Les Annales RECITS (http://www.lrecits.usthb.dz)
Vol. 02, 2015, pages 15-26

Scheduling with agreements: new results

Mohamed Bendraouche[1,2] and Mourad Boudhar[2]

[1] Faculty of Sciences, Saad Dahleb University, Route de Soumaa, BP 270, Blida, Algeria
[2] RECITS Laboratory, Faculty of Mathematics, USTHB University, BP 32 El-Alia, BEZ, Algiers, Algeria

mbendraouche@yahoo.fr , mboudhar@usthb.dz

Abstract: In this paper, the problem of scheduling with agreements (SWA) is considered. This consists in scheduling a set of jobs non-preemptively on identical machines subject to constraints that only some specific jobs can be scheduled concurrently on different machines. These constraints are given by an agreement graph and the aim is to minimize the makespan. In the case of two machines we extend two NP-hardness results of SWA with processing times at most 3 that hold for bipartite agreement graphs to any class of agreement graphs with a specific structure. Complexity results of SWA are established in the case of split and complement of bipartite graphs. Finally we present some approximation results for SWA.

Keywords: Scheduling, agreement graph, makespan, complexity, approximation.

Résumé : Nous considérons le problème d'ordonnancement avec concordance. Il consiste à ordonnancer un ensemble de tâches sans interruption sur des machines identiques sous la contrainte que quelques tâches spécifiques peuvent être ordonnancées simultanément sur des machines différentes. Ces contraintes sont représentées par un graphe. Dans le cas de deux machines, deux résultats de NP-complexité sont étendus. De nouveaux résultats, dans les cas d'un graphe scindé et du complémentaire d'un graphe biparti, sont établis. Des résultats d'approximation sont aussi présentés.

Mots clés : Ordonnancement, graphe de concordance, makespan, complexité, approximation.

1 Introduction

We recall the problem of scheduling with agreements (SWA in short) [2]: the input consists of a set $V = \{J_1, J_2, .., J_n\}$ of n jobs with processing and release times p_i and r_i respectively $(i = 1, 2, ..., n)$, and a set of m identical machines. Each machine can process at most one job at a time and each job can be processed on only one machine. The preemption of the jobs is not allowed. We assume that there is a graph $G = (V, E)$ over the jobs, called the agreement graph. Each edge in E models a pair of agreeing jobs that can be scheduled concurrently (on different machines). The agreement constraints are such that only the agreeing jobs can be scheduled concurrently. A schedule is an assignment of jobs to the machines which specifies for each job the time interval and the machine on which this job is to be processed. A feasible schedule is a non-preemptive one which respects the agreement constraints. The aim is to find a feasible schedule that minimizes the makespan.

The SWA problem can be found in the resource-constrained scheduling problems when the resources are non-sharable. A motivation of this problem is the school exam planning: at school, there are n exams $E_1, .., E_n$ to be scheduled at the end of the half-year. Each exam has to be taken by a set of students and lasts either one or two hours. The exams are taken in class-rooms whose number is m. Also we suppose that the capacity of a class-room is big enough so that any exam can be taken in any class-room. We seek a schedule which minimizes the length of the examination period. This problem can be modelised as follows: to each exam E_i corresponds a job J_i such that the processing time of J_i equals the time taken by its corresponding exam E_i. We regard the class-rooms as being the machines. The agreement graph $G = (V, E)$ is such that $V = \{J_1, J_2, ..., J_n\}$ and a pair $\{J_i, J_j\}$ is an edge if and only if there are no students who take both exams E_i and E_j at a time. This problem can be formulated as the SWA problem in which the agreement graph is $G = (V, E)$ with processing times in {1,2}.

When restricted to unit processing times, the SWA problem is equivalent to finding a partition of a given graph into a minimum number of cliques, each with size at most m. This problem is equivalent to the mutual exclusion scheduling problem introduced by [1], by considering the complement of the agreement graph, called the conflict graph. Many results concerning this problem can be found in the literature (see for example [2, 4, 9, 11, 13, 14]).

Scheduling with agreement graph or scheduling with agreements (SWA) is equivalent to scheduling with conflict graph (the complement of the agreement graph). The latter is known as scheduling with conflicts which was first studied by Irani and Leung [12] and then by Chrobak et al. [7]. Even et al. [8] have also worked with conflict graphs and have considered the SWA problem for fixed m under the name: scheduling with conflicts (SWC in short). In the case of two machines Even et al. [8] have proposed a polynomial time algorithm when the processing times are equal to 1 or 2 and have proved that it is APX-hard when $p_i \in \{1, 2, 3, 4\}$. Bendraouche and Boudhar [2] have established that the SWA problem is strongly NP-hard in the case of two machines when $p_i \in \{1, 2, 3\}$, even for arbitrary bipartite agreement graphs. In a recent paper [3], the authors have thoroughly closed the complexity status of SWA on two machines with two fixed processing times.

The remainder of this paper consists of five sections and is organized as follows. Section 2 presents the generalization of two NP-hardness results of SWA in the case of two machines, processing times at most 3 for bipartite agreement graphs to any class of agreement graphs with a specific structure. In Section 3, a polynomial result in the case of split agreement graphs is established. Section 4 is devoted to complexity results related to complement of bipartite agreement graphs. In Section 5, we establish some approximation results. Concluding remarks constitute the last section.

