Abstract:
This paper investigates a variant of parallel-machine scheduling problems with conflict constraints. The conflict constraints specify pairs of jobs that are mutually disjoint due to resource availability. Jobs conflicting to each other cannot be processed simultaneously. This paper discusses the specific two-machine problem where each machine has a designated set of jobs to process. The objective is to find a feasible schedule to minimize the makespan. NP-hardness proof of the case with a fixed processing sequence on one machine is presented. We design a polynomial-time dynamic programming algorithm for the case where the processing sequences on both machines are known in a priori.

Keywords: Project scheduling, conflict constraints, dynamic programming.

1. Introduction
Scheduling refers to decisions about allocation of resources to economic activities along the time horizon with respect to prescribed managerial performance measures (Pinedo 2012). This paper investigates a scheduling problem on parallel dedicated machines to minimize the makespan. The schedule should abide conflict constraints among the jobs. The problem arises from the applications context where specific resource availability is limited. A job demands a certain type of resource, e.g., a skilled technician for setting up the machine, or, a senior surgical physician for conducting critical operations. Conflicts exists some jobs arise when their total demand of certain resources exceeds the supply. Overlapped processing of conflicting jobs is not allowed at any time point (Gendreau et al. 2004, Epstein et al. 2008, Pereira 2016).
The problem of interest is formally defined as follows. There are two machines \( M_1 \) and \( M_2 \) dedicated for processing two disjoint sets of jobs \( N_1 = \{ J_{1,1}, J_{1,2}, \ldots, J_{1,n_1} \} \) and \( N_2 = \{ J_{2,1}, J_{2,2}, \ldots, J_{2,n_2} \} \), in which \( n_1 \) and \( n_2 \) are the numbers of jobs in the two sets, respectively. At any time, any machine processes at most one job. Let \( p_{i,j} \) denote the processing time of job \( J_{i,j} \), \( i = 1 \) or \( 2 \), \( j = 1, 2, \ldots, n_i \). Conflict constraints are prescribed by an undirected bipartite graph \( G = (V_1 U V_2, E) \), in which \( V_1 = N_1 \) and \( V_2 = N_2 \) are the two job sets, and edge \( (J_{1,j_1}, J_{1,j_2}) \) in \( E \) if job \( J_{1,j_1} \) in \( N_1 \) and job \( J_{2,j_2} \) in \( N_2 \) have conflicts with each other. Processing of any two conflicting jobs cannot overlap at any time.

We denote by \( C_{i,j} \) the completion time of job \( J_{i,j} \) under a particular feasible schedule. The makespan \( C_{\text{max}} \) of a schedule is defined as the time at which all jobs are completed, i.e., \( C_{\text{max}} = \max \{ i=1,2; j=1,\ldots,n_i \} \{ C_{i,j} \} \). This paper aims to compose a feasible schedule that attains the minimum makespan. We use \( \text{PD}_2|\text{conflict}|C_{\text{max}} \) to denote the studied problem. Since the classical two-machine problem \( P2||C_{\text{max}} \) is known to be NP-hard (Garey and Johnson 1979), we consider the variant in which assignment of jobs to the machines is already done. The notation will be augmented with parameters “seq1” and “seq2” as \( \text{PD}_2|\text{conflict,seq1}|C_{\text{max}} \) and \( \text{PD}_2|\text{conflict,seq2}|C_{\text{max}} \) to indicate the condition that a processing sequence is assumed for one of the machines.

The paper is organized in five sections. Section 2 is dedicated to literature review and justification for the assumption of fixed sequences. In Section 3, we classify the computational complexities of different cases. Section 4 is for the development of polynomial time dynamic programming algorithm for the scenario with two fixed job sequences. We then summarize the paper and suggest future researches in Section 5.

