HAL open science

Tree and local search for parallel machine scheduling problems with precedence constraints and setup times

Bernat Gacias, Christian Artigues, Pierre Lopez

To cite this version:

Bernat Gacias, Christian Artigues, Pierre Lopez. Tree and local search for parallel machine scheduling problems with precedence constraints and setup times. PMS 2008, Apr 2008, Istanbul, Turkey. pp.79-84. hal-00278735

HAL Id: hal-00278735

https://hal.science/hal-00278735

Submitted on 13 May 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Tree and local search for parallel machine scheduling problems with precedence constraints and setup times

B. Gacias, C. Artigues and P. Lopez

LAAS-CNRS, Université de Toulouse, France
{bgacias,artigues,lopez}@laas.fr

Keywords: Parallel machine scheduling, setup times, precedence constraints, limited discrepancy search, local search.

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 the maximum lateness. These two criteria have a particular interest in production scheduling. The sum of completion times is a criterion that maximizes the production flow and makes possible the minimization of the work-in-process inventories. In the minimization of maximum lateness, the due dates can be associated to the delivery dates of products. This is a goal of due date satisfaction in order to punish as less as possible the customer who is delivered with the longest delay. These problems are strongly *NP-hard* (Graham et al., 1979). The parallel machine scheduling problem has been widely studied (Cheng and Sin, 1990), 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). However, the literature on parallel machine scheduling with precedence constraints and setup times is quite limited. The problems that only have either precedence constraints or setup times but not both, can be solved by list scheduling algorithms. That means, it exists a total ordering of the jobs (a list) that, when a given allocation rule is applied, reaches the optimal solution (Schutten, 1994). This rule is the Earliest Completion Time (ECT). It consists in allocating every job to the machine that allows it to be completed earlier. That reasoning is unfortunately unlikely to work when precedence constraints and setup times are considered together, as shown in Hurink and Knust (2001), so we have to modify the way to solve the problem and consider both scheduling and resource allocation decisions.

In Section 2, we define formally our particular case, the parallel machine scheduling problem with setup times and precedence constraints between jobs. The methods and techniques of local and tree search used to solve the problem are described in Sections 3 and 4. Section 5 is dedicated to the computational experiments.

2 Problem definition

We consider the following problem, in which a set J of n jobs needs to be processed on m parallel machines. The precedence relations between the jobs and the setup times, considered when differents jobs are sequenced on the same machine, must be respected. The preemption is not allowed, that means that 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 S_i, start time of job i, and C_i, completion time of job i, where $C_i = S_i + p_i$. r_i and d_i denote the release date and the due date of job i, respectively. We denote by E the set of precedence constraints between jobs. The relation $(i, j) \in E$, with i and $j \in J$, means that job i is performed before job j ($i \prec j$). So 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 of job j 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. The problems are then: $P|prec, s_{ij}|\sum C_i$ and $P|prec, s_{ij}|L_{\max}$.

3 A hybrid Tree-Local search method

3.1 Limited discrepancy tree search

To solve the problems under consideration, we use a method based on the discrepancies regarding a reference heuristic. Such a method is based on the assumed good performance of this reference heuristic, thus making an ordered local search around the solution given by the heuristic. First, it explores the solutions with few discrepancies from the heuristic solution and then it moves away from this solution until it has covered the whole search space. In this context, the principle of *LDS* (Limited Discrepancy Search) (Harvey and Ginsberg, 1995) is to explore first the solutions with discrepancies on the top of the tree, since it assumes that the heuristic makes the most important mistakes in the high levels where it still has taken very few decisions.

Several methods based on *LDS* have been proposed in order to increase the efficiency. For instance, *ILDS* (Korf, 1996), *DDS* (Walsh, 1997) or *DBDFS* (Beck and Perron, 2000), which have been devised to avoid the redundancy, and *YIELDS* (Karoui et al., 2007) where learning process notions are integrated.

3.2 Large neighborhood local search based on LDS

CDS (Climbing Discrepancy Search) (Milano and Roli, 2002) is a large neighborhood search method based on *LDS*. At each iteration it carries out a k-discrepancy search around the best current solution. If a better solution is found, then *CDS* takes its neighborhood as the new neighborhood to explore. In the case of no better solution is found, then k is increased by one. *CDDS (Climbing Depth-bounded Discrepancy Search)* mixes principles of *CDS* and of *DDS* (Hmida et al., 2007). The neighborhood of the best solution is limited not only by the number of discrepancies but also by the depth in the tree.

