Finding a specific permutation of jobs for a single machine scheduling problems with deadlines

Thanh Thuy Tien Ta, Jean-Charles Billaut

Finding a specific permutation of jobs for a single machine scheduling problem with deadlines

TA Thanh Thuy Tien[1] and BILLAUT Jean-Charles[1]

Université de Tours, CNRS, LIFAT, ERL CNRS ROOT 6305, Tours, France.
jean-charles.billaut@univ-tours.fr, thanhthuytien.ta@etu.univ-tours.fr

Keywords: single machine, deadlines, lattice, new objective function.

1 Introduction

It is well known that a lot of scheduling problems have a huge number of optimal solutions. This is particularly true for some polynomial problems such as $1||L_{max}$, $1|r_j|C_{\max}$, $F2||C_{max}$, etc. (Smith, W.E. *et. al.* 1956). The purpose of this paper is to contribute to the characterization of all the optimal solutions of such a polynomial scheduling problem.

A general framework to characterize the set of optimal solutions has been proposed in (Billaut, J-C. *et. al.* 2011b) and (Billaut, J-C. *et. al.* 2012), based on the properties of the lattice of permutations (also called permutohedron). We consider in this paper the single machine scheduling problem with maximum lateness minimization, denoted by $1||L_{max}$ (Jackson, J-R. *et. al.* 1955). We assume that a pre-treatment in $O(n \log n)$ is performed so that the jobs are numbered in EDD order and due dates are modified into deadlines so that any optimal sequence has to be feasible with respect to these deadlines.

In the framework based on the lattice, one problem is to find a feasible sequence, as deep as possible. Indeed, any feasible sequence in the lattice is such that all its predecessors are also feasible (simple pairwise exchange argument) and it is possible to give easily the characteristics of all these predecessors. To denote the level of a feasible sequence in the lattice, a new function has been introduced and we want this level to be as small as possible. Let remember (see (Billaut, J-C. *et. al.* 2012)) that the top sequence is EDD with level $\frac{1}{2}n(n-1)$ and the bottom sequence is the inverse EDD sequence with level 0. Typically, if the inverse EDD sequence is feasible, it means that all the predecessors, i.e.e the n! sequences, are feasible.

The new objective function denoted by $\sum N_j$ has led to the introduction of some other new objective functions, based on the position of the jobs in the sequence, which have been studied in (Ta, T.T.Tien *et. al.* 2017a) and (Ta, T.T.Tien *et. al.* 2017b).

2 Definition of function $\sum N_j$ and first results

We consider a set of n jobs to schedule. To each job J_j, $1 \leq j \leq n$, is associated a processing time p_j and a deadline $\tilde{d}_j$. Without loss of generality, it is assumed that $\tilde{d}_1 \leq \tilde{d}_2 \leq ... \leq \tilde{d}_n$ and that sequence $EDD = (J_1, J_2, ..., J_n)$ is feasible.

Let σ be a sequence. The level of σ in the lattice is the number of couples (J_j, J_k) so that $j < k$ and J_j precedes J_k. Therefore, the contribution of J_j to this objective function is the number of jobs after J_j with an index greater than j. We denote this number by N_j.

Let suppose that $x_{j,k}$ is a binary variable equal to 1 if J_j is in position k. We have: $N_j = \sum_{i=j+1}^{n} \sum_{h=k+1}^{n} x_{i,h}$.

This objective function has other denominations in the litterature: the Kendall's tau distance (counts the number of pairwise disagreements between two ranking lists) and the

crossing number between the considered sequence and the inverse numbering sequence. Notice that a problem, presenting similarities with our problem, is proved NP-hard in (Biedl, T. *et. al.* 2005).

We can notice that this objective function does not depend on the jobs completion times, which is unusual in scheduling. This remark leads to some first (simple) results.

• **Problem** $1||\sum N_j$

Problem $1||\sum N_j$ (without due date or deadlines) is trivial. Scheduling the jobs in the reverse order of their numbering leads to a solution with $\sum N_j = 0$.

• **Problem** $1|p_j = p, \widetilde{d}_j|\sum N_j$

Let consider first the $1|p_j = 1, \widetilde{d}_j|\sum N_j$ problem and consider the following Backward algorithm (Alg. 1): schedule starting by the end the feasible job with minimum index. This algorithm solves problem $1|p_j = 1, \widetilde{d}_j|\sum N_j$ to optimality (the proof is admitted here).

It is easy to see that this algorithm can also solve problem $1|p_j = p, \widetilde{d}_j|\sum N_j$.