2. Literature Review

To discuss the computational complexity of makespan minimization in parallel-machine scheduling with conflict graphs, we start with the cases that all jobs have the same unit execution time (UET), equivalent to finding a graph partition with a minimum number of cliques, each with size bounded from above by the number of machines (Baker and Coffman 1996). Scheduling UET jobs with conflict graphs on two machines is easy because a maximum matching in the conflict graph corresponds to an optimal solution. Strong NP-hardness of the scheduling problem with conflict constraints is established for \( m \geq 3 \), even if only UET jobs are considered (Baker and Coffman 1996). This paper addresses the assumption of dedicated
machines, i.e., jobs are already assigned to machines. Agnetis et al. (2012) investigated several regular criteria on parallel dedicated machines subject to chain precedence constraints. They proposed polynomial time algorithms, fully polynomial time approximation schemes and NP-hardness proofs for different problem settings. Glass et al. (2000) considered scheduling to minimize the makespan for two parallel dedicated machines in which the setup phase of each operation needs to be attended by a common server, which could be a hoister or a skilled technician. They proved that the problem remains NP-hard even if all setup times are uniform or if all processing times are uniform. Huang et al. (2010) studied the scheduling problem to minimize the makespan with sequence-dependent setups and a single server. They designed efficient algorithms for a special case, where all jobs have the uniform processing time and all the setup times are greater than or equal to $p$. For the general case, an MILP program and a hybrid genetic algorithm were proposed. Scheduling on parallel dedicated machines under a single non-shared resource was proven to be unary NP-hard when the number of machines is part of the input (Kellerer and Strusevich 2003a). Later, they further showed that the scheduling problem with two parallel dedicated machines and multiple resource constraints is NP-hard under the situation that there are two types of resources, two units of each type of resources, and each job will consume two units of resources (Kellerer and Strusevich 2003b). The problem becomes polynomial solvable if each job will consume at most one unit of each type of resource and there are only one unit of each type of resources, even if the number of resource types is arbitrary. Goemans (1995) proposed a 7/6-approximation algorithm for scheduling jobs on their dedicated machines, where the term “parallel dedicated machines” indicates that every job has a designated subset of machines are required at the same time for processing this job.

Motivated by the NP-hardness of parallel-machine scheduling with conflict graphs, we will consider the setting where the processing sequences of jobs on the machines are known in advance. For computationally challenging problems, different solution approaches, including exact methods, like branch-and-bound algorithms, and approximation heuristics or meta-heuristics, can be adopted, depending on whether optimal solutions or approximate ones are to be found. Permutations of jobs on the machines are commonly deployed for representing solutions. In many scheduling problems, optimal schedules consisting of exact start times or finish times of operations can be easily derived if processing sequences are known. Shafransky and Strusevich (1998), Kononov and Lin (2006), Hwang and Lin (2011), and Lin et al. (2016) investigated several scheduling problems that remain NP-hard even if one or more job
sequences are presumed. More justifications can be found in Shafransky and Strusevich (1998) and Hwang and Lin (2012).

3. Complexity Classification

This section discusses the complexity status of two special cases. The first reduction is from 3-PARTITION, which is known to be strongly NP-hard.

3-PARTITION: Given a constant $B$ and a set of $3u$ elements $A=\{1, 2, \ldots, 3u\}$ in which each $j$ has a positive size $a_j$ satisfying $\sum_{j \in A} a_j = uB$ and $B/4 < a_j < B/2$ for all $j$ in $A$, can set $A$ be partitioned into $u$ subsets $A_1, A_2, \ldots, A_u$ such that $\sum_{j \in A_1} a_j = \cdots = \sum_{j \in A_u} a_j = B$?

**Theorem 1:** $PD2|\text{conflict}, \text{seq1}|C_{\text{max}}$ is strongly NP-hard.

**Proof:** Given an instance of 3-Partition, we construct a scheduling instance of $5u$ jobs with a conflict graph as follows.

- $2u$ jobs in $N_1$: $p_{1,j}=B, 1 \leq j \leq 2u$;
- $3u$ jobs in $N_2$: $p_{2,j}=a_j, 1 \leq j \leq 3u$;

The processing sequence $(J_{1,1}, J_{1,2}, \ldots, J_{1,2u})$ is given for machine $M_1$. Conflict relation defines conflicting pairs $(J_{1,j}, J_{2,j'})$ for $1 \leq j \leq 3u, j'=2, 4, \ldots, 2u$. It can be proved that there exists a solution to 3-Partition if and only if there is a feasible schedule with a makespan not greater than $2uB$. ■

Another special case is for the setting where the jobs are partitioned into two subsets, one for each machine without a predetermined processing sequence for any machine. Kellerer and Strusevich (2003) gave an NP-hardness proof of a resource-constrained scheduling problem for two parallel dedicated machines. Any two jobs from different machines cannot overlap if their total resource requirement exceeds the resource availability. We introduce a conflict edge between such two jobs. Then, the $PD2|\text{conflict}|C_{\text{max}}$ problem with two designated sets of jobs is hard as well.

