Exact and heuristic methods for characterizing optimal solutions for the $1||L_{max}$

Tifenn Rault, Ronan Bocquillon and Jean-Charles Billaut

Université de Tours, LIFAT EA 6300, CNRS, ROOT ERL CNRS 7002
64 Avenue Jean Portalis, 37200 Tours
tifenn.rault, ronan.bocquillon, jean-charles.billaut@univ-tours.fr

Keywords: scheduling, single machine, characterization of solutions, deadlines, lattice.

1 Introduction

The $1||L_{max}$ problem can be solved in $\mathcal{O}(n \log n)$ applying EDD rule (Jackson 1955). Still, as many scheduling problems, $1||L_{max}$ may have a huge number of optimal solutions. Our objective is to characterize easily a set of optimal solutions, without enumerating them. Such an approach is useful in a dynamic environment, to react in real time to unexpected events, or to data uncertainty. Some preliminary studies in this direction have been conducted using the lattice of permutations as support (Billaut and Lopez 2011, Billaut *et. al.* 2012, Ta 2018). In such a framework, it is assumed that jobs are renumbered in EDD order and due dates are transformed in deadlines so that sequence EDD is feasible and is the root sequence of the lattice. A property is that all the predecessors of a feasible sequence in the lattice are feasible as well, and therefore optimal sequences for the $1||L_{max}$. These predecessors can be characterized very easily by a partial order between jobs. The distance to the bottom sequence (reverse EDD sequence), also called "level", is denoted by $\sum N_j$. It is expected that a sequence with minimum level will characterize a lot of optimal solutions. In this paper, we study the problem $1|\widetilde{d}_j|\sum N_j$ of finding a sequence with minimum level. We improve the existing branch and bound algorithm introduced in (Ta 2018) thanks to memorization, and we propose a new heuristic method. Our results show that we improve state-of-the-art resolution methods, solving to optimality new instances.

2 Problem description

We consider a set of n jobs $J_j, 1 \leq j \leq n$. To each job is associated a processing time p_j and a due date d_j. For any instance, we apply the following pre-treatment in polynomial time: (1) we apply EDD rule and we obtain L^*_{max}, the optimal maximum lateness value. Then, (2) we modify the due dates in order to obtain deadlines: $\widetilde{d}_j = \min(d_j + L^*_{max}, \sum p_j)$, for any $j \in \{1, 2, .., n\}$. Finally, (3) we renumber the jobs in EDD order, and in case of a tie, in LPT order. At the end we know that sequence $\sigma = (J_1, J_2, ..., J_n)$ is feasible and $i < j \iff (\widetilde{d}_i < \widetilde{d}_j)||(\widetilde{d}_i = \widetilde{d}_j, p_i \geq p_j)$.

The lattice of permutations is a digraph constructed as follows. The root node is the EDD sequence, i.e $(J_1, J_2, ..., J_n)$. The children of a sequence σ are created by inverting two jobs J_k and J_l iff J_k is just before J_l in σ, and $k < l$. The process is repeated for newly created sequences until the sink node (i.e reverse EDD sequence) is reached. Each sequence in the lattice can be associated with a *level* which represents the number of inversions from the reverse EDD sequence at level 0, i.e the number of times we have J_i before J_j and $i < j$ (see (Billaut *et. al.* 2012) for more details). The level of a sequence also gives the number of precedence relations characterized by this sequence. One nice property of this lattice is that all predecessors of a feasible sequence are also feasible (and therefore optimal for the $1||L_{max}$).

Finding a sequence as deep as possible in this lattice, or finding a sequence as far as possible from EDD sequence, is the same, and is equivalent to minimising the objective function $\sum N_j$. The variable N_j, the contribution of job J_j to the objective function, is equal to the number of jobs after J_j with and index greater than j. The complexity of problem $1|\widetilde{d}_j|\sum N_j$ remains open (Ta 2018). It has been shown in (Ta *et. al.* 2017, Ta 2018) that the expression $\sum N_j$ is somewhat related to the positions P_j of the jobs, and the problem $1|\widetilde{d}_j|\sum w_j P_j$ is proved strongly NP-hard by reduction from Numerical 3-Dimensional Matching problem.