3 Properties and resolution methods for $1|\widetilde{d}_j|\sum N_j$

Property 1: An optimal solution can always be decomposed in a succession of batches defined as follows: the "head" of the batch is the last job of the batch ; the jobs in the batch are in decreasing numbering order and have an index greater than the head. Therefore, the index of the heads are increasing, starting with index 1.

Proof. admitted.

Exact resolution methods

For exact resolution, two MILP models were presented in (Billaut, J-C. *et. al.* 2012). The first model uses positional variables, the second model uses relative position variables.

In this paper, a branch-and-bound algorithm is proposed with some dominance rules.

The $B\&B$ method for $\sum N_j$ has the following characteristics. A node is defined by a partial sequence S of k jobs starting by the end of the schedule, a set of $n-k$ unscheduled jobs $\bar{S}$, a lower bound $LB(S)$, the index idx of the head of the current batch and t the starting time of the jobs in S: $t = \sum_{J_j \in \bar{S}} p_j$.

At the root node, the unscheduled jobs are $\{J_n, J_{n-1}, ..., J_1\}$. The initial upper bound UB is given by a Backward algorithm of the same type as Alg. 1. The strategy of branching consists in adding a job of $\bar{S}$ in first position of S, respecting the deadlines, and the exploration is done by $depth - first$ (the list of nodes is managed as a LIFO list).

Some dominance rules are used for this method. Let consider a current node and let us denote by J_ℓ the first job in S and by J_h the job in $\bar{S}$ to schedule before J_ℓ. The child node is created only if $\tilde{d}_h \geq t$. Furthermore, if $h < \ell$ and $h > idx$, the node is not created (see Property 1). If $h < \ell$ and $h < idx$, the idx of the child node is set to h. If $h = 1$, the sequence is completed by the jobs in $\bar{S}$ in their inverse numbering order and this node is considered immediately as a leaf of the tree (see Property 1).

The lower bound works as follows: a dummy sequence is built with the jobs in $\bar{S}$ in reverse number ordering, plus the jobs in S. The evaluation of this a priori non feasible sequence is the lower bound. However, if the set of unscheduled jobs is $(J_n, J_{n-1}, ..., J_1)$ in this order, it is possible to compute the lower bound in $O(1)$ time.

Heuristic and metaheuristic methods

Two polynomial time heuristic methods are proposed: a Backward algorithm (denoted BW, Alg. 1) and a Forward algorithm (denoted FW). BW builds a solution by the end, putting in last position the feasible job with the smallest index; FW takes the jobs in EDD

order, put each job as late as possible and insert the feasible job with the biggest index before it.

Two metaheuristic methods are proposed: a Tabu search (denoted TS) and a Simulated Annealing (SA), with several (common) neighborhoods operators. The initial solution of TS and SA is the best solution of BW and FW.

4 Computational experiments

After a study about a related problem based on jobs positions, which was proved to be strongly NP-hard ((Ta T.T.Tien, *et. al.* 2017a), (Ta, T.T.Tien, *et. al.* 2017b)), two types of instances were generated. One type of pure random instances, and one type of "difficult" instances. Even if the problems are not the same, we kept these data for our computational experiments.

Data sets For each type of instance, 30 instances have been generated for each value of n, with $n \in \{10, 20, ..., 100\}$.

• For the instances of **type I**, random data sets have been generated as follows: $p_j \in [1, 100]$, $w_j \in [1, 100]$, $d_j \in [(\alpha - \beta/2)P, (\alpha + \beta/2)P]$, with $P = \sum p_j$, $\alpha = 0.75$ and $\beta = 0.25$.

These instances receive a pre-treatment: (1) EDD rule is applied, giving L^*_{max}. Then, (2) due dates are modified to give deadlines: $\tilde{d}_j = d_j + L^*_{max}$, for any $j \in \{1, 2, ..., n\}$, limiting the deadlines to $\sum p_j$. Finally, (3) the jobs are renumbered in EDD order.

• For the instances of **type II**, random data sets have been generated as follows:

For $n' = \lfloor n/4 \rfloor$ jobs: $p_j = 1$; $w_j = 0$; $\tilde{d}_j = 4jP/n$

For the $(n - n')$ remaining jobs: $p_j \in [1, 100]$, $w_j = w_{0j} + P$, with $w_{0j} \in [1, 100]$ and $P = \sum p_j$; $\tilde{d}_j = P + \lfloor n/4 \rfloor$

These instances do not need the pre-treatment.

