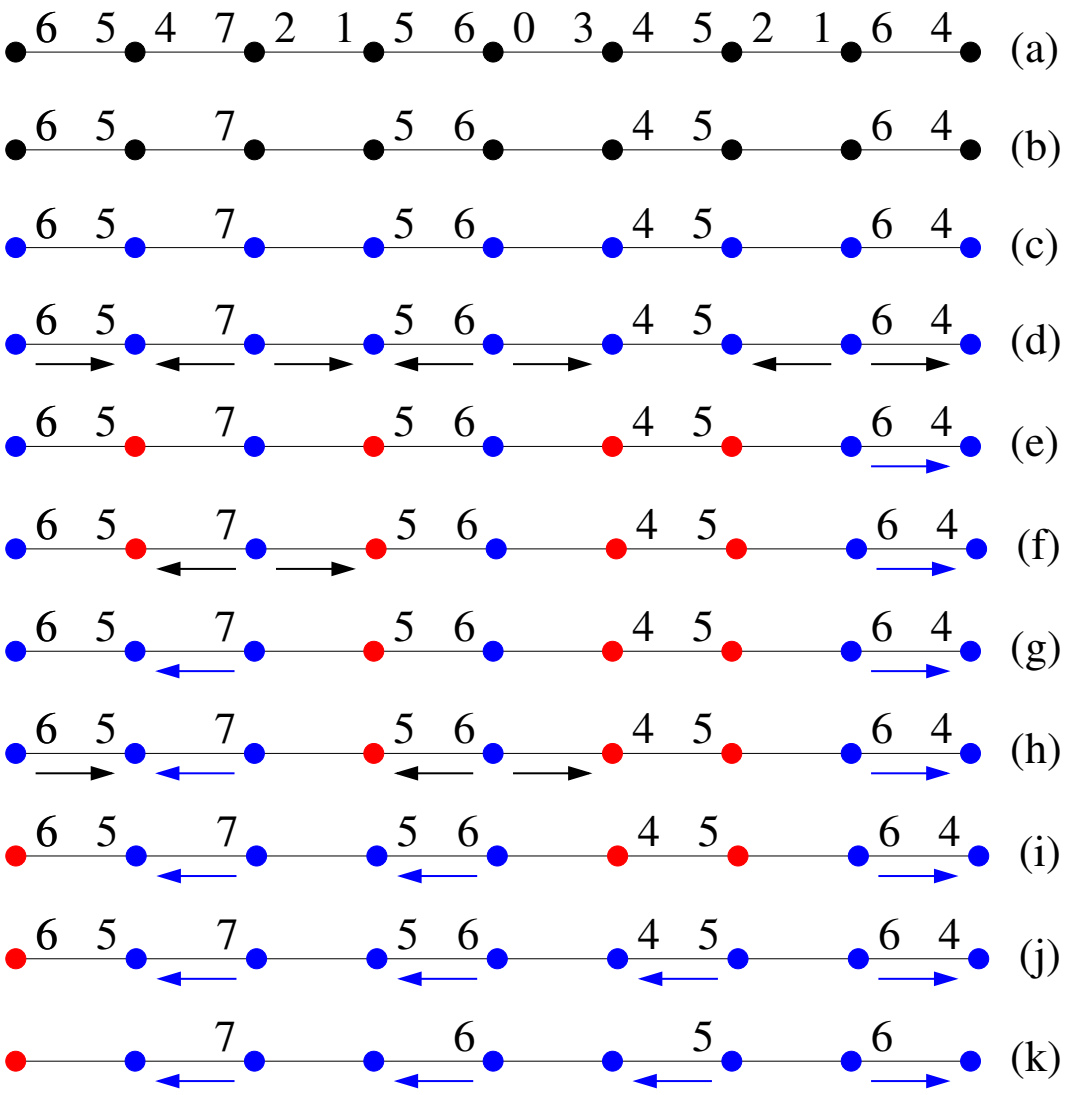


Figure 7: Generalization of Algorithm Log for a 9-nodes path, $d = 0$ and $K = 8$.