**Theorem 2:** $PD2|\text{conflict}|C_{\text{max}}$ is ordinary NP-hard.

Since the problem without presumed fixed sequences on the machines and the problem with a fixed processing sequence on one machine remain difficult to solve, further restrictions of two
fixed job sequences may render the problem polynomial solvable. We discuss this case in the next section.

4. Dynamic Programming Algorithms

In this section, we discuss the case where the processing sequences on both machines are given. Therefore, the decision is about which machine will proceed first when a processing conflict arises. We will propose a dynamic programming algorithm to solve the problem.

We develop blocks as the base for recursions in the dynamic programming algorithm. A schedule can be interpreted as a concatenation of jobs and blocks that are separated by idle slots on either of the machines. Blocks describe non-interrupted processing scenarios of jobs. For two subsequences \( J_{1,j_1:j_1'} = (J_{1,j_1}, J_{1,j_1+1}, \ldots, J_{1,j_1'}) \) and \( J_{2,j_2:j_2'} = (J_{2,j_2}, J_{2,j_2+1}, \ldots, J_{2,j_2'}) \), we define a block \( (j_1, j_1', j_2, j_2') \) that satisfies the following two conditions:

1. No idle time is inserted between any two jobs on the same machine.
2. \( J_{1,j_1} \) is the only job satisfying \( C_{1,j_1'} \geq C_{2,j_2'} \), or \( J_{2,j_2} \) is the only job satisfying \( C_{2,j_2'} \geq C_{1,j_1'} \).

Condition (2) is to ensure minimal inclusion of jobs to avoid a long tail. For a given 4-tuple \( (j_1, j_1', j_2, j_2') \), we can easily verify if it is a valid block. Let

\[
L(j_1, j_1', j_2, j_2') = \max \left\{ \sum_{k=j_1}^{j_1'} p_{1,k}, \sum_{k=j_2}^{j_2'} p_{2,k} \right\}
\]

denote the processing length of block \((j_1, j_1', j_2, j_2')\). Let \( \mathcal{B}(j_1, j_2) \) be the set of ordered pairs \((j_1', j_2')\) for which \( J_{1,j_1:j_1'} \) and \( J_{2,j_2:j_2'} \) constitute valid blocks.

Following the block structures defined above, we can then develop dynamic programming algorithms for the studied problems for minimizing the makespan. We first define state \( (j_1, j_2) \) to contain all feasible schedules formed by the jobs of sub-sequences \( J_{1,j_1:j_1'} \) and \( J_{1,j_2:j_2'} \), in which dummy jobs \( J_{1,n_1+1} \) and \( J_{2,n_2+1} \) with \( p_{1,n_1+1} = p_{2,n_2+1} = 0 \) are used to define the boundary conditions. Let function \( f(j_1, j_2) \) be the optimal makespan of the schedules in the state \((j_1, j_2)\). The dynamic programming algorithm is given as in the following:
In recurrence formula (1), the first term and the second term indicate the processing of a single job on machine $M_1$ or machine $M_2$. The third term is to take the minimum makespan among all recursions with valid blocks of $(j_1, j_1', j_2, j_2')$ for $(j_1', j_2')$ in $\mathcal{B}(j_1, j_2)$. We next analyze the run time of the dynamic program Algorithm 1. There are $O(n_1 n_2)$ states, each of which needs to examine $O(n_1 n_2)$ elements of $\mathcal{B}(j_1, j_2)$. Hence, the overall run time is $O(n_1^2 n_2^2)$, and the following theorem follows.

**Theorem 1:** PD2|conflict, seq1, seq2|C$_{\text{max}}$ can be solved in $O(n_1^2 n_2^2)$ time.

4. Conclusions

This paper studies a scheduling problem for two dedicated machines subject to conflict constraints. For the objectives of makespan minimization, the problem remains strongly NP-hard, even if the processing sequence on one machine is given and fixed. For the case where the processing sequences on both machines are given, we designed polynomial-time dynamic
programs. For future research, it may be interesting to investigate the complexity of for other objective functions, like total completion time. Another direction is to generalize the problem into the general case with an arbitrary number of dedicated machines.

References:
