Computers & Operations Research 37 (2010) 2141–2151

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Parallel machine scheduling with precedence constraints and setup times

Bernat Gacias [a,b], Christian Artigues [a,b], Pierre Lopez [a,b,*]

[a] CNRS; LAAS; 7 avenue du Colonel Roche, F-31077 Toulouse, France
[b] Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; F-31077 Toulouse, France

ARTICLE INFO

Available online 7 March 2010

Keywords:
Parallel machine scheduling
Setup times
Precedence constraints
Dominance conditions
Branch-and-bound
Limited discrepancy search
Local search

ABSTRACT

This paper presents different methods for solving parallel machine scheduling problems with precedence constraints and setup times between the jobs. These problems are strongly NP-hard and it is even conjectured that no list scheduling algorithm can be defined without explicitly considering jointly scheduling and resource allocation. We propose dominance conditions based on the analysis of the problem structure and an extension to setup times of the energetic reasoning constraint propagation algorithm. An exact branch-and-bound procedure and a climbing discrepancy search (CDS) heuristic based on these components are defined. We show how the proposed dominance rules can still be valid in the CDS scheme. The proposed methods are evaluated on a set of randomly generated instances and compared with previous results from the literature and those obtained with an efficient commercial solver. We conclude that our propositions are quite competitive and our results even outperform other approaches in most cases.

1. Introduction

This paper deals with parallel machine scheduling with precedence constraints and setup times between the execution of jobs. We consider the optimization of two different criteria: the minimization of the sum of completion times and the minimization of maximum lateness. These two criteria are of great interest in production scheduling. The sum of completion times is a criterion that maximizes the production flow and minimizes the work-in-process inventories. Due dates of jobs can be associated to the delivery dates of products. Therefore, the minimization of maximum lateness is a goal of due date satisfaction in order to disturb as less as possible the customer who is delivered with the longest delay. These problems are strongly *NP-hard* [1].

The parallel machine scheduling problem has been widely studied [2], specially because it appears as a relaxation of more complex problems like the hybrid flow shop scheduling problem or the RCPSP (resource-constrained project scheduling problem). Several methods have been proposed to solve this problem. In Chen and Powell [3], a column generation strategy is proposed. Pearn et al. [4] propose a linear program and an efficient heuristic for large-size instances for the resolution of priority constraints and family setup times problem. Salem et al. [5] solve the problem with a tree search method. More recently, Néron et al. [6] compare two different branching schemes and several tree search strategies for the problem with release dates and tails for the makespan minimization case.

However, the literature on parallel machine scheduling with precedence constraints and setup times is quite limited. Baev et al. [7] and van den Akker et al. [8] deal with the problem with precedence constraints for the minimization of the sum of completion times and maximum lateness, respectively. The setup times case is considered in Schutten and Leussink [9] and in Ovacik and Uzsoy [10] for the minimization of maximum lateness. Uzsoy and Velasquez [11] deal with the same criterion on a single machine with family-dependent setup times. Finally, Nessah et al. [12] propose a lower bound and a branch-and-bound method for the minimization of the sum of completion times.

Problems that have either precedence constraints or setup times, but not both, can be solved by list scheduling algorithms. It means there exists a total ordering of the jobs (i.e., a list) that, when a given machine assignment rule is applied, reaches the optimal solution [13]. For a regular criterion, this rule is called earliest completion time (ECT). It consists in allocating every job to the machine that allows it to be completed at the earliest. This reasoning unfortunately does not work when precedence constraints and setup times are considered together, as shown in Hurink and Knust [14]. We have then to modify the way to solve the problem and consider both scheduling and resource allocation decisions.

In this paper we propose to solve these problems through branch-and-bound for small-sized instances and local search for large-scale instances. To compensate search tree explosion due to machine assignment enumeration, we propose new constraint

* Corresponding author at: CNRS; LAAS; 7 avenue du Colonel Roche, F-31077 Toulouse, France.
E-mail addresses: bgacias@laas.fr (B. Gacias), artigues@laas.fr (C. Artigues), lopez@laas.fr (P. Lopez).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.03.003

propagation and dominance rules. For the local search method, we retain the principle of neighborhood exploration using a tree search structure in order to discard dominated solutions as soon as possible and we propose variants of the climbing discrepancy search method (CDS) proposed by Milano and Roli [15]. In particular we propose adapted dominance rules to avoid discarding solutions that do not have a dominant counterpart in the explored neighborhood.

In Section 2, we define formally the parallel machine scheduling problem with setup times and precedence constraints between jobs. The solution properties and the implications on branch-and-bound and local search are presented in Section 3. In Section 4 we present the branch-and-bound method and its components: tree structure, lower bounds, and dominance rules. Discrepancy-based tree search methods are described in Section 5. Section 6 is dedicated to computational experiments.