8 Conclusion

We proposed in this article two distributed algorithms for the call scheduling problem in wireless networks. They are the first proposed algorithms with a constant overhead and working for any binary interference model. The second is optimal in sense that the set of active links found is maximal. Furthermore Simulation results prove their efficiency in terms of throughput and stability. We have also a sufficient condition of stability for Algorithm Log 2: if $\forall e \in E$, $\sum_{e' \in \varepsilon(e) \cup \{e\}} \overline{A}(e') < c_{min}(e)$ with $c_{min}(e)$ the smallest capacity among edges in $\varepsilon(e)$. Under this condition, the associated Markov chain is stable. Finally we can adapt our algorithms to queues located on nodes and also taking into account multi-hop traffic.

References

[1] Andrew Brzezinski, Gil Zussman, and Eytan Modiano. Enabling distributed throughput maximization in wireless mesh networks: a partitioning approach. In *MobiCom*, pages 26–37. ACM, 2006.

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

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

[4] Ralf Klasing, Nelson Morales, and Stéphane Pérennes. On the complexity of bandwidth allocation in radio networks. *Theoretical Computer Science*, 406(3):225 – 239, 2008. Algorithmic Aspects of Global Computing.

[5] Eytan Modiano, Devavrat Shah, and Gil Zussman. Maximizing throughput in wireless networks via gossiping. *SIGMETRICS Perform. Eval. Rev.*, 34(1):27–38, 2006.

[6] Sujay Sanghavi, Loc Bui, and R. Srikant. Distributed link scheduling with constant overhead. In *SIGMETRICS*, pages 313–324, 2007.

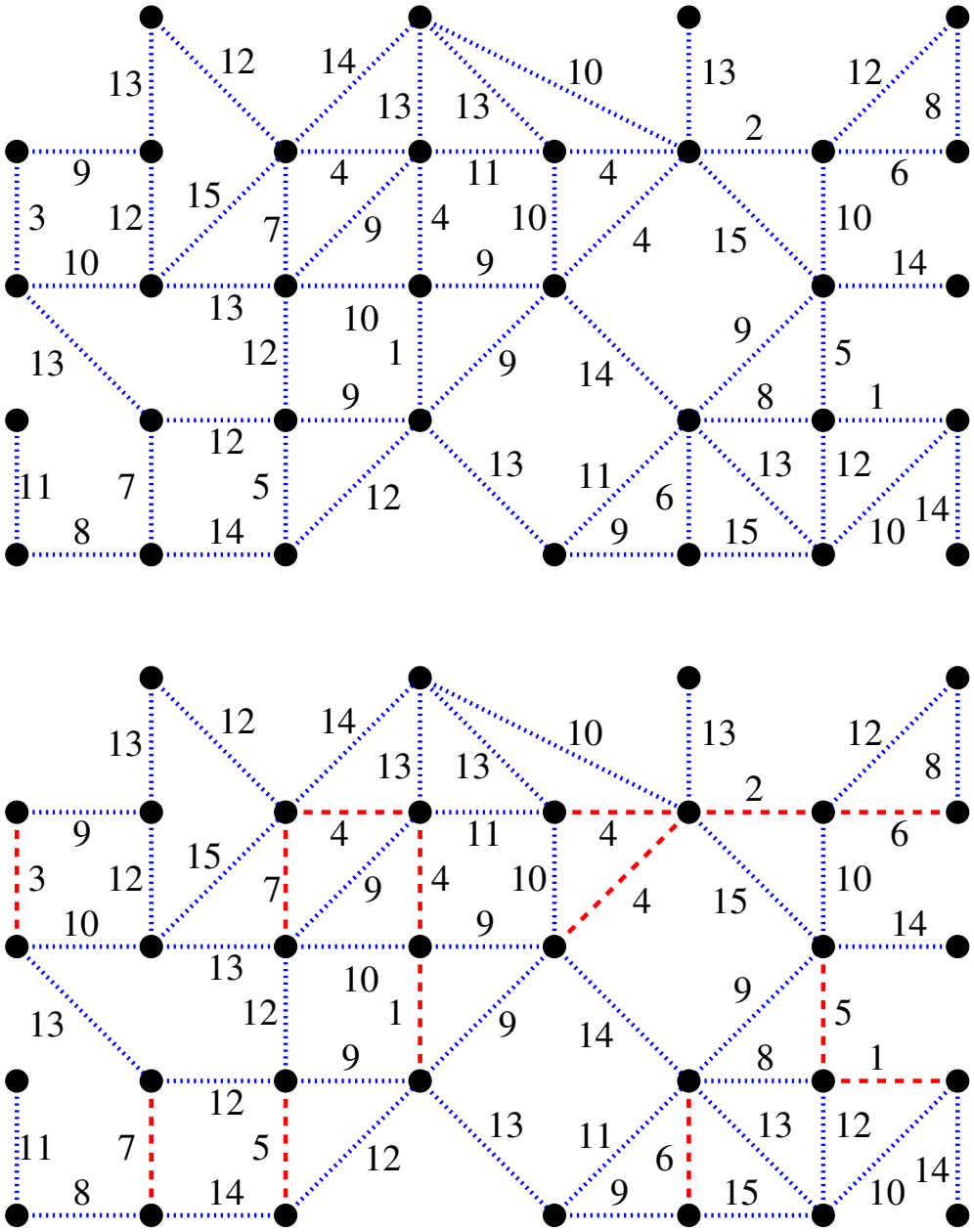
[7] L. Tassiulas. Scheduling and performance limits of networks with constantly changing topology. *IEEE Transactions on Information Theory*, 43(3):1067–1073, 1997.

[8] 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.

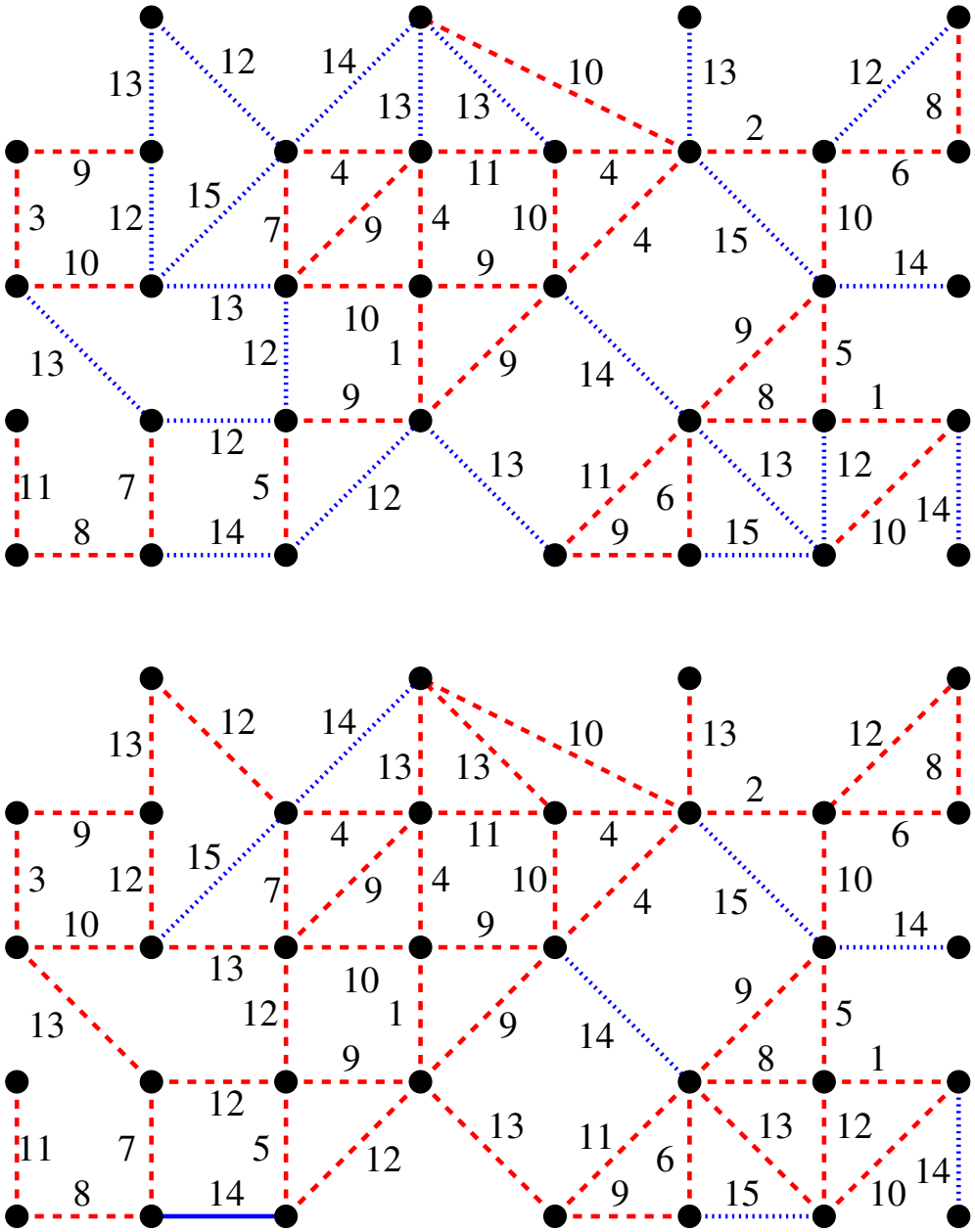
9 Appendix

We can see (in the above figure) the regular subphase of Algorithm Log 1 for a random transmission graph $G = (V, E)$ with the interference model $d = 1$.