In this work, we propose two variants of *CDS* and *CDDS* for the problems at hand. *HD-CDDS (Hybrid Discrepancy CDDS)* consists in a search similar to *CDDS*. But, if for a defined depth level $dmax$ we cannot find a best solution, then we authorize a small number of discrepancies for all levels. This method solves the problem of incompatibility between the limitation by depth level and the precedence constraints. The second one, *MC-CDS (Mix Counting CDS)*, is an application of *CDS* but with a modification in the way to count the discrepancies. We consider a binary counting for the discrepancies at the top levels of the tree and a non-binary counting way for the rest of levels. We define in Section 4 the concept of binary and non-binary discrepancy counting as well as the other components of the *LDS* called at each iteration for the *CDS* local search method.

4 Branch-and-Bound components for $P|prec, s_{ij}|\sum C_i$ and $P|prec, s_{ij}|L_{\max}$

A tree structure with both levels of decisions (scheduling and resource allocation) is defined in 4.1. The exploration strategy (branching rules), the heuristics, and the definition of discrepancy are explained in 4.2. The specific methods of node evaluation like lower bounds, constraint propagation mechanisms and dominance rules are introduced in 4.3.

4.1 Tree structure

The problem cannot always be efficiently solved by a list algorithm since it includes precedence constraints and setup times together (Hurink and Knust, 2001). In our case we have not only to find the best list of jobs but also to specify the best resource allocation. For practical purposes, we have mixed both levels of decision: one branch is associated to the choice of the next job to schedule and also to the choice of the machine. One node represents a list of p jobs and a partial scheduling of these p jobs, and it entails maximum $(n-p)m$ child nodes. A solution is reached when we have a node with $p = n$. We suggest the following proposition in order to reduce the number of nodes to explore: for every job x having t (direct or undirect) successor jobs, we consider the assignments on the first $[\min(m, t+1)]^{th}$ machines that allows x to be completed as soon as possible. So, we are going to consider the schedule according to *ECT* rule for all jobs, except for the previous jobs. In that case, we consider to

schedule them on more than one machine to prevent that the previous jobs could avoid the best assignment for its successor jobs.

4.2 Exploration strategy

An initial solution is first obtained by the use of simple heuristics. For the job selection we use *SPT (Smallest Processing Time)* rule for $\min \sum C_i$ and *EDD (Earliest Due Date)* for $\min L_{\max}$. Once the job set, it is assigned to a machine according to *ECT (Earliest Completion Time)* heuristic.

Because of the existence of two types of decisions, we consider here two types of discrepancies: discrepancy on job selection and discrepancy on resource allocation. In the case of p-ary tree, we have two different ways to count the discrepancies. In the first mode (*binary*), we consider that choosing the heuristic decision corresponds to 0 discrepancy, while any other value corresponds to 1 discrepancy. The other mode (*non-binary*) consists in considering that the more far we are from the heuristic choice the more discrepancies we have to count. We suggest to test both modes for the heuristic for job selection. For these decisions, the heuristic is likely to make important errors, since the setup times are not considered and they have a main role in job scheduling. On the other hand, for the choice of the machine, we use the non-binary mode since we assume that the allocation heuristic only makes a few errors (*ECT* is a high-performance heuristic for this problem).

We propose three different branching rules. The first one, called *LDS-depth*, is a classical depth-first search but where the solutions obtained are limited by the allowed discrepancies. The other two strategies consider the number of discrepancies in the order the solutions are reached. The node to explore is the node with the less number of discrepancies, and with the smallest depth for the strategy called *LDS-top*, and with the largest depth for the strategy called *LDS-low*.

4.3 Node evaluation

A node evaluation differs depending on the studied criterion. For $\min \sum C_i$, it consists in computing a lower bound. We selected the bound suggested in Nessah et al. (2005), it is based on the resolution of a one-machine relaxation of the problem.

For $\min L_{\max}$, the evaluation consists in triggering a satisfiability test based on constraint propagation involving energetic reasoning (Lopez and Esquirol, 1996). The energy is produced by the resources and it is consumed by the jobs. We apply it to verify whether the best solution reached from the current node will be at least as good as the best current solution. We determine the minimum energy consumed by the jobs ($E_{consumed}$) over a time interval $\Delta = [t_1, t_2]$ and we compare it with the available energy ($E_{produced} = m(t_2 - t1)$); if $E_{consumed} > E_{produced}$ we can cut the branch. In our problem we also have to consider the energy consumed by the setup times. For an interval Δ where there is a set F of k jobs that consume, we can easily show that the minimum quantity of setups which occurs is $k - m$. So, we have to take the shortest $k-m$ setup times of the set $\{s_{ij}\}, i, j \in F$ into account and the energy consumed in an interval Δ is $E_{consumed} = \sum_i \max(0, min(p_i, t_2 - t_1, r_i + p_i - t_1, t_2 - d'_i + p_i)) + \sum_i^{k-m} s_{[ij]}$ where $s_{[ij]}$ are the setup times of the set $\{s_{ij}\}, i, j \in F$ sorted in non-decreasing order and $d'_i = Z_{best} + d_i$.

