Scheduling conflicting jobs: application and new results

Amin Mallek
RECITS lab, Faculty of Mathematics
USTHB University
Bab Ezzouar, Algiers, Algeria
Aminn.mallek@gmail.com

Mohamed Bendraouche
RECITS lab, Faculty of Mathematics
USTHB University
Bab Ezzouar, Algiers, Algeria
mbendraouche@yahoo.fr

Mourad Boudhar
RECITS lab, Faculty of Mathematics
USTHB University
Bab Ezzouar, Algiers, Algeria
mboudhar@yahoo.fr

***Abstract*—In this paper, we consider an assignment problem in a health care facility where doctors with different working volumes have to be assigned to continuous shifts. A doctor is not allowed to work two consecutive shifts, and sometimes both the first and the last shifts in the month cannot be served by the same doctor. We show that this problem is equivalent to minimising the makespan C_{max} of scheduling n unit-time conflicting jobs on m parallel uniform machines with speeds $s_1 \geq s_2 \geq \ldots \geq s_m$. The conflict is modelled by a graph G, in which two adjacent jobs cannot be processed on the same machine. Herein, for the sake of the assignment problem, G is restricted to a path P_n and a cycle C_n. We present polynomial time algorithms to solve both problems to optimality.**

***Index Terms*—Scheduling, Conflict graph, Incompatible jobs, Uniform machines, Continuous shifts, Assignment problem.**

I. Introduction

We address, in this paper, the following scheduling problem: given a set J of n identical jobs, $J = \{J_1, J_2, \ldots, J_n\}$, with one unit processing times, $p_i = 1$ $(i = 1, \ldots, n)$, to be scheduled on a set M of m parallel uniform machines, $M = \{M_1, M_2, ..., M_m\}$. The machines run at different speeds s_j $(j = 1, \ldots, m)$, so as a machine M_j completes one job execution in $1/s_j$ units of time. It is supposed here that $s_1 \geq s_2 \geq ... \geq s_m$. We assume that the jobs are subjected to conflicting constraints modelled by an undirected graph G, called the conflict graph. Each job is represented by a vertex in G. Two jobs are incompatible if and only if there exists an edge between them in G, which also means that these jobs cannot be processed on the same machine.

A feasible schedule is an assignment of the jobs to the machines where no two jobs on the same machine overlap or are adjacent in G. In a given schedule, the completion time of a job, denoted c_i, is defined as the time by which the machine completes its execution. The largest one among them defines the makespan of the schedule C_{max}. The objective is to find a feasible schedule with the minimum makespan. Using three field notation introduced in [10], this problem is written in symbols as follows : $Q|p_i = 1, G = \text{conflict graph}|C_{max}$.

The scheduling problem, described above (referred to as SCU), when restricted to path graphs P_n or cycles C_n is related to one of the continuous shifts assignment problems, we refer to as CSA. We often encounter this kind of problems in hospitals, surveillance services, maintenance services, etc. The CSA problem considered here is described as follows : A health care facility is contracted with many doctors. The type of contract is not the same for all of them, some work full-time, others part-time, the rest is working according to a specific volume mentioned in the contract. The facility working 24h without cease. The day is divided into k shifts i.e. in one month there is $30k$ $(28k, 29k$ or $31k)$ shifts to be assigned to the doctors with respect to their working volume. Due to stress and fatigue caused by this kind of jobs, two consecutive shifts cannot be assigned to the same doctor. Sometimes we forbid to send the same doctor to work the first and the last shifts in the month owing to additional paper work in those shifts. The objective is to find a feasible assignment of the doctors to the shifts with respect to the afore-cited constraints.

The rest of this paper is structured as follows. In section 2 we report some related work. A linear formulation and a modelling of the problem is given in section 3. Section 4 presents polynomial time algorithms to solve the problem. We conclude the paper in section 5.