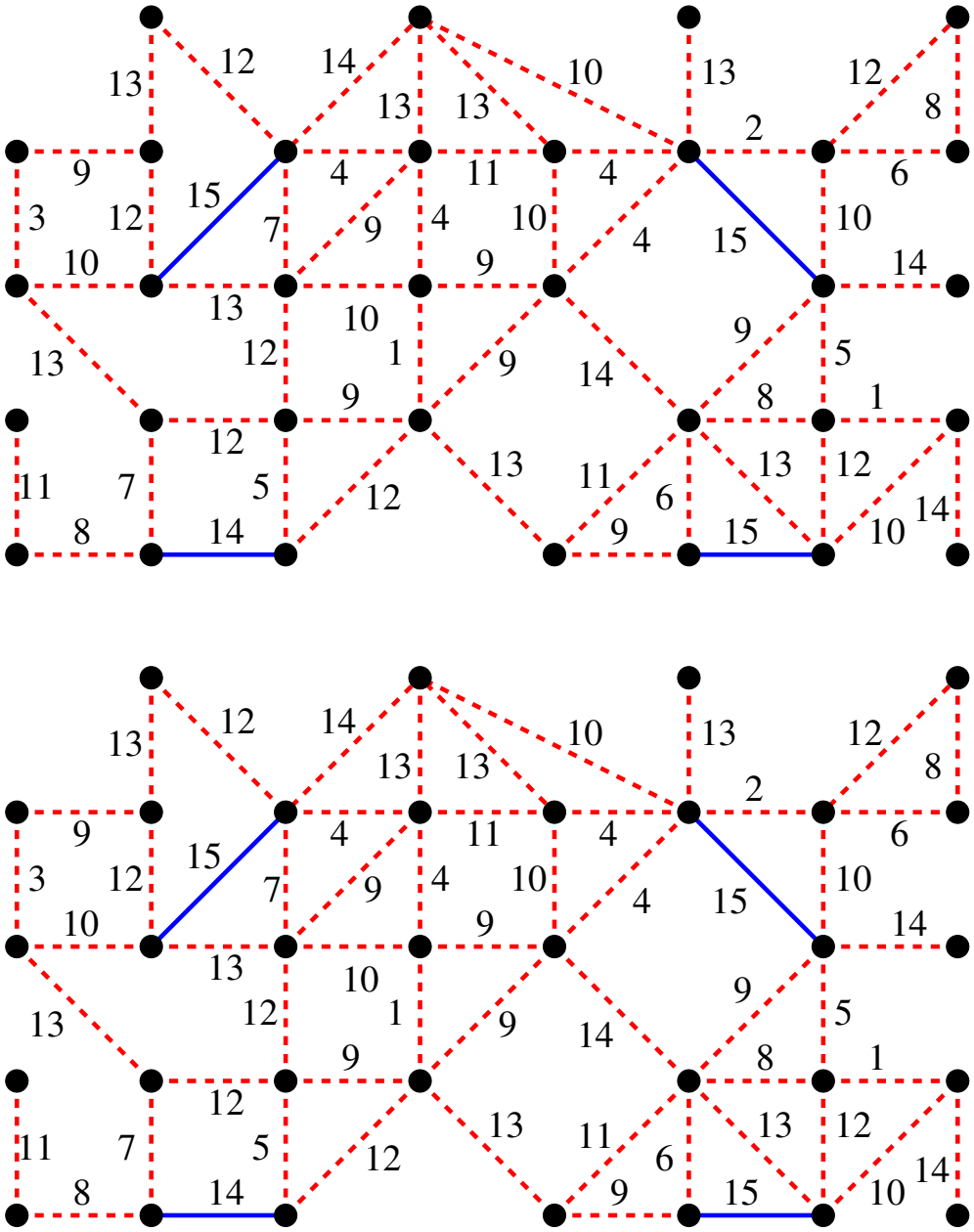
- **(a)** The first subfigure represents the transmission graph before the control subphase of Algorithm Log 1 of step t. Each edge is undetermined, represented in G by blue dotted-lines.
- **(b)** Each edge $e \in E$ such that $v_{t,e}(1) = 1$ sends a control message, that is to say each edge with weight in the finite set $[8, 15]$. In this example, each of these edges stays undetermined because it receives at least one control message. All others (with weight in the set $[0, 7]$ become inactive (red).
- **(c)** Undetermined edges $e \in E$ such that $v_{t,e}(2) = 1$ send a control message. It corresponds to edges with weight in $[12, 15]$.
- **(d)** Before the third control time-slot, edges with weight 14 or 15 are the unique undetermined edges. Others are inactives. After this time-slot, we have one active, the edge 14.
- **(e)** After time-slot 4, we have a maximal set of active edges corresponding to the interference model $d = 1$. This set represents in the transmission graph an induced matching.
- **(f)** Thus the last time-slot does not influence in this example the choice of the active set.


13 12 14 10 13 12
13 13 2 8
9 4 11 4 6
3 12 15 7 9 4 10 4 15 10
10 9 14
13 10 9
13 12 1 9 14 5
9 8 1
12 13 11 13 12
11 7 5 12 6 14
8 14 9 15 10

13 12 14 10 13 12
13 13 2 8
9 4 11 4 6
3 12 15 7 9 4 10 4 15 10
10 9 14
13 10 9
13 12 1 9 14 5
9 8 1
12 13 11 13 12
11 7 5 12 6 14
8 14 9 15 10


13 12 14 10 13 12 8
13 13 2
9 4 11 4 6
3 12 15 7 9 4 10 4 15 10
10 9 14
13 10 9
13 12 1 9 14 9 5
9 8 1
12
11 7 5 12 13 11 13 12
6 14
8 14 9 15 10

13 12 14 10 13 12 8
13 13 2
9 4 11 4 6
3 12 15 7 9 4 10 4 15 10
10 9 14
13 10 9
13 12 1 9 14 9 5
9 8 1
12
11 7 5 12 13 11 13 12
6 14
8 14 9 15 10



13 12 14 10 13 12
13 13 2 8
9 4 11 4 6
15
3 12 7 9 4 10 4 10
10 15
9 14
13 10 9
13 12 1 9 14 5
9 8 1
12
11 7 5 12 13 11 13 12
6 14
8 14 9 15 10

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