3 Exact method

The original B&B method for the $1|\widetilde{d}_j|\sum N_j$ problem works as follows. Consider first the initial set of unscheduled jobs in reverse numbering order $S = \{J_n, J_{n-1}, ..., J_1\}$, and $\sigma = \emptyset$ the sequence being build, starting by the end. At each level, add a new unscheduled job J_j in first place of σ. While $S \neq \emptyset$, the job J_j is chosen in S from the smallest to the biggest index so that: the deadlines are respected (put only in the first position of sequence σ a non tardy job), and the dominance conditions too (keep only the sequences satisfying dominance properties).

The initial upper bound is given by a polynomial time heuristic method. The lower bound is computed in $O(1)$. Indeed, when adding a job J_j in front of σ, the contribution of J_j to the level of all unscheduled jobs is exactly its position in S minus 1. Several properties of optimal (or feasible) sequences are presented in (Ta 2018) and exploited by the original Branch-and-Bound. Two new dominance rules are presented hereafter.

Property 1 *Two jobs J_i and J_j with identical deadlines and $i < j$ must be sequenced so that J_i is after J_j in the sequence.*

Sketch of the proof Let us consider two jobs J_i and J_j such that $\widetilde{d}_i = \widetilde{d}_j$ and $i < j$. Due to our renumbering scheme (see Section 2), we know that $p_i \geq p_j$. Suppose sequence is built in a backward manner. If a choice has to be made between J_i or J_j, it is always preferable to place J_i since it will possibly enable to place a job with a smaller index (i.e. a smaller deadline) right after, and J_i has a smallest index than J_j, thus decreasing the objective function.

Property 2 *If two partial solutions contain exactly the same subset of jobs, then the one with the lowest level dominates the other.*

Sketch of the proof Suppose we have two partial solutions σ_u and σ_v such that σ_v contains exactly the same subset of jobs as σ_u (but in different order), and $level(\sigma_u) = \sum_{i \in \sigma_u} N_i \leq \sum_{j \in \sigma_v} N_j = level(\sigma_v)$. We have $S_u = S_v = S$. Let S^* be the optimal way of arranging the jobs in S, s_u^* be the concatenation of S^* and σ_u, and s_v* be the concatenation of S^* and σ_v. We have $level(s_u^*) = level(S^*) + level(\sigma_u)$ and $level(s_v^*) = level(S^*) + level(\sigma_v)$. So, $level(s_u^*) \leq level(s_v^*)$: the partial solution σ_u dominates σ_v.

We denote by B&BP1 the B&B integrating property 1, and we call "Branch-and-Bound with memorization" (denoted B&BM) the B&B integrating properties 1 and 2. To leverage property 2, the set of jobs in σ of visited nodes must be stored, so that dominated nodes can be pruned. In practice, it requires a balance between time gain and memory storage. This method takes as a parameter the maximum cardinality of the recorded sets. Typically, in the results table, B&BM x means that only sets whose cardinality is less than x percent of n are registered. This strategy is motivated by the fact that the more a branch is high, the more pruning it will have an impact.

4 Heuristic method

In addition to the exact method, we investigated a new heuristic strategy based on the Limited Discrepancy Search (LDS). For large search tree, exhaustive search is not tractable. The LDS technique was introduced by Harvey and Ginsberg (Harvey and Ginsberg 1995) to cope with this limitation. A "discrepancy" is a right branch in a heuristically ordered tree. Originally LDS is an iterative procedure. It first generates the leftmost path. Next, it generates those paths with at most one right branch from the root to the leaf. The next set of paths generated by LDS are those with at most two right branches, etc. The process continues until every path has been generated, with the rightmost path being generated last.

This technique seems well suited to our problem since we expect that in the search tree, the feasible solutions with the lowest levels are located among the leaves as far as possible on the left of the search tree. Indeed, the levels of the solutions at the leaves are *almost* distributed from the smallest to the largest, from the left to the right. In LDS-k, we therefore allow during the search the generation of paths with at most k right branches. Note that we use it as a non-iterative heuristic. We can combine LDS-k with property 2 in order to cut branches, and we denote this technique LDS-k M-x, where x means that only sequences whose size is less than x percent of n are registered.

5 Computational results

We compare the original B&B with B&BP1 and with B&BM. We also compare the LDS heuristic with the backward heuristic (denoted BW) introduced in (Ta 2018), which builds the solution by the end, putting in last position the feasible job with the smallest index. We used type I data sets used in (Ta 2018). For each value of n, with $n \in \{10, 20, ..., 100\}$, there are 30 instances. The experiments were run on Intel E5-2670V2 processors running at 2.5GHz (one core per instance). The memorization database is capped at 10^7 entries and the CPU time to solve each instance has been limited to 180 seconds.

Table 1. Performances of exact methods

	(Ta 2018)		**B&BP1**		**B&BM 30**		**B&BM 70**		**B&BM 90**	
n	**opt**	**cpu**	**opt**	**cpu**	**opt**	**cpu**	**opt**	**cpu**	**opt**	**cpu**
10	30	0,00	30	0,00	30	0,00	30	0,00	30	0,00
20	30	0,00	30	0,00	30	0,00	30	0,00	30	0,00
30	30	0,00	30	0,00	30	0,00	30	0,00	30	0,00
40	30	0,00	30	0,00	30	0,00	30	0,00	30	0,00
50	30	0,00	30	0,00	30	0,01	30	0,01	30	0,01
60	30	0,17	30	0,14	30	0,12	30	0,12	30	0,12
70	30	1,88	30	**1,52**	30	**0,61**	30	**0,61**	30	**0,62**
80	28	36,17	**30**	**29,38**	**30**	**6,95**	**30**	**6,81**	**30**	**6,98**
90	12	133,31	12	**128,49**	**17**	**100,26**	**17**	**98,13**	**18**	**99,17**
100	1	175,59	1	175,27	**2**	**170,15**	**2**	**170,21**	**2**	**170,21**

Table 1 compares the performances of the exact methods. It can be seen that we solved to optimality 9 new instances for $n \geq 80$. More precisely, Property 1 allows us to solve two new instances, while B&BM x solves more instances as parameter x increases.

Table 2. Performances of heuristic methods

	BW	**LDS-1**		**LDS-1 M-30**		**LDS-2**		**LDS-2 M-30**	
n	**gap**	**cpu**	**gap**	**cpu**	**gap**	**cpu**	**gap**	**cpu**	**gap**
10	2,0%	0,00	0,0%	0,00	0,0%	0,00	0,0%	0,00	0,0%
20	20,8%	0,00	1,7%	0,00	1,7%	0,00	0,0%	0,00	0,0%
30	27,3%	0,00	7,4%	0,00	7,4%	0,00	1,1%	0,00	1,1%
40	30,5%	0,00	8,8%	0,00	8,6%	0,00	2,0%	0,00	2,0%
50	42,6%	0,00	15,4%	0,00	14,8%	0,00	4,4%	0,00	4,3%
60	41,0%	0,00	15,3%	0,01	13,9%	0,00	6,6%	0,01	6,5%
70	41,6%	0,03	17,8%	0,10	16,0%	0,01	8,3%	0,07	7,8%
80	45,9%	0,06	23,3%	1,31	18,6%	0,03	12,0%	0,87	10,1%
90	42,0%	5,08	21,2%	28,73	15,5%	2,08	10,6%	23,70	7,4%
100	39,6%	47,18	15,3%	111,04	9,5%	34,93	8,2%	127,61	1,1%

Regarding the heuristic methods, Table 2 shows that the LDS-k approach exhibits lowest gap than the backward heuristic. For $n \geq 70$, this comes at an increased computation time. Still we can point out that LDS-2 and LDS-2 M-30 offer a good trade-off between computation time and solution quality for $n \geq 90$. Indeed, it provides solutions with gaps between $1.1\% - 10.6\%$ faster than the B&B approach.

6 Conclusion

In this paper, we addressed the problem $1|\widetilde{d_j}|\sum N_j$ whose complexity remains open. We introduced two new efficient dominance rules that allowed us to solve new instances to optimality. A new heuristic is presented. Its performances offer a good trade-off between computation time and solution quality.

Future directions include generating larger and harder instances, and studying different policies for choosing which sets to remove from the memorization database when it is full.

References

Billaut J.-C, P. Lopez, 2011, "Characterization of all p-approximated sequences for some scheduling problem", *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA.*

Billaut J.-C., E. Hebrard and P. Lopez, 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, France.*

Harvey W. D., M. L. Ginsberg, 1995, "Limited Discrepancy Search", *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 607-613.

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

Ta T.T.T., R. Kivin and J.-C. Billaut, 2017, "New objective functions based on jobs positions for single machine scheduling with deadlines", *7th International Conference on Industrial Engineering and Systems Management (IESM 2017), Saarbrucken, Germany.*

Ta T.T.T., 2018, "New single machine scheduling problems with deadlines for the characterization of optimal solutions", Thèse, *Université de Tours.*