II. Literature review

The problem of scheduling incompatible jobs, also known as scheduling with conflicts in literature, has been investigated in many papers. First of all, as regards the problem with arbitrary job processing times and identical parallel machines in presence of an arbitrary conflict graph, Bodlaender and Jansen in [4] and Kowalczyk and Leus in [9] mentioned that this problem is NP-complete even if G is empty, as it contains both 3-partition and graph colouring problems as special cases, which makes it even hard to approximate. Bodlaender and Jansen presented in [4] several heuristic approaches according to the class of G, whereas, Kowalczyk and Leus introduced in [9] an exact algorithm to solve the problem in general. Furthermore, Bodlaender and Jansen. proved in [2] that this problem remains NP-complete even if all the jobs have one unit processing times by making reduction from the problem of partitioning a graph into bounded independent sets introduced by themselves in [3]. Concerning parallel uniform machines, Dessouky et al. proved in [5] that the problem of scheduling identical jobs on parallel uniform machines is solvable in polynomial time when the conflict graph is empty. Based on the work of Bodlaender and Jansen in [2] [3], Furmanczyk and Kubale proved in [6] that the scheduling problem $Q4|p_i = 1, G = \text{Bipartite}|C_{max}$ is NP-hard. The same authors, relying on their work in [8], proved in [7] the NP-hardness of the problem $Q3|p_i = 1, G = \text{Cubic}|C_{max}$ when G is 3-chromatic. They also gave an $O(n^2)$ time algorithm to solve the same problem when G is 2-chromatic. The problems of personnel assignment to work shifts have been widely investigated due to their large practical importance. Here, we are only interested in scheduling problems and related works.

III. Formulation and modelling

A. Linear formulation of CSA

We provide here an integer linear programming formulation (P) for the problem.

(P) is formulated as follows. For every shift i and doctor j a binary variable x_{ij} equals 1 if i is assigned to j and 0 otherwise. For each doctor j, we denote his working volume by v_j. The CSA model can be structured as follows :

$$(P)\left|\begin{array}{lll} \textit{Find solution to} & & (1) \\ \sum_{j=1}^{m} x_{ij} = 1 & \forall i = \overline{1,n} & (2) \\ x_{ij} + x_{(i+1)j} \leq 1 & \forall i = \overline{1,n-1};\ \forall j = \overline{1,m} & (3) \\ \sum_{i=1}^{n} x_{ij} = v_j & \forall j = \overline{1,m} & (4) \\ x_{ij} \in \{0,1\} & \forall i = \overline{1,n};\ \forall j = \overline{1,m} & (5) \end{array}\right.$$

The objective is to obtain a feasible assignment. Constraints (2) assure that each shift is assigned to exactly one doctor. The conflict is modelled by inequalities (3) as they force consecutive shifts to be taken by different doctors. Inequalities (4) ensure that for each doctor the amount of assigned shifts is equal to his working volume. The condition on variables x_{ij} is given in (5).

If the first and the last shifts cannot be served by the same doctor we add, to the model, the following constraints :

$$x_{1j} + x_{nj} \leq 1, \forall j = \overline{1,m}$$

B. Modelling

In this section we show how CSA can be modelled as SCU restricted to paths and cycles. Solving these two cases of SCU will allow us to solve CSA in the same way.

Each shift is considered as a job. A doctor is represented by a machine, where the work volume is designated as its speed. For a path graph P_n, two consecutive jobs cannot be processed by the same machine which means, in CSA, that two consecutive shifts cannot be assigned to the same doctor. For a cycle C_n, we can solve the extension to the main problem where the first shift and the last one cannot be assigned to the same doctor. Minimising the makespan, C_{max}, of SCU is equivalent to finding a feasible assignment to the continuous shifts assignment problem. Note that the number of shifts to be worked by each doctor is determined according to his work volume and set as a machine's speed. Consequently, 1 is a lower bound to C_{max}.

IV. Resolution

We investigate here the complexity of our scheduling problem when the conflict graph is restricted to paths and cycles. In this section, we prove that these two problems are efficiently solvable in polynomial time. In what follows, for convenience we will refer to the jobs numbered odd (respectively even) as odd jobs (respectively even jobs).

A. Basic result

We first state one of the basic results as regards scheduling problems on parallel uniform machines, that is when the conflict graph is empty (the vertices form an independent set). Two different methods have been presented in [5] to solve this problem in polynomial time, namely in $O(n + m. \log m)$ time or in $O(n. \log m)$ time. The $O(n + m. \log m)$ time method is summarised in Algorithm 1.

First, consider the following linear formulation :

$$(P1)\left|\begin{array}{lll} \textit{Minimise} & C_{max} & (1) \\ \textit{Subject to} & & \\ & \sum_{j=1}^{m} x_j = n & (2) \\ & x_j / s_j \leq C_{max} \quad \forall j = \overline{1,m} & (3) \\ & C_{max} \geq 0 & (4) \\ & x_j \in \{0,1\} \quad \forall j = \overline{1,m} & (5) \end{array}\right.$$

The decision variable x_j denotes the number of jobs to be assigned to machine M_j $(j = 1, \ldots, m)$.

A priority queue is a data structure for a set of n ordered elements, in which inserting or deleting an element takes $O(\log n)$ time [1]. Also, note that a queue of n elements can be ordered in $O(n. \log n)$ time.

Algorithm 1: Scheduling independent jobs

```
Data: $n$ unit-time jobs, $m$ uniform machines.
Result: Optimal schedule.
1 Solve the linear programming relaxation of (P1);
2 Assign $\lfloor x_j \rfloor$ jobs to $M_j$ $(j = 1, \dots, m)$;
3 Let $r = n - \sum_{j=1}^{m} \lfloor x_j \rfloor$ be the number of unscheduled
    jobs;
4 If $r > 0$, initialise a priority queue of the $m$ current
    machines completion times, send each of the $r$ jobs
    to the machine with the earliest completion time in
    the queue; then update the queue according to the
    new machine's completion time.
```

B. Path graphs

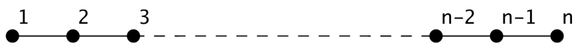
In what follows, we present a polynomial time algorithm to solve SCU restricted to paths (Algorithm 2). Then, we show why the produced schedule is optimal.

Algorithm 2: $Q|p_i = 1, G = \mathit{Path}|C_{max}$

```
Data: $n$ conflicting jobs modelled by a path, $m$
      uniform machines ($m \geq 2$).
Result: Optimal schedule.
1  Label the jobs (vertices) in the path increasingly from
     1 to $n$;
2  Compute through Algorithm 1 (regardless of conflict)
     the number of jobs $n_j$ to be sent to machine $M_j$;
3  if $m = 2$ then
4      Assign to $M_1$ (respectively $M_2$) all the odd jobs
         (respectively all the even jobs);
5  else
6      if $n_1 > \lceil \frac{n}{2} \rceil$ then
7          Assign to $M_1$ all the odd jobs;
8          Apply Algorithm 1 to schedule the even jobs
             on $M - M_1$;
9      else
10         Apply Algorithm 1 to determine $n_j$ the
             number of jobs to be assigned to $M_j$;
11         Assign $n_j$ jobs to $M_j$, starting with the odd
             jobs first then the even ones.
```

Theorem 1. *The scheduling problem $Q|p_i = 1, G = \mathit{Path}|C_{max}$ can be solved in $O(n + m. \log m)$ time using Algorithm 2.*

Proof. We first label the jobs in the path increasingly from 1 to n as follows:



It is so obvious that the number of odd jobs is greater than or equal to the number of even ones. It is also clear that no machine can receive more than $\lceil n/2 \rceil$ jobs, which is the size of the maximum independent set in P_n (also the number of odd jobs).

In case $m = 2$, the optimal solution can be found by assigning the jobs numbered odd to the fastest machine and the even ones to the other machine.

For $m \geq 3$: using *Algorithm 1*, we determine the number of jobs n_j, to be assigned to M_j $(j = 1, \dots, m)$, regardless of the conflict constraints. Due to the machines speed we get: $n_1 \geq \dots \geq n_m$.