2 NP-hardness generalizations for two machines

Although for a given agreement graph the SWA problem with three values of processing times is a subproblem of the two values-case, it is still interesting to know the complexity of the former for different classes of agreement graphs. For this purpose we establish the following results.

2.1 Case $p_i \in \{1, 2, 3\}$

As it has previously been mentioned, the SWA problem with two machines and $p_i \in \{1, 2, 3\}$ is strongly NP-hard for arbitrary bipartite agreement graphs. We generalize this result to any class of arbitrary agreement graphs with a specific structure, as stated in the following theorem.

Theorem 1 *The SWA problem with two machines and $p_i \in \{1, 2, 3\}$ is strongly NP-hard for any class of agreement graphs $G = (A \cup B, E)$ where A is an independent set with $|A| = 2a$ and B is a set of cardinality $2a - b$ inducing a subgraph with a specific structure ϕ. The parameters a and b are two arbitrary integers satisfying $b \leq a$.*

Proof. Consider a class C_ϕ of agreement graphs of the form cited, associated with a specific structure ϕ. For example ϕ can either be a discrete, complete, bipartite, or split graph-like structure. We use a similar reduction from the 3-Dimensional Matching problem(3-DM) as it has been done in [2] for the case of arbitrary bipartite graphs $G = (S_1, S_2; E)$. The agreement graph of the scheduling instance is similar: $G = (S \cup W, E)$ where $S = V_Y \cup V_Z \cup V_D \cup V_R$ and $W = V_M \cup V_Q$. With respect to the bipartite version the sets W and S play the role of S_1 and S_2 respectively. The edge-set E is constructed as follows. The edges of E between S and W are constructed similarly as those between S_1 and S_2 in the bipartite case. S is an independent set in G. The only difference is that we impose the subgraph induced by W to have the structure ϕ. Clearly we have $|S| = 2|M|$ and $|W| = 2|M| - q$. Note that $|S|$ and $|W|$ are of the form $2a$ and $2a - b$ respectively $(a = |M|, b = q)$ such that $b \leq a$, and this makes the agreement graph G belong to the class C_ϕ.

Next we show that 3-DM has a matching M' if and only if there is a feasible schedule σ with makespan $C_{\max}(\sigma) \leq 4|M| - 2q$.

For the necessary part, the construction of a feasible schedule σ satisfying $C_{\max}(\sigma) \leq 4|M| - 2q$, is the same as in the bipartite case [2].

Conversely, suppose there is a feasible schedule σ with $C_{\max}(\sigma) \leq 4|M| - 2q$. Since the total processing times equals $8|M| - 4q$, then all the jobs are scheduled on both machines without idle times and $C_{\max}(\sigma) = 4|M| - 2q$.

We claim that for every set V_i $(i = 1, ..., q)$, there are $(|M_i| - 1)$ quadruplets of the form $\{R_{i,k}, Q_{i,k}, D_{i,k}$, one job of $V_{M_i}\}$ $(k \in \{1, 2, ..., |M_i| - 1\})$ which must have been scheduled either as shown in Figure 1 or in its symmetric configuration.

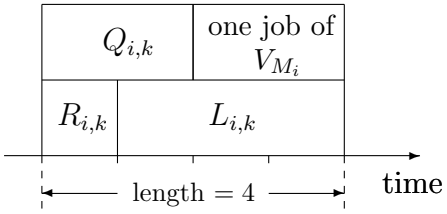

Figure 1: Schedule of the jobs of V_i

To prove this, let $R_{i,k}$ be a job of V_{R_i}, therefore the corresponding job $Q_{i,k}$ of V_{Q_i} must have been scheduled in parallel with $R_{i,k}$ as shown in Figure 2 or in its symmetric configuration. Consider the case of Figure 2, the symmetric case can be done similarly.

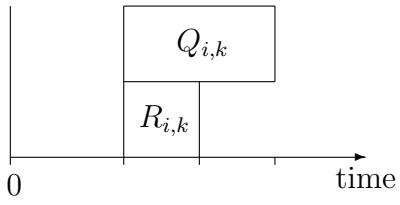

Figure 2: Gantt diagram of case 1

By construction of the agreement graph only four cases may have happened:

case 1: $Q_{i,k}$ has been scheduled in parallel with some job $Q_{j,k'}$ of some set V_{Q_j}. Here two subcases may have happened:
case 1.1: The job $R_{j,k'}$ of V_{R_j} which is in correspondence with the job $Q_{j,k'}$ has been scheduled in parallel with $Q_{j,k'}$, as shown in Figure 3.
The time left is equal to $4|M| - 2q - 3$. On the other hand, the remaining independent jobs not represented in the Gantt diagram of this figure are: $(|M| - q - 2)$ jobs of V_R and all the jobs of V_D, V_Y and V_Z. The minimum time that has been taken by all of these jobs is: $|M| - q - 2 + 3(|M| - q) + 2q = 4|M| - 2q - 2 > 4|M| - 2q - 3$. This implies that the remaining independent jobs just cited could not have been fitted in the time left, therefore this subcase hasn't happened.

case 1.2: The job $D_{j,k'}$ of V_{D_j} which is in correspondence with the job $Q_{j,k'}$ has been scheduled in parallel with $Q_{j,k'}$ and thus the job $D_{j,k'}$ must have been scheduled in parallel

