An exact method for the bi-objective one-machine problem with maximum lateness and unit family setup cost objectives

Christian Artigues, Nicolas Jozefowiez, Mohamed Ali Aloulou

An exact method for the bi-objective one-machine problem with maximum lateness and unit family setup cost objectives

Christian Artigues [a, b,], Nicolas Jozefowiez [a, b, 1]

[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*

Mohamed Ali Aloulou [c, d, 2]

[c] *LAMSADE; Université Paris Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France*

[d] *CNRS, FRE 3234, F-75016 Paris, France*

Abstract

This paper deals with an NP-hard bi-objective one-machine problem with ready times involving maximum lateness and unit family setup cost objectives. Considering separately both objectives, the maximum lateness one-machine problem is also NP-hard but efficiently solved by Carlier's algorithm while the unit family setup cost one machine-problem with two families can be solved in polynomial timeby Darte's algorithm, even when precedence constraints are considered. Under the $\epsilon-$constraint framework we propose a branch-and-bound method to minimize the first objevtive with a given upper bound on the second.

Keywords: bi-objective one-machine scheduling, maximum lateness, unit family setup costs, branch-and-bound.

1 Introduction

In this paper, we define a bi-objective scheduling problem and design an exact multi-objective algorithm to solve it. We consider a one-machine scheduling problem, with each job being associated to a ready time, a due date, and a color (or family type) out of two possible colors. Any color change on the machine involves a unit cost. Therefore, we have two objectives: the minimization of the maximum lateness and the number of switches between the two colors. Considering separately both objectives, the maximum lateness one-machine problem is NP-hard, albeit efficiently solved by Carlier's branch-and-bound algorithm [2] while the unit family setup cost one machine-problem with two families can be solved in polynomial time, even when precedence constraints are considered by Darte's algorithm [3]. We consider here the bi-objective problem, which is NP-hard as one of the single-objective problem is NP-hard. Formally, a multi-objective problem can be stated as follows: $(MOP) : \min_{x \in D} \; F(x) = (f_1(x), f_2(x), \ldots, f_n(x))$, where $n \geq 2$ is the number of objective functions, $x = (x_1, x_2, \ldots, x_r)$ is the decision variable vector or solution, D is the feasible solution space, and $F(x)$ is the objective vector. The set $O = F(D)$ corresponds to the images of the feasible solutions in the objective space, and $y = (y_1, y_2, \ldots, y_n)$, where $y_i = f_i(x)$, is a point of the objective space. A MOP solution is the set of non-dominated solutions called the Pareto set (PS), or non-dominated set. Dominance is defined as follows:

Definition 1 *A solution x dominates a solution z if and only if $\forall\, i \in \{1 \ldots n\}$, $f_i(x) \leq f_i(z)$ and $\exists\, i \in \{1 \ldots n\}$, such that $f_i(x) < f_i(z)$.*

The main contributions of this paper are the modelization of the problem by means of an integer program, the definition of a branch-and-bound procedure based on Carlier's algorithm to solve the minimum maximum lateness problem with a given upper bound on the total setup cost, denoted $1|r_i, SC_{si,b} = 1|\epsilon(L_{\max}|TSC)$ under the notations proposed by [1,4] where Lmax and TSC stand for maximum lateness and total setup cost, respectively, and its use in an ϵ-constraint method to generate the optimal Pareto set. The paper is organized as follows. The problem is defined and modeled in Section 2. The single objective algorithms for each objective [2,3] are also described. The branch-and-bound algorithm and the ϵ-constraint method are described in Section 3. where Lmax and TSC stand for maximum lateness and total setup cost, respectively.

[1] Email: christian.artigues, nicolas.jozefowiez@laas.fr
[2] Email: Mohamed-Ali.Aloulou@dauphine.fr

2 Problem

2.1 Mathematical model

The following integer program models the problem. Let V be the set of jobs to schedule. For each job $i \in V$, r_i is its ready time, d_i its due date, and p_i its duration. s and f are two fictitious jobs representing the start and the end of the schedule respectively. For two jobs $i \in V$ and $j \in V$, c_{ij} is equal to 1 if i and j are not of the same color, 0 otherwise. For each job $i \in V$, t_i is the start time of i. For each pair of jobs i and j in $V \cup \{s, t\}$, x_{ij} is a binary variable equal to 1 if and only if j is scheduled immediately after i. $L_{\max}$ is maximum lateness. Then, the problem can be formulated as follows:

$$\text{(1)} \quad \min \quad L_{\max}$$

$$\text{(2)} \quad \min \quad \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

$$\text{(3)} \quad \sum_{j \in V \cup \{f\}} x_{ij} = 1 \qquad (i \in V)$$

$$\text{(4)} \quad \sum_{j \in V \cup \{s\}} x_{ji} = 1 \qquad (i \in V)$$

$$\text{(5)} \quad \sum_{i \in V} x_{si} = 1$$

$$\text{(6)} \quad t_j \geq t_i + p_i - M(1 - x_{ij}) \quad (i, j \in V)$$

$$\text{(7)} \quad L_{\max} \geq t_i + p_i - d_i \qquad (i \in V)$$

$$\text{(8)} \quad t_i \geq r_i \qquad (i \in V)$$

$$\text{(9)} \quad x_{ij} \in \{0, 1\} \qquad (i \in V \cup \{s\}, j \in V \cup \{p\})$$

Objectives (1) and (2) respectively minimizes the maximum lateness and the number of switches. Constraints (3) verify that only a single job succeeds immediately a job i. Constraints (4) work similarly to identify the job immediately preceeding a job i. Contraint (5) initializes the flow. Constraints (6) compute the earliest possible starting times according the sequence (M is a large value). Constraints (7) set the value of the maximum lateness. Constraints (8) and (9) bound the variables.

2.2 Minimizing the total lateness [2]

When the maximum lateness objective is considered, the problem (denoted $1|r_i|L_{\max}$) can be efficiently solved by Carlier's algorithm [2], which is a branch

and bound method based on the following components. First, the problem is transformed by replacing the due dates d_i by tails q_i with $q_i = N - d_i$ for each job $i \in V$ where N is an arbitrary constant. The considered objective is then a modified makespan $C_{qmax} = \max_{i \in V}(t_i + p_i + q_i)$. At each node, a lower bound is obtained by computing the Jackson's premptive schedule, which is an optimal solution of problem $1|r_i, pmtn|L_{\max}$ a relaxation of $1|r_i|L_{\max}$ allowing job preemption. Branching is done by modifying release times (heads) or tails according to the following principle. A feasible solution is computed at each node by Shrage's algorithm, a list scheduling heuristic, which, at each decision time t (equal to a release date or a job completion time), selects the available job with the largest tail. From the Schrage solution, a critical block (a set of consecutive jobs $J^* = \{i_1, \ldots, i_m\}$ verifying $C_{qmax} = r_{i_1} + \sum_{a=1}^{m} p_a + q_{i_m}$) is extracted. If all but last jobs of this block i_a, $a < p$ verify $q_{i_a} \geq q_{i_p}$ the schrage solution is optimal w.r.t. modified heads and tails and the node is pruned. Otherwise, the "pivot" job i_c verifying $q_{i_c} < q_{i_p}$ and c is maximal in $[1, p]$ is identified and subblock $J_c = \{i_{c+1}, \ldots, i_p\}$ is considered. The properties of Schrage's solution allow to define a binary search tree. Indeed, the objective can only be improved by sequencing i_c *before* all jobs of J_c ($i_c \prec J_c$) which can be ensured during the Schrage algorithm process by setting

$$q_{i_c} := \max(q_{i_c}, \sum_{k=c}^{m} p_{i_k} + q_{i_p}) \tag{10}$$

or *after* all jobs of J_c ($J_c \prec i_c$) by setting

$$r_{i_c} := \max(r_{i_c}, \min_{i_k \in J_c} r_{i_k} + \sum_{k=c}^{m} p_{i_k}) \tag{11}$$

2.3 Minimizing the number of color switches under precedence constraints [3]

Without precedence constraints between jobs, minimizing the color switches is trivial. With precedence constraints, $1|r_i, prec, SC_{si,b}|TSC$ can be solved in polynomial time by Darte's algorithm [3] as follows. The method computes only two solutions. For the first solution, one of the two colors is selected, say $c1$. All jobs of color $c1$ without predecessor are scheduled jointly in any order and their precedence constraints are deleted. Then a color change must occur and the process restart with $c2$ jobs, until all jobs are scheduled. For solution 2, $c2$ is selected first. The solution with the minimum number of color switches is selected.

3 Algorithms

3.1 ϵ-constraint method

In the bi-objective case, the ϵ-constraint method adds a new constraint to the problem : $f_i(x) \leq \epsilon$, where f_i is one objective and ϵ is a given value, and it works to optimize only the second objective. Varying the ϵ parameter allows different problems with different solutions to be generated. If the problems are solved with an exact algorithm, the solutions are therefore Pareto optimal solutions.

In our case, we transform the objective minimizing the number of switches into a constraint and obtain the problem $1|r_i, SC_{si,b} = 1|\epsilon(L_{\max}|TSC)$, where TSC stands for total setup cost. For a given value of ϵ, the problem is solved by the branch-and-bound algorithm described in 3.2. The following algorithm ensures that all the non-dominated points are found and that there is no computaiton reduntancy by selecting the values of ϵ properly.

STEP 1 Set $\epsilon \leftarrow |V| - 1$ (i.e. the maximum number of switches possible).

STEP 2 If $\epsilon = 1$, the algorithm is over. Solve $1|r_i, SC_{si,b} = 1|\epsilon(L_{\max}|TSC)$ by branch-and-bound (see 3.2)

STEP 3 The solution is stored as a solution in the non-dominated set and ϵ is fixed to $\epsilon^* - 1$ where ϵ^* is the number of switches in the solution found by the branch-and-bound algorithm. Go to *STEP 2*.

3.2 Branch-and-bound algorithm for $1|r_i, SC_{si,b} = 1|\epsilon(L_{\max}|TSC)$

The branch and bound we propose is based on the integration of Carlier's and Darte's algorithm [2,3] described in Sections 2.2 and 2.3 to tackle constraint $TSC \leq \epsilon$.

Initial solution. An initial solution is computed by a modified Schrage's algorithm: In case of tie for the maximum tails, a job of the same color as the previously scheduled one is selected. Let a "group" be a maximal set of consecutive jobs having the same color in a given sequence. Once the solution (not necessarily respecting $TSC \leq \epsilon$) is obtained, a local search method is carried out. A move consist in inserting all the jobs of a group into another group of identical color. At each step, among all possible group merges, the one involving the lower maximum lateness increase is selected. The process stops as soon as $TSC \leq \epsilon$. Then, inside each group, the Shrage's algorithm is applied to further reduce the maximum lateness. This solution initializes the upper bound, which can also be set by a previously found solution inside the

$\epsilon-$constraint method (see "Non-dominated solutions storing" below).

Branching scheme As for Carlier's algorithm we propose a branching scheme based on necessary conditions to improve a feasible solution computed at each node. One of the major difficulty brought by constraint $TSC \leq \epsilon$ is that the Schrage property of the feasible solution used to define the binary branching scheme in Carlier's algorithm no more holds in general. We propose the following workaround. A critical block J^* is computed as in Section 2.2. If the block verifies the Schrage solution property (for all $k \in [1, m-1]$ $q_{i_{k+1}} \leq q_{i_k}$ or $r_{i_{k+1}} > r_{i_k}$), the branching scheme generating two nodes described by updates (10) and (11) is used. Otherwise, to improve the feasible solution, it is necessary to move one job of J^* (not only i_c) before or after all other jobs of J^* which yields a n-ary branching. For each candidate node $i \prec J^* \setminus \{i\}$ or $i \succ J^* \setminus \{i\}$, we check whether this branching is feasible w.r.t. the existing precedence constraints and the authorized number of color switches.

In all cases, a node consists in a decision $i \prec X$ or $X \prec i$ where X is a set of jobs. Contrarily to Carlier's algorithm in which these constraints are represented only by ready times or due dates updates in relation with the behavior of Schrage's algorithm, we propose here to store explicitely the associated precedence constraints, which will be useful for node evaluation and improvement mechanisms, described below.

Node evaluation. As the precedence constraints set by branching are explicitely stored, the node feasibility w.r.t. $TSC \leq \epsilon$ is checked directely by Darte's algorithm which provides the exact value of the number of switches reachable by the node. The node is fathomed if this value exceeds ϵ. For the maximum lateness criterion, a lower bound can still be obtained by computing the Jackson's preemtive schedule of value LB. The node is then fathomed if LB is not lower than the best feasible solution value. Classical resource constraint propagation methods are used to reduce job time windows and generate additional precedence constraints.

Feasible solution update at each node. Since Darte's algorithm is exact for the number of color switches objective, we take advantage of its use to compute the feasible solution associated with the node for upper bounding and branching purposes, except for the root node where the modified schrage method is used. The solution computed by the standard Darte's algorithm may have a high C_{qmax}. We propose the following enhancements. We compute the critical block J^* associated with Darte's solution and we apply a local search principle until no improvement of the C_{qmax} can be found without exceeding ϵ color switches. The considered moves in the local search scheme are based on the same necessary improvement conditions as the branching

scheme. A move thus consists in setting $i \prec J^* \setminus \{i\}$ or $i \succ J^* \setminus \{i\}$ constraints until the stopping condition is met.

Non-dominated solutions storing. Solutions found during the branch-and-bound algorithm are stored into an archive along with the best found solutions. That way, they can be used to provide an initial upperbound instead of the solution returned by the construction heuristic. If the problem $1|r_i, SC_{si,b} = 1|\epsilon(L_{\max}|TSC)$ under consideration is such that there is a solution with no more than ϵ switches in the archive, this solution is used as the initial upperbound, otherwise the usual heuristic is used.

4 Computational results

The algorithm has been tested on instances generated according to [2] with random colors. The CPU time was limited to five minutes, therefore on some instances, we do not obtain the optimal Pareto set but an approximation. Table 1 reports for each tested instance (Inst.) the size of the instance (n), the number of created nodes in the search tree (#NC), the number of fathomed nodes in the search tree (#ND), the time in seconds (Seconds), the number of optimal Pareto solutions found (#Pareto), and the last epsilon value reached before hiting the time limit.

5 Conclusions

The results show our method is able to generate the exact Pareto set for $n = 20$ and is able to generate a significant number of compromise solutions for $n \geq 30$. This is to our knowledge the first exact method proposed so far for this bi-objective problem. Further work will consist in extending the method to an arbitrary number of colors and setup costs. Even with two families, a critical issued is to propose a lower bound for the maximum lateness taking account on the upper bound on the number of switches.

Acknowledgements

The research of the third author was partially sponsored by the project ANR-GUEPARD

Table 1
Computation results.

#instance	n	#NC	#ND	Seconds	#Pareto	Last ϵ
110	20	37	8	0.14	7	ended
117	20	251	234	0.97	8	ended
119	20	240	232	1.12	5	ended
171	20	14108	14108	77.95	13	ended
221	20	506	430	1.93	6	ended
227	20	3405	3363	18.81	6	ended
24	20	21	13	0.04	7	ended
264	20	638	638	4.08	7	ended
30	20	46	9	0.04	4	end
53	20	18895	18667	90.63	10	ended
89	20	122	104	0.40	5	ended
117	30	14789	14626	127.34	9	ended
119	30	15322	13757	306.22	5	$\varepsilon = 7$
171	30	24021	21107	301.29	15	$\varepsilon = 3$
221	30	40355	33582	393.73	14	$\varepsilon = 2$
227	30	63268	53987	975.64	10	$\varepsilon = 5$
24	30	484	53	1.59	7	ended
264	30	19365	16196	330.12	11	$\varepsilon = 5$
30	30	13640	4318	326.37	3	$\varepsilon = 2$
53	30	12117	8408	115.92	3	$\varepsilon = 2$
89	30	27302	23240	146.86	9	$\varepsilon = 6$
117	50	7258	3985	126.86	10	$\varepsilon = 8$
119	50	2961	917	48.39	1	$\varepsilon = 15$
171	50	18464	2647	115.71	4	$\varepsilon = 6$
221	50	23221	14182	71.49	19	$\varepsilon = 2$
227	50	3030	1404	72.46	18	$\varepsilon = 6$
24	50	7627	557	300.07	2	$\varepsilon = 23$
264	50	12115	8685	674.499	9	$\varepsilon = 11$
30	50	5	3	0.09	12	$\varepsilon = 5$
53	50	9716	598	129.54	10	$\varepsilon = 9$
89	50	15861	2239	309.00	2	$\varepsilon = 13$

References

[1] A. Allahverdi, C.T. Ng, T.C.E. Cheng and M. Kovalyov. A Survey of scheduling problems with setup times or costs. *European journal of Operational Research*, 187:985–1032, 2008.

[2] J. Carlier. The one-machine sequencing problem. *European Journal of Operational Research*, 11(1):42–47, 1982.

[3] A. Darte. On the complexity of loop fusion. *Parallel Computing*, 26(9):1175–1193, 2000.

[4] H. Hoogeveen. Multicriteria scheduling. *European journal of Operational Research*, 167:592–623, 2005.