Results The computational experiments have been run on a HP ProBook, Intel(R) Core(TM) i5-6300 CPU @ 2.40GHz 2.50 GHz, RAM 16,0Go, System style 64 bit. The MILP models have been solved by IBM ILOG CPLEX 12.6. The CPU time to solve each instance has been limited to 180 seconds for all the resolution methods. Results for instances of type I and II are presented in Table 1. Columns MILP1, MILP2 and $B\&B$ concern the exact methods, 'cpu' indicates the average computation time and 'opt' indicates the number of instances solved to optimality in less than 180 seconds. The other columns concern the heuristic methods. Columns '$N°$B' indicate the number of times the method is the best among all the methods, and $\Delta B1$ is a relative deviation defined by: $MIN = \min(MIP1, MIP2, B\&B, BW, FW)$ and $\Delta B1(H) = \frac{H - MIN}{H}, \forall H \in \{BW, FW, TS, SA\}$

For Type I instances, one can see that MIP1 is better than MIP2 for small instances, but B&B is the best exact method, solving quite all instances up to 70 jobs. With 90 jobs the $B\&B$ remains interesting but for larger instances, the best method is the Simulated Annealing algorithm. For Type II instances, one can see that the exact methods are limited to instances with up to 20 jobs. Among the heuristic algorithms, BW is the best method and the Tabu Search and the Simulated Annealing are not able, in the limited computation time of 180 seconds, to improve the initial solution.

Table 1. Results of Type I & II instances

n	MIP1 cpu (s)	MIP1 opt	MIP2 cpu (s)	MIP2 opt	B&B cpu (s)	B&B opt	BW N°B	BW $\Delta B1$ (%)	FW N°B	FW $\Delta B1$ (%)	TS N°B	TS $\Delta B1$ (%)	SA N°B	SA $\Delta B1$ (%)
Results of Type I instances														
10	0,26	30	0,27	30	3.10^{-5}	30	22	2,00	28	0,43	30	0	30	0
20	47,2	30	105	20	4.10^{-4}	30	5	15,36	13	4,21	26	0,36	20	0,65
30	180	0	180	0	5.10^{-3}	30	1	20,66	5	6,50	13	2,39	11	1,03
40	180	0	180	0	0,014	30	0	24,09	1	8,23	13	4,04	7	1,55
50	180	0	180	0	0,077	30	0	28,92	0	7,44	5	3,68	0	1,72
60	180	0	180	0	2,391	30	0	28,90	0	7,11	10	2,42	0	1,28
70	180	0	180	0	23,79	29	0	28,49	0	7,64	3	3,59	1	1,37
80	180	0	180	0	127,3	15	0	31,33	0	7,07	7	2,90	7	0,44
90	180	0	180	0	174,5	1	0	25,85	0	1,76	14	-2,42	13	-3,30
100	180	0	180	0	180	0	0	28,79	0	0,30	12	-1,62	19	-3,10
Results of Type II instances														
10	0,001	30	0,1	30	2.10^{-3}	30	28	0,58	6	13,39	30	0	30	0
20	0.504	30	111	20	33,21	29	30	0	0	26,14	30	0	30	0
30..100	180	0	180	0	180	0	30	0	0	$\simeq$27%	30	0	30	0

5 Conclusions and Perspectives

In this paper, we have identified a new category of scheduling problems, with the definition of a new objective function. Some trivial problems are identified but the general problem with deadlines remains open. We propose some exact and exponential methods, as well as heuristic and meta-heuristic algorithms. These methods are evaluated by some computational experiments on randomly generated instances. In the future, we will continue to improve the exact methods by introducing cuts and more dominance conditions, but the most important point is to investigate the complexity of the general problem.

References

Biedl, T., Brandenburg, Franz J., Deng X., "Crossings and Permutations", In: Healy P., Nikolov N.S. (eds) Graph Drawing. GD 2005. Lecture Notes in Computer Science, vol 3843. Springer, Berlin, Heidelberg, 2005.

Billaut, J-C., Lopez, P. 2011b, "Characterization of all $\rho-$approximated sequences for some scheduling problems", *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, art.* . No. 6059026.

Billaut, J-C., Hebrard, E. and Lopez, P. 2012, "Complete Characterization of Near-Optimal Sequences for the Two Machine Flow Shop Scheduling Problem", *Ninth International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming (CPAIOR'2012), Nantes, June 2012.*

Jackson, J-R., 1955 "Scheduling a production line to minimize maximum tardiness", *Research report 43, Management Science Research Project, University of California, Los Angeles, 1955.*

Smith, W.E. 1956 "Various optimizers for single stage production", *Naval Research Logistics Quarterly*, 3(1-2):59-66, 1956.

Ta, T.T.Tien, Billaut, J-C., Febuary 2017 "Characterization of optimal solutions for some problems scheduling", *ROADEF Conference, Metz, France.*

Ta, T.T.Tien, Ringeard, K., Billaut, J-C., October 2017 "New objective functions based on jobs positions for single machine scheduling with deadlines ", *7th IESM Conference, Saarbrucken, Germany.*