2. Problem definition

We consider a set J of n jobs to be processed on m parallel machines. The precedence relations between the jobs and the setup times, considered when different jobs are sequenced on the same machine, must be satisfied. The preemption is not allowed, so each job is continually processed during p_i time units on the same machine. The machine can process no more than one job at a time. The decision variables of the problem are the start times of every job $i=1\ldots n$, S_i, and let us define C_i as the completion time of job i, where $C_i=S_i+p_i$. Let r_i and d_i be the release date and the due date of job i, respectively. Due dates are only considered for job lateness computation. We denote by E the set of precedence constraints between jobs. The relation $(i,j)\in E$, with $i,j \in J$, means that job i is performed before job j ($i \prec j$) such that job j can start only after the end of job i ($S_j \geq C_i$). Finally, we define s_{ij} as the setup time needed when job j is processed immediately after job i on the same machine. Thus, for two jobs i and j processed successively on the same machine, we have either $S_j \geq C_i+s_{ij}$ if i precedes j, or $S_i \geq C_j+s_{ji}$ if j precedes i. Using the notation of Graham et al. [1], the problems under consideration are denoted: $Pm|prec,s_{ij},r_i|\sum C_i$ for the minimization of the sum of completion times and $Pm|prec,s_{ij},r_i|L_{\max}$ for the minimization of the maximum lateness.

Example

A set of five jobs ($n=5$) must be executed on two parallel machines ($m=2$). For every job i, we give p_i, r_i, d_i, and s_{ij} (see Table 1). Besides, for that example we have the precedence constraints: $1 \prec 4$ and $2 \prec 5$.

Table 1
Example 1 data.

(a)

n	p_i	r_i	d_i
1	4	1	7
2	3	0	5
3	4	3	8
4	3	3	10
5	2	1	5

(b)

s_{ij}	**1**	**2**	**3**	**4**	**5**
1	0	2	3	4	5
2	7	0	6	1	3
3	2	4	0	7	1
4	4	4	8	0	1
5	3	4	8	5	0

Fig. 1 displays a feasible solution for this problem. The set of precedence constraints is satisfied: $S_5=13 \geq 3=C_2$ and $S_4=5 \geq 5=C_1$. We stress that job 4 must postpone its start time on M_2 by one time unit because of the precedence constraint. On the other hand, we have to check that, for every job i, $r_i \leq S_i$ and that setup times between two sequenced jobs on the same machine are also respected. For the evaluation of the solution, we observe that for the minimization of the sum of completion times the value of the function is $z=\sum C_i=43$ and for the minimization of maximum lateness $z=L_{\max}=L_5=10$.

3. Solution properties and impact on branch-and-bound and local search

3.1. Solution properties

In Schutten [13], the author proves that the parallel machine scheduling problem with either setup times between jobs or precedence constraints can be solved to optimality by a list scheduling algorithm. He demonstrates that the schedules built using the list scheduling algorithm with earliest completion time assignment rule are dominant schedules.

However, precedence constraints and setup times parallel machine scheduling problems may not be efficiently solved by a list algorithm as conjectured by Hurink and Knust [14]. It means that there possibly does not exist a job assignment rule that reaches an optimal solution when all the possible lists of jobs are enumerated. Let us consider the minimization of the sum of completion times for four jobs scheduled on two parallel machines. The data of the problem are displayed in Table 2.

If we consider the problem without precedence constraints, we find two optimal solutions ($\sum C_i=9$) when we allocate the jobs following the earliest completion time rule for the lists {1,2,4,3} and {2,1,4,3} (see Fig. 2a). Now, let us consider the same problem with the precedence constraint $3 \prec 4$. In that case, there does not exist any allocation rule that reaches an optimal solution for any list of jobs that respects the precedence constraint. The optimal

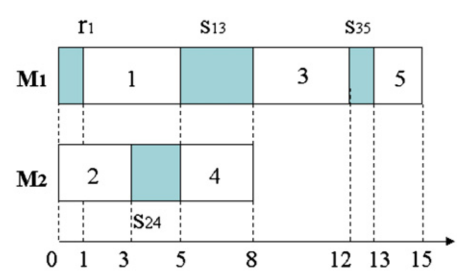

Fig. 1. Feasible schedule.

Table 2
Example 2 data.

(a)

n	p_i	r_i
1	1	0
2	1	0
3	1	2
4	1	2

(b)

s_{ij}	**1**	**2**	**3**	**4**
1	0	10	2	10
2	10	0	1	1
3	10	10	0	10
4	10	10	10	0