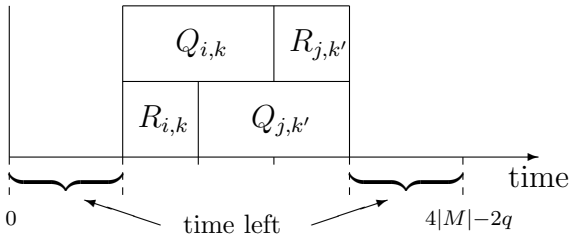

Figure 3: Gantt diagram of case 1.1

with some job of V_{M_j}, see Figure 4.
The remaining independent jobs not represented in the Gantt diagram of this figure are: $(|M| - q - 1)$ jobs of V_R, $(|M| - q - 1)$ jobs of V_D and the $2q$ jobs of V_Y and V_Z. These require a total time of at least $|M| - q - 1 + 3(|M| - q - 1) + 2q = 4|M| - 2q - 4 > 4|M| - 2q - 5$ (that is the time left), contradiction, so this subcase hasn't occurred either.

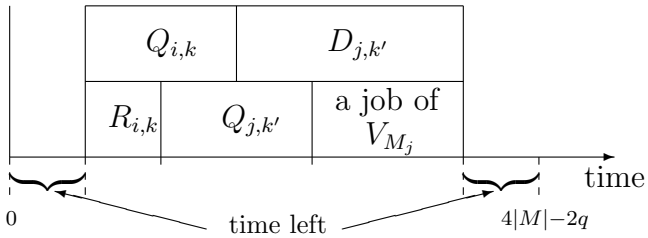

Figure 4: Gantt diagram of case 1.2

case 2: $Q_{i,k}$ has been scheduled in parallel with some job of V_M. The maximum time left is equal to $4|M| - 2q - 2$ (see Figure 5). The remaining independent jobs not represented in this figure are: $(|M| - q - 1)$ jobs of V_R and all the jobs of V_D, V_Y and V_Z. The time that has been taken by all of these jobs is: $|M| - q - 1 + 3(|M| - q) + 2q = 4|M| - 2q - 1 > 4|M| - 2q - 2$. Therefore the independent jobs just cited could not have been scheduled in the maximum time left and thus this case hasn't happened. We deduce that the only possibility that has happened is the following case.

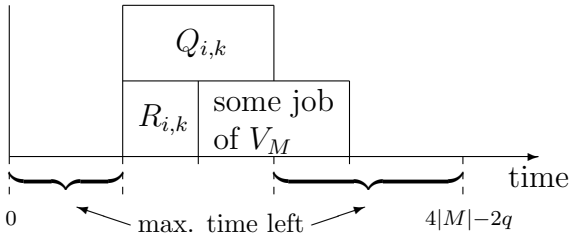

Figure 5: Gantt diagram of case 2

case 3: The job $D_{i,k}$ of V_{D_i} corresponding to the job $Q_{i,k}$ has been scheduled in parallel with $Q_{i,k}$ and some job of V_{M_i} must have been scheduled in parallel with $D_{i,k}$, see Figure 1. The proof of our claim is terminated. With the same argument as in [2], one puts in evidence the required matching. ■

Remark 1 *In the case of two machines, it is worth noting that one can derive many new results by considering different structures ϕ, for instance if ϕ is a discrete graph-like structure, then C_ϕ is a subclass of the class of bipartite agreement graphs and therefore we re-obtain the old NP-hardness result cited in reference [2], for the class of bipartite agreement graphs. If ϕ is a complete graph-like structure, this similarly implies that SWA is strongly NP-hard when $p_i \in \{1, 2, 3\}$ for the class of split graphs. Since the latter is a subclass of chordal graphs, we also have an analogous result for chordal graphs. In case ϕ is a bipartite graph-like structure, we deduce an analogous NP-hardness result for the class of 3-partite agreement graphs.*

2.2 Case $p_i \in \{1, 2\},\ r_i \in \{0, r\}$ (r arbitrary)

We recall that in reference [2], it has been proved that when $m = 2$, $p_i \in \{1, 2\}$ and $r_i \in \{0, r\}$ (r arbitrary), the SWA problem is strongly NP-hard for arbitrary bipartite agreement graphs.
An analogous NP-hardness generalization as in theorem 1, can also been obtained in the case $m = 2$, $p_i \in \{1, 2\}$ and $r_i \in \{0, r\}$ (r arbitrary), bipartite agreement graphs, and this is stated in the following theorem.

Theorem 2 *The SWA problem with two machines, $p_i \in \{1, 2\}$ and $r_i \in \{0, r\}$ (r arbitrary) is strongly NP-hard for any class of agreement graphs $G = (A \cup B, E)$ where A is an independent set with $|A| = 2a$ and B is a set of cardinality $2a - b$ inducing a subgraph with a specific structure. The parameters a and b are two arbitrary integers satisfying $b \leq a$.*

Proof.

Here again we consider the similar reduction from 3-DM, used in Theorem 2 of reference [2], for bipartite agreement graphs. The corresponding scheduling instance is similar, except that the agreement graph is $G = (S \cup W, E)$ where $S = V_Y \cup V_Z \cup V_D$ and $W = V_M$. The edge-set E is constructed in such a way that S is an independent set and that the subgraph induced by W possesses the structure ϕ.

Next we show that 3-DM has a matching M' if and only if there is a feasible schedule σ with makespan $C_{\max}(\sigma) \leq 2|M|$.

The necessary condition is the same as in the bipartite case.

Conversely, in a similar way we show that $|M| - q$ jobs of W must have been scheduled in the time interval $[2q,\ 2|M|]$, opposite to those of V_D, see Figure 5 of reference [2]. As for the $3q$ remaining jobs, the possibility that two jobs of W have been scheduled in parallel cannot happen since the jobs of the set $V_Y \cup V_Z$ could not have been scheduled. Consequently, this gives the required matching. ■

The same previous remark can be drawn with respect to this theorem, in particular we re-obtain the old analogous NP-hardness result of SWA, reference [2], with two machines, $p_i \in \{1, 2\}$, $r_i \in \{0, r\}$ (r arbitrary) and bipartite agreement graphs.

3 Case of Split graphs with unit processing times: a polynomial result

By Remark 1, one can deduce that the SWA problem is strongly NP-hard for arbitrary split agreement. Arbitrary split graphs are denoted $G = (K, S; E)$, where K and S are the clique and the independent set of G respectively. Next we prove that it becomes polynomial when the jobs of the clique K have unit processing times and those of the independent set S are arbitrary.

Proposition 3 *The SWA problem for arbitrary split agreement graphs $G = (K; S, E)$, can polynomially be solved in $O(n^3)$ when the processing times of the jobs of K are equal to one and those of S are arbitrary.*

This problem can be reduced to the maximum flow problem, in the same way as it has been done in [2], in the case of two machines and bipartite graphs $G = (S_1, S_2)$ where the jobs of S_1 have unit processing times. The same Algorithm 1 can be adjusted for the problem at hand by replacing S_1 by the clique K and S_2 by the stable set S, except for some changes in such a way that the main steps of the new Algorithm 1 will take the form:

Algorithm 1
Begin

1: Construct the network R similarly, except that an arc J_i from S to p has capacity $p_i(m-1)$.
2: Determine a maximum feasible flow f^* in R.
3: In Schedule the jobs of S successively in the time interval $[0, \sum_{i=1}^{|S|} p_i]$, on $P1$.
4: In the interval $[0, \sum_{i=1}^{|S|} p_i]$, schedule the entering jobs of K with unit flow value into blocks successively on the $m-1$ machines, such that each block is scheduled with its corresponding job of S.
5: Schedule the remaining jobs of K successively and alternately on the m machines, in the time interval $[\sum_{i=1}^{|S|} p_i, \sum_{i=1}^{|S|} p_i + \lceil \frac{|K| - f^*(u_r)}{m} \rceil]$.

end

Note that step 4 can be implemented in $O(n)$, thus the complexity of this algorithm is $O(n^3)$.

Remark 2 *This theorem generalizes a result of Bodlaender and Jansen (1995).*

4 Case of complement of bipartite agreement graphs, $m = n$, $p_i = 1$

In this section, we consider the SWA problem with unit processing times when the agreement graph is an arbitrary complement of a bipartite graph which is denoted by $G = (K_1, K_2; E)$, where K_1, K_2 are two cliques of the graph G. We also suppose that $m = n$.

4.1 Case $r_i \in \{0, 1, 2\}$

The condition $m = n$ means that there are no processor constraints, then the problem at hand is equivalent to the minimum coloring problem for arbitrary bipartite graphs which is known to be polynomial.

Next we prove that if further we add the release time condition $r_i \in \{0, 1, 2\}$, we get a strongly NP-hard problem.

Theorem 4 *The SWA problem with arbitrary complement of bipartite agreement graphs, unit processing times and $r_i \in \{0, 1, 2\}$ is strongly NP-hard even if $m = n$.*

Proof. Consider the 1-Precoloring Extension problem for bipartite graphs (1-PrExt for short): **Instance:** a bipartite graph $G = (S_1, S_2; E)$ and 3 vertices $v_1, v_2, v_3 \in S_1$. **Question:** is there a 3-coloring of G such that v_i receives color i, for all $i (i = 1, 2, 3)$?
This problem is NP-complete [5]. We prove that the latter can polynomially be reduced to the following problem (D'):
instance: let $m = n$ and let $G' = (K_1; K_2 \cup \{u_1, u_2, u_3\}, E')$ be composed of $\overline{G} = (K_1, K_2; \overline{E})$ so that K_1 and K_2 are two cliques of G' corresponding to S_1 and S_2 respectively, and three dummy jobs u_1, u_2, u_3 such that $K_2 \cup \{u_1, u_2, u_3\}$ is also a clique in G'. Other edges of E' are defined as follows: for each i $(i = 1, 2, 3)$, u_i is adjacent to v_i and to all jobs of $K_1 \backslash \{v_1, v_2, v_3\}$. Thus K_1 and $K_2 \cup \{u_1, u_2, u_3\}$ are two cliques of G' making it the complement of a bipartite graph see Figure 6 (a). All the processing times of the jobs of G' are equal to 1. $r_{v_i} = r_{u_i} = i - 1$ for all i $(i = 1, 2, 3)$, the release times for the rest of the jobs are zero.
Question: does there exist a feasible schedule σ with $C_{\max}(\sigma) \leq 3$?

Suppose there is a 3-coloring (C_1, C_2, C_3) of G such that $v_i \in C_i$ for all i $(i = 1, 2, 3)$. We construct the schedule σ as follows: the jobs of C_i and the job u_i are scheduled at time $t_i = i - 1$ for each i $(i = 1, 2, 3)$. The schedule σ is represented in Figure 6 (b). One can verify that σ is a feasible schedule of the problem D' satisfying $C_{\max}(\sigma) \leq 3$.

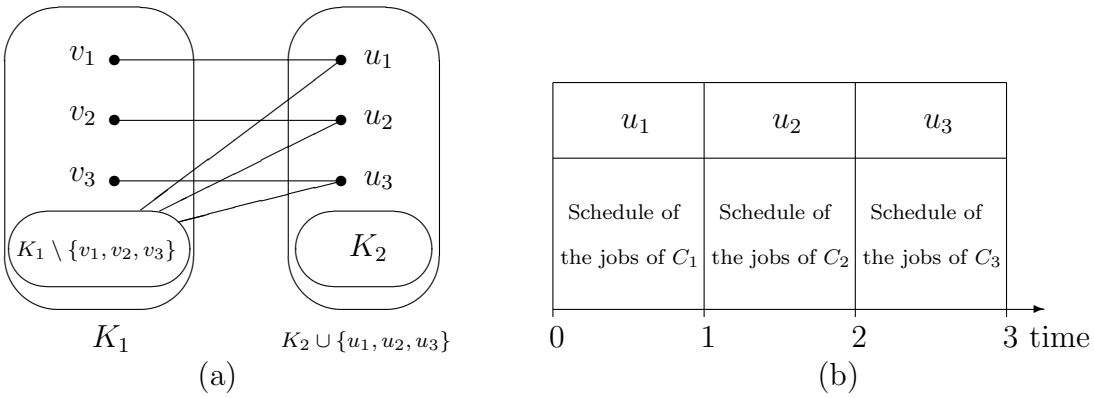

Figure 6: (a) The agreement graph G' (b) The Gantt diagram representing σ

Conversely, suppose there exists a feasible schedule σ of D' with $C_{\max}(\sigma) \leq 3$. With unit processing times, one can restrict itself to the case where jobs start at integer dates. This can be shown by rearranging jobs from left to right if they have non-integer starting dates. The jobs u_3 and v_3 having release times 2, they must have been scheduled in the same time interval $[2, 3]$. By construction, the jobs u_i and v_i must have been scheduled in $[i-1, i]$ for each i $(i = 1, 2)$. Then for each i $(i = 1, 2, 3)$ let C_i= {jobs (different from u_i) scheduled at time $i-1$}. $r_{u_3} = r_{v_3} = 2$ implies that u_3, v_3 must have been scheduled in $[2, 3]$. $r_{u_2} = r_{v_2} = 1$, this means that u_2, v_2 have been scheduled in $[1, 3]$. On the other hand u_2 is not adjacent to v_3 in G' therefore u_2 must have been scheduled in $[1, 2]$. Similarly v_2 is not adjacent to u_3 in G' therefore v_2 must have been scheduled in $[1, 2]$. With the same argument we prove that u_1, v_1 have been scheduled in $[0, 1]$. The sets (C_1, C_2, C_3) correspond to cliques in G', therefore they form a 3-coloring of G such that $v_i \in C_i$ for all i $(i = 1, 2, 3)$. It can be verified that this reduction is polynomial and that $(D') \in NP$, completing the proof of the above theorem. ■

4.2 Unit processing times and arbitrary release times with the same parity

Theorem 4 implies that the SWA problem is strongly NP-hard in the case of complement of bipartite graphs with $m = n$, $p_i = 1$ and arbitrary release times. We prove that we get a polynomial result if one restricts itself to odd release times, as stated in the next theorem.

Note that the condition $m = n$ means that there are extra machines and therefore one needs to concentrate only on the starting times of the jobs. For any job J_i let t_i represent its starting time. We start by supposing that the release times are arbitrary and odd. Suppose $K_1 = \{J_1, J_2\ , .., J_s\ \}$ and $K_2 = \{J_{s+1}, J_{s+2}\ , .., J_n\}$ and consider the following greedy algorithm.

Algorithm 2
Begin
1: **for** $i := 1$ to s **do**

2: $t_i := r_i$
3: **end for**
4: **for** $i := s+1$ to n **do**
5: **if** (all the jobs of K_1 scheduled at time r_i agree with J_i) **then**
6: $t_i := r_i$
7: **else**
8: $t_i := r_i + 1$
9: **end if**
10: **end for**

end

Theorem 5 *Algorithm 2 solves the SWA problem in the case of arbitrary complement of bipartite agreement graphs, arbitrary odd release times, unit processing times and $m = n$, polynomially in $O(n^2)$.*

Proof. Let τ be the schedule obtained by Algorithm 2 and let $C^*_{\max}$ be the optimal makespan. One can verify that τ is feasible and let us prove that it is optimal. We set $L = \max_{1\leq i\leq n}\{t_i\}$, thus $C_{\max}(\tau) = L+1$. Let J_k be a job satisfying $L = t_k$. Two possibilities may happen:
case 1: t_k is even: then by construction of the algorithm $J_k \in K_2$. On the other hand J_k must have been scheduled at step 8 of the algorithm and we have $t_k = r_k + 1$. Also, there must exist a job $J_i \in K_1$ scheduled at time t_i such that $t_i = r_k$, conflicting with J_k. Since $J_i \in K_1$ then $t_i = r_i$ and hence $r_i = r_k$. Since the jobs J_i and J_k are conflicting jobs with equal release times, then $C^*_{\max} \geq r_k + 2 = t_k + 1 = L + 1 = C_{\max}(\tau)$.
case 2: t_k is odd: so J_k has not been scheduled at step 8 of the algorithm and hence $t_k = r_k$. This implies that $C^*_{\max} \geq r_k + 1 = L + 1 = C_{\max}(\tau)$ and thus the schedule τ is optimal.
Time complexity of Algorithm 2: the for-loop through steps 1-3 can be implemented in $O(n)$ time at worse, the for-loop through steps 4-10 requires at most $O(n^2)$ iterations. Then the time complexity of Algorithm 2 equals $O(n^2)$. ■

By using a similar argument, one can prove the following result:

Corollary 6 *The SWA problem for arbitrary complement of bipartite agreement graphs with arbitrary even release times, unit processing times and $m = n$, can polynomially be solved in $O(n^2)$ time.*

4.3 Identical processing times with two distinct release times

We suppose that the processing times of all jobs are identical and equal to p with two distinct release times 0 and r. Let V_0 and V_r denote the set of jobs with release times 0 and r respectively. We recall that the agreement graph G is the complement of a bipartite graph and so it is not complete. Consider the following greedy algorithm GA.

Algorithm GA
Begin

Case 1: V_r is not a clique.
- Process the jobs of K_1 at time r.
- Process the jobs of K_2 at time $r + p$.

Case 2: Both V_r and V_0 are cliques.
- Process the jobs of V_0 at time 0.
- Process the jobs of V_r at time $\max\{p, r\}$.

Case 3: V_r is a clique and V_0 is not.

3.a: $r \geq 2p$.
- Process the jobs of $V_0 \cap K_1$ at time 0.
- Process the jobs of $V_0 \cap K_2$ at time p.
- Process the jobs of V_r at time r.

3.b: $r < 2p$.

3.b.1: If V_0 can be partitioned into 2 cliques A and B such that $B \cup V_r$ is a clique.
- Process the jobs of A at time 0.
- Process the jobs of $B \cup V_r$ at time $\max\{p, r\}$.

3.b.2: If not, there are two subcases:

If $r < p$ then
- Process the jobs of K_1 at time r.
- Process the jobs of K_2 at time $r + p$.

If $r \geq p$ then
- Process the jobs of $V_0 \cap K_1$ at time 0.
- Process the jobs of $V_0 \cap K_2$ at time p.
- Process the jobs of V_r at time $2p$.

end

In this part, we shall prove that the problem at hand is polynomial, as stated in Theorem 9. For this we start by proving the following two lemmas, needed in the proof of this theorem.

Lemma 7 *The question "can V_0 be partitioned into 2 cliques A and B such that $B \cup V_r$ is a clique of G?" can be checked in polynomial time.*

Proof. Note that, as the graph G is the complement of a bipartite graph, then V_0 can always be partitioned into two cliques A and B.
Let G_{V_0} denote the subgraph of G induced by V_0. The partitioning of V_0 into two cliques A and B, can be formulated by the equations : $x_i + x_j = 1$ for all non-agreeing pairs of jobs $J_i, J_j \in V_0$, $x_i \in \{0, 1\}$. Since A and B play a symmetric role, we may assume that the jobs J_i of A correspond to $x_i = 1$, and those of B correspond to $x_i = 0$. Thus, we have to solve the system of linear equations $MX = \vec{1}$, where $\vec{1} = (1, \ldots, 1)^t$, $X = (x_1, \ldots, x_{|V_0|})^t$ and M is the transpose of the vertex-edge incidence matrix of the complement graph of G_{V_0}. Since the latter is bipartite, then M is totally unimodular. The fact that "$B \cup V_r$ is a clique of G " is equivalent to saying that any job J_i non-adjacent with some job of V_r cannot be in B, and so it is necessarily in A. Therefore, for the condition "$B \cup V_r$ is a

clique of G ", we simply set $x_i = 1$ if the job J_i does not agree with some job of V_r. The system of linear equations $MX = \vec{1}$ for some fixed values of $X \in \{0,1\}^{|V_0|}$ and M totally unimodular is solved in polynomial time by the linear programming. ■

Lemma 8 *If V_r is a clique, V_0 is not a clique of G and V_0 cannot be partitioned into two cliques A and B such that $B \cup V_r$ is a clique of G, we have:*
(1) If $r < p$ then $C^*_{max} \geq r + 2p$.
(2) If $p \leq r < 2p$ then $C^*_{max} \geq 3p$.