Now, we have to prove that there is a way to schedule the jobs with respect to n_j $(j = 1, \dots, m)$ without having two conflicting jobs on the same machine, otherwise, any change we make keeps the schedule optimal. To do so we distinguish two cases :

1) $n_1 > \lceil n/2 \rceil$: we stated above that no machine can receive more than $\lceil n/2 \rceil$ jobs. Hence, in this case we assign to M_1 all the odd jobs, then we schedule the even ones on the remaining machines, $M - M_1$, using *Algorithm 1*.
2) $n_1 \leq \lceil n/2 \rceil$: Starting from the fastest machine M_1 to the slowest one M_m, we schedule increasingly the odd jobs first, then the even ones right after. Let M_k, $1 \leq k \leq m$, be the machine that contains both odd and even jobs, if the case exists. These jobs are compatible if and only if for any two jobs s and t in M_k we have $|s - t| \geq 2$, which also can be viewed as the distance between them in the path.
Let α denote the number of odd jobs in M_k, let r be the first odd job scheduled on M_k, $r \geq 2n_{k-1} + 1$. Figure 1 depicts how the jobs are ordered in machines.

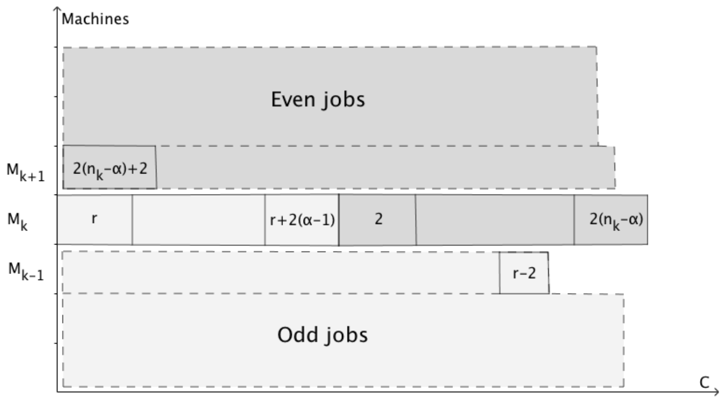


Fig. 1. The order of the jobs in the machines.

To make sure that all the jobs in M_k are compatible, we have to check that the distance between the closest odd job and even job (scheduled on M_k) is greater than or equals 2.
$2(n_k - \alpha)$ and r are the closest ones (see Figure 2), the order comes from the fact that $r \geq 2n_k + 1 \geq 2(n_k - \alpha)$ (note that $n_{k-1} \geq n_k$).
In the following, the result of subtracting $2(n_k - \alpha)$ from

r is a distance that equals at least 3.

$$|r - 2(n_k - \alpha)| = |r - 2n_k + 2\alpha| \geq 2\alpha + 1 \geq 3$$

($r - 2n_k \geq 1$ and $\alpha \geq 1$).

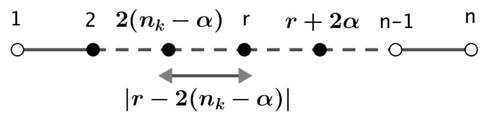


Fig. 2. The distance between the closest odd job and even job scheduled on M_k.

□

C. Cycle graphs

Based on the previous result, we prove that for cycle graphs Algorithm 3 yields an optimal schedule to the problem in polynomial time.

```
Algorithm 3: $Q|p_i = 1, G = Cycle|C_{max}$
  Data: $n$ conflicting jobs modelled by a cycle, $m$
        uniform machines ($m \geq 2$).
  Result: Optimal schedule.
1 Label the jobs (vertices) in the cycle increasingly from
    1 to $n$;
2 if $n$ is even then
3     Apply Algorithm 2;
4 else if $m \geq 3$ then
5     if $n_1 > \lceil \frac{n}{2} \rceil$ then
6         Assign to $M_1$ all the odd jobs except the job
            numbered $n$;
7         Apply Algorithm 1 to schedule the even jobs
            and the job numbered $n$ (as one independent
            set) on $M - M_1$, such that job $n-1$
            (respectively job $n$) is assigned to $M_2$
            (respectively $M_3$);
8     else
9         Apply Algorithm 2.
```

Corollary 1. *The scheduling problem $Q|p_i = 1, G = Cycle|C_{max}$ can be solved in $O(n + m.\log m)$ time using Algorithm 3.*

Proof. Using the same fashion described above with the paths, we can produce an optimal schedule considering the following cases :

1) n even: it is the same case of the path with an extra condition which is jobs 1 and n cannot be processed by the same machine. Since 1 is odd and n is even, the procedure used with the paths assures that those jobs can never be together on the same machine.
2) n odd: in this case the graph is 3-chromatic, hence, we need at least 3 machines ($m \geq 3$). We almost do the same thing we did with the paths under the restriction that 1 and n cannot be processed by the same machine. An optimal schedule can be found as follows : if $n_1 > \lfloor n/2 \rfloor$ then M_1 receives all the odd jobs except n which will be scheduled with the even jobs on the remaining machines ($M - M_1$) as one independent set using *Algorithm 1*. We only need to put the jobs n and $n-1$ on different machines.
Otherwise, $n_1 \leq \lfloor n/2 \rfloor$, the method used with the paths assures that the job 1 is scheduled on M_1 and the job n on some other machine since $n_1 \leq \lfloor n/2 \rfloor$. Thus, we wind up with the same restrictions as in a path of n jobs. Hence, we can produce an optimal solution by applying the same method used with the paths.

□

V. Conclusion

The work done in this paper was motivated by a continuous shifts assignment problem encountered in a health care facility and in many similar contexts. We showed that this problem is equivalent to minimising the makespan of scheduling identical conflicting jobs on uniform machines when the conflict is modelled by a path P_n or a cycle C_n. We proved that the solution to the problem produced by *Algorithm 2* and *Algorithm 3* is optimal and is obtained in polynomial time.

References

[1] Aho A.V., Hopcroft J.E., and Ullman J.D. (1982). Data structures and algorithms, Addison-Wesley, Reading, Massachusetts.
[2] Bodlaender H.L., Jansen K. (1993). On the complexity of scheduling incompatible jobs with unit-times, LNCS 711: 291300.
[3] Bodlaender H.L., Jansen K. (1995). Restrictions of graph partition problems. Part I. Theoretical Computer Science 148: 93109.
[4] Bodlaender H.L., Jansen K. and Woeginger G.J. (1994). Scheduling with incompatible jobs. Discrete Applied Mathematics 55: 219-232.
[5] Dessouky M.I., Lageweg B.J., Lenstra J.K. and van de Velde S.L. (1990). Scheduling identical jobs on parallel uniform machines. Statistica Neerlandica 44(3): 115123.
[6] Furmanczyk H., Kubale M. (2017). Scheduling of unit-length jobs with bipartite incompatibility graphs on four uniform machines. Bulletin of the Polish Academy of Sciences Technical Sciences 65(1): 2934.
[7] Furmanczyk H., Kubale M. (2018). Scheduling of unit-length jobs with cubic incompatibility graphs on three uniform machines. Discrete Applied Mathematics 234: 210217.
[8] Furmannczyk H., Kubale M. (2015). Equitable and semi-equitable colouring of cubic graphs and its application in scheduling, Arch. Control Sci. 25: 109-116.
[9] Kowalczyk D., Leus R. (2017). An exact algorithm for parallel machine scheduling with conflicts. Journal of Scheduling 20(4): 355-372.
[10] Lawler, E.L.,Lenstra, J.K. and Rinnooy Kan, A. H. G. (1982). Recent developments in deterministic sequencing and scheduling : A survey. In M. A. H. Dempster, J. K. Lenstra, and A. H. G. Rinnooy Kan (Eds.), Deterministic and stochastic scheduling (pp. 35-73), NATO advanced study Institutes series New York: Springer.