We also propose some dominance rules to solve the problems. They consist in trying to find whether there exists a dominant node, visited earlier or later, that allows us to prune the current node. The first one is a global dominance rule based on active schedules and max flow computation (Leus and Herroelen, 2003). The other two rules have been designed for being compatible with the allowed discrepancies. They are also based on active schedules. For a given schedule, the dominance rules search for a combination of jobs such that one job starts earlier ($S'_i < S_i$), and for the other jobs the start time cannot be delayed, ($S'_j \leq S_j, \forall j \neq i$). This combination of jobs has to be accepted for the number of authorized discrepancies.

5 Computational experiments

In the literature we have not found instances for this particular problem, so we propose to test the methods on a set of randomly generated instances. We first compare the

different proposed variants of the *LDS* method to determine the best one for being included inside the *CDS* scheme.

In the comparaison between the two different ways to count the discrepancies, binary and non-binary, we can say that the binary mode has shown a higher performance than the non-binary one. Out of a set of 60 instances, binary mode has found the best solution over 90 % of the instances, independently of the branching rule. For the three branch rules comparison we find that *LDS-top* (83.33 %) is the most efficient, since it reaches the best solutions more times and also with the shortest average search time.

For the evaluation of the lower bound and of the energetic reasoning we find that both allow the reduction of the search time and when the search cannot be finished we find better solutions when we use them (55 % for $lb(\sum C_i)$ and 87 % for *the energetic reasoning*) than we do not (45 % and 68 %, respectively). The best combinations for the node evaluation are the lower bound (for $\min \sum C_i$) and the energetic reasoning (for $\min L_{max}$) mixed with the local dominance rule (90 % and 93 %, respectively).

Finally, we compare the four variants of the hybrid tree local search methods (*CDS*, *CDDS*, *HD-CDDS*, *MC-CDS*) implemented with *LDS-top*, local dominance rule and binary counting (except for *MC-CDS* wich supposes a mix counting). *HD-CDDS* reaches the best solution for more instances (70 %) than the other methods and it also presents the smallest mean deviation from the best known solution (about 3 %).

6 Conclusion

In this paper we have studied limited discrepancy-based search methods and we have also proposed local search methods based on them. We have suggested an energetic reasoning scheme integrating setup times and we have proposed new global and local dominance rules that consider the discrepancies.

These methods could be used to solve more complex problems involving setup times, like the hybrid flow shop or the RCPSP.

References

J. C. Beck and L. Perron. Discrepancy-bounded depth first search. *In Second International Workshop on Integration of AI and OR Technologies for Combinatorial Optimization Problems (CP-AI-OR'00), Paderborn, Germany*, 2000.

T. Cheng and C. Sin. A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research, Vol.47:271-292*, 1990.

R.L Graham, E.L Lawler, J.K Lenstra, and A. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling:a survey. *Annals of Discrete Mathematics :287-326*, 1979.

W. D. Harvey and M. L. Ginsberg. Limited discrepancy search. *In Proceedings of 14th IJCAI*, 1995.

A. Hmida, M. J. Huguet, P. Lopez, and M. Haouari. Climbing depth-bounded discrepancy search for solving hybrid flow shop scheduling problems. *European J. of Industrial Engineering 1, No.2 : 223 - 243*, 2007.

J. Hurink and S. Knust. List scheduling in a parallel machine environment with precedence constraints and setup times. *Operations Research Letters 29: 231-239*, 2001.

W. Karoui, M.-J. Huguet, P. Lopez, and W. Naanaa. YIELDS: A yet improved limited discrepancy search for csps. *LNCS 4510, pp.99-111, Springer, 4th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR'07), Brussels, Belgium*, 2007.

R. Korf. Improved limited discrepancy search. *In Proceedings of 13th AAAI*, 1996.

R. Leus and W. Herroelen. Stability and resource allocation in project planning. *IIE Transactions,36:7,667-682*, 2003.

P. Lopez and P. Esquirol. Consistency enforcing in scheduling: A general formulation based on energetic reasoning. *5th International Workshop on Projet Management and Scheduling (PMS'96), pp.155-158, Poznan (Poland)*, 1996.

M. Milano and A. Roli. On the relation between complete and incomplete search: an informal discussion. *In Proceedings CPAIOR'02, Le Croisic, France*, 2002.

R. Nessah, Ch. Chu, and F. Yalaoui. An exact method for $Pm/sds, r_i/\sum C_i$ problem. *Computers and Operations Research 34: 2840-2848*, 2005.

J.M.J. Schutten. List scheduling revisited. *Operations Research Letters 18, 167-170*, 1994.

T. Walsh. Depth-bounded discrepancy search. *APES Group, Department of Computer Science*, 1997.