Proof. We start by proving (1). To the contrary, suppose $C^*_{max} < r + 2p$, then there would exist a feasible schedule σ with $C_{max}(\sigma) < r + 2p$. Let $A = \{J_i \in V : t_i < r\}$, where t_i is the starting time of the job J_i with respect to the schedule σ. Let us show that $A \neq \phi$. Suppose $A = \phi$, then $t_i \geq r$ for all jobs J_i of V, in particular for those of V_0. As V_0 is not a clique, then there exist two non-agreeing jobs in V_0, thus $C_{max}(\sigma) \geq r + 2p$, contradiction and so $A \neq \phi$. Moreover, we claim that $A \neq V_0$ since otherwise for all jobs J_i of $V_0 : t_i < r_i$. As processing times of the jobs is p and $r < p$ then the jobs of V_0 will be pairwise concurrently scheduled in σ. This would imply that V_0 is a clique, contradiction with the hypothesis and this terminates our claim. In addition, since all the jobs of V_r have to start not before the time r, then A is a non-empty proper subset of V_0. Let $X = V \setminus A$. The set X contains the set V_r since any job J_i of V_r has a starting time $t_i \geq r$ and so it is in X. X is a clique of G, since otherwise there would be two non-agreeing jobs in X leading to $C_{max}(\sigma) \geq r + 2p$, contradiction. As $A \neq V_0$, then X can be written $X = B \cup V_r$ where $B = V_0 \setminus A$, and subsequently the sets A and B would form a partition of V_0 such that $B \cup V_r$ is a clique of G, contradicting the hypothesis. The statement (1) is proved. As far as statement (2) is concerned, one can adopt the same argument used in proving statement (1), by setting $A = \{J_i \in V : t_i < p\}$ and proceed similarly until arriving at the same contradiction with the hypothesis. This ends the proof of Lemma 8. ■

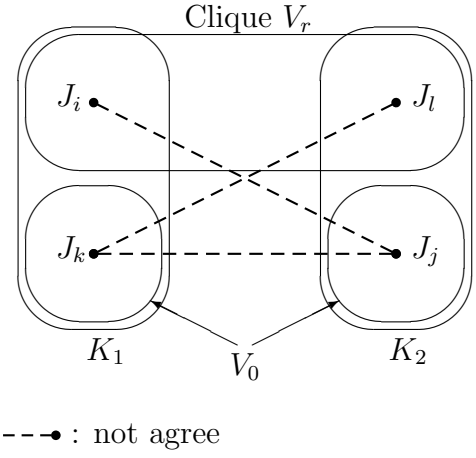

Figure 7: Illustration of Lemma 2: a situation where V_r is a clique, V_0 is not a clique and V_0 cannot be partitioned into 2 cliques A and B such that $B \cup V_r$ is a clique.

Figure 7 illustrates a possibility where V_0 is not a clique and V_r is a clique. Let us explain why V_0 cannot be partitioned into two cliques A and B such that $B \cup V_r$ is a clique. Suppose it is not the case, that is V_0 can be partitioned into two cliques A and B such that $B \cup V_r$ is a clique. Consider the pair of jobs (J_k, J_j). Since these two jobs are not adjacent then either $J_k \in B$ or $J_j \in B$ but not both. Now, in view of this figure, neither J_k nor J_j is adjacent to V_r and hence $V_r \cup B$ cannot be a clique, contradiction.

Theorem 9 *The algorithm GA cited above solves the SWA problem in the case of complement of bipartite agreement graphs, two distinct release times (*0 *and* r*), identical processing times (*$p_i = p$*) and* $m = n$*, in polynomial time.*

Proof. We denote by C^*_{max} the makespan of an optimal solution of the problem at hand. Since we have supposed that agreement graph $G = (K_1; K_2, E)$ is not complete, so there are two non-agreeing jobs, one from K_1 and the other from K_2 which implies that $C^*_{max} \geq 2p$. On the other hand $C^*_{max} \geq r + p$ ($V_r \neq \phi$ since by hypothesis 0 and r are distinct), then we obtain the inequality $C^*_{max} \geq \max\{2p, r + p\}$, which can be written

$$C^*_{max} \geq \max\{r, p\} + p \tag{1}$$

Case 1: V_r is not a clique. In this case, there are two non-agreeing jobs in V_r and thus $C^*_{max} \geq r + 2p$. therefore Algorithm GA returns an optimal solution.

Case 2: Both V_r and V_0 are cliques. The solution obtained by Algorithm GA has a makespan equals $\max\{r, p\} + p$, by inequality 1 it is optimal.

Case 3: V_r is a clique and V_0 is not. We distinguish the two sub-cases (3.a) and (3.b).

Case 3.a: $r \geq 2p$. Since $\max\{r, p\} = r$, then from inequality 1 the solution obtained by Algorithm GA is optimal with $C_{max} = r + p$

Case 3.b: $r < 2p$. Here as well, we discuss the two possibles cases (3.b.1) and (3.b.2).

Case 3.b.1: V_0 can be partitioned into two cliques A and B such that $B \cup V_r$ is a clique of G. One can verify that the schedule obtained by Algorithm GA is feasible with $C_{max} = \max\{r, p\} + p$ and hence it is optimal by inequality 1.

Case 3.b.2: V_0 cannot be partitioned into two cliques A and B such that $B \cup V_r$ is a clique in G. If $r < p$ then $C_{max} = r + 2p$ and hence, by Lemma 8, the obtained solution is optimal. A similar argument can be done when $r \geq p$ with $C_{max} = 3p$.

For all the cases cited above, the algorithm GA gives the optimal solution, and this terminates the proof of theorem 9. ■

5 Approximation results

The first result is a direct consequence of Theorem 3.

Corollary 10 *It is NP-hard to approximate with factor better than* $4/3-\epsilon$ *for any* $\epsilon > 0$*, the SWA problem with arbitrary complement of bipartite agreement graphs, unit processing times and* $r_i \in \{0, 1, 2\}$ *even if* $m = n$*.*

Proof. In the reduction used in the proof of Theorem 4, it has been proved that there is 3-coloring (C_1, C_2, C_3) of G such that $v_i \in C_i$, for all $i (i = 1, 2, 3)$ if and only if there is a feasible schedule σ for D' with $C_{\max}(\sigma) \leq 3$.
This reduction is in fact a gap reduction since one can show that if the problem 1-PrExt is ρ-approximable with $\rho < 4/3$ then it will be polynomial, and this implies $P = NP$. ■

The next result is a positive one, concerning complement of bipartite agreement graphs.

Theorem 11 *The SWA problem with arbitrary complement of bipartite agreement graphs,* $m = n$*, identical processing times* $(p_i = p)$ *and arbitrary release times has a polynomial time approximation algorithm A with performance guarantee* $|A(I) - OPT(I)| \leq p$*.*

Proof. We have $OPT(I) \geq \max\limits_{1 \leq i \leq n} \{r_i\} + p$ for all instances I. If we consider the algorithm A which schedules the jobs of one clique at time $\max\{r_i\}$ and the jobs of the other clique at time $\max\{r_i\} + p$, the makespan obtained is $A(I) \leq \max\{r_i\} + 2p$ for any instance I. Then $A(I) \leq OPT(I) + p$, which implies that $|A(I) - OPT(I)| \leq p$. ■

Note that this algorithm gives a 2-approximation for the problem at hand. It is also worth noting that in the case of unit processing times, there is an approximation which guarantees $|A(I) - OPT(I)| \leq 1$.

In reference [8], the authors have proved that for $m = 2$ there is no polynomial time approximation algorithm A for SWA with performance guarantee $|A(I) - OPT(I)| \leq K$ for any fixed constant K. Next we generalize this result for any NP-hard subproblem of SWA.

Proposition 12 *If* $P \neq NP$ *then there is no polynomial time approximation algorithms A for any NP-hard subproblem of SWA which can guarantee* $|A(I) - OPT(I)| \leq K$ *for any fixed constant* K*.*

Proof. We proceed by contradiction using a scaling argument. Assume such an algorithm A exists. Let I be an instance of SWA and I' is similar to I except that the processing times are scaled up by a factor of $K+1$. One can verify that to each feasible schedule σ for

the instance I there corresponds a feasible schedule σ' for the instance I' and vice-versa such that $C_{\max}(\sigma') = (K+1)C_{\max}(\sigma)$. We now run algorithm A on both instances I and I' and let σ_A the schedule obtained by A. Clearly $|A(I') - OPT(I')| \leq K$ which implies that σ_A) is an optimal schedule for SWA, contradicting $P \neq NP$. ■

6 Conclusion

In this paper, the problem of scheduling with agreements (SWA) is addressed. We have generalized two old NP-hardness results for SWA in the case of two machines, processing times at most 3 that hold for bipartite agreement graphs to any class of agreement graphs with a specific structure. A polynomial result for the case of split agreement graphs has been presented, followed by some complexity results in the case of complement of bipartite agreement graphs. Finally we have established some approximation results for SWA.

References

[1] B.S. Baker, E.G. Coffman, Mutual Exclusion Scheduling, Theoretical Computer Science. 162 (1996) 225–243.

[2] M. Bendraouche, M. Boudhar, Scheduling jobs on identical machines with agreement graph, Computers and Operations Research. 39 (2012) 382–390.

[3] M. Bendraouche, M. Boudhar, A. Oulamara, Scheduling: Agreement graph vs resource constraints. European Journal of Operational Research. 240 (2015) 355–360.

[4] H. L. Bodlaender, K. Jansen, Restrictions of graph partition problems part I, Theoretical Computer Science. 148 (1995) 93–109.

[5] H. L. Bodlaender, K. Jansen, G.J. Woeginger, Scheduling with incompatible jobs, Discrete Applied Mathematics. 55 (1995) 219–232.

[6] J. Cheriyan, S.N. Maheshwari, Analysis of preflow push algorithms for maximum network flow, SIAM Journal on Computing. 18 (1989) 1057–1086.

[7] M. Chrobak, J. Csirik,C. Imreh, J. Noga, J. Sgall, G.J. Woeginger, The buffer minimization problem for multiprocessor scheduling with conflicts, In Proc. 28th International Colloquium on Automata, Languages, and Programming. (2001) 862–874.

[8] G. Even, M.M. Halldorson, L. Kaplan, D. Ron, Scheduling with conflicts: online and offline algorithms, Journal of scheduling. 12 (2009) 199–224.

[9] F. Gardi, Mutual exclusion scheduling with interval graphs or related classes Part I, Discrete Appl Math. 157 (2009) 19–35.

[10] M.R. Garey, D.S. Johnson, Computers and Intractability, New York: Freeman, 1979.

[11] P. Hansen, A. Hertz, J. Kuplinski, Bounded vertex colorings of graphs, Discrete Math. 111 (1993) 305–312.

[12] S. Irani, V. Leung, Scheduling with conflicts, and applications to traffic signal control, Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms. (1996) 85–94.

[13] K. Jansen, The mutual exclusion scheduling problem for permutation and comparability graphs, Inform and Comput. 180(2) (2003), 71–81.

[14] J. Jarvis, B. Zhou, Bounded vertex coloring of trees, Discrete Math. 232 (2001) 145–151.