A bi-objective model for the single-machine scheduling problem with rejection cost and total tardiness minimization

Roberto Cordone, Pierre Hosteins

A bi-objective model for the single-machine scheduling problem with rejection cost and total tardiness minimization

Roberto Cordone
Dipartimento di Informatica
Università degli Studi di Milano, Italy

Pierre Hosteins
Univ Lille Nord de France, IFSTTAR, COSYS, ESTAS
F-59650 Villeneuve d'Ascq, France

Abstract

We study the problem of scheduling jobs on a single machine with a rejection possibility, concurrently minimizing the total tardiness of the scheduled jobs and the total cost of the rejected ones. The model we consider is fully bi-objective, i.e. its aim is to enumerate the Pareto front. We tackle the problem both with and without the presence of hard deadlines. For the case without deadlines, we provide a pseudo-polynomial time algorithm, based on the dynamic program of [25], thereby proving that the problem is weakly NP-hard. For the case with deadlines, we propose a branch-and-bound algorithm and prove its efficiency by comparing it to an ε-constrained approach on benchmark instances based on those proposed in the literature on similar problems.

Keywords: scheduling with rejection, total tardiness, bi-objective optimization, dynamic programming, branch-and-bound.

1 Introduction

Since the last fifteen years we have witnessed a trend in the area of scheduling with rejection, where jobs can be discarded at a specific cost if their scheduling is deemed too expensive. A recent survey of the results obtained in the area of scheduling with rejection [22] notes numerous applications, including make-to-order production systems with limited production capacity and tight delivery requirements, as well as scheduling with an outsourcing option. Oversubscribed scheduling problems also arise when machines represent highly valuable resources to be exploited at the maximum extent, as is the case with Earth observing satellites [2].

In this paper, we consider in particular the problem of scheduling jobs on a single machine so as to minimize at the same time their total tardiness with respect to the due dates and the total cost of the rejected jobs. The two objectives are clearly conflicting, and our aim is to enumerate the solutions of the Pareto front. Hard deadlines can also be imposed on some or all jobs. Steiner and Zhang [25] have proved that the problem of minimizing the sum of the total tardiness and the rejection cost is equivalent to a problem in which due dates are given,

but can be either delayed paying a cost or violated paying another cost. This situation can be of importance for international companies with large and costly orders, such as Airbus and Boeing in the aerospatial industry, or the suppliers of large companies, such as Wal-Mart in the retail sector. The introduction of deadlines in this framework would be a sensible extension, corresponding to a limitation on the range of the feasible delayed due dates.

The scheduling problems with rejection belong to the general category of bi-criteria scheduling problems, where two objectives F_1 and F_2 must be optimized contemporarily. F_1 usually represents the cost of the schedule, while F_2 represents the total rejection cost RC. According to [22], these problems can be roughly categorized into four different variations: minimize the sum of the objectives $F_1 + F_2$ (P1), minimize F_1 with a constraint on F_2 (P2), minimize F_2 with a constraint on F_1 (P3) and recover the full Pareto front (P4). We recall that a solution to a multi-criteria minimization problem is said to be *dominated* in the Pareto sense if another solution exists that is strictly better with respect to all criteria. Otherwise, the solution is *weakly non-dominated*. In particular, it is *strictly non-dominated* when no other solution exists that has lower or equal values for all criteria and a strictly smaller value for at least one criterion. The *Pareto front* targeted by variation P4 is the set of all strictly non-dominated solutions. As none of these solutions can be considered superior to the others, the final choice among them is a question of trade-off between the optimization criteria. This last approach is therefore fully bi-objective, as opposed to the other three.

Let us observe that several approaches exist to recover relevant solutions for the Decision Maker (DM). Some of these methods try to set the priorities of the DM before solving the problem, for example reducing it to the optimization of a single objective. In this sense, other single-criterion approaches can be defined besides variations P1, P2 and P3 (e.g., a weighted sum or a hierarchical lexicographic combination of the two objectives). As well, other approaches involve the DM in choices that guide the resolution process towards interesting compromises [16]. The full enumeration of the Pareto front (P4), for example, preludes to a post-processing phase which will select a subset of relevant solutions, or a single one, with a further intervention of the DM [10].

Throughout the paper we will use the three-field notation of [9] with the denominations and conventions of [22]. The general bi-criteria problem is written as $\alpha/rej/F1, F_2$, while the specific variations listed above are written as $\alpha/rej/F_1+F_2$ (P1), $\alpha/rej/\varepsilon(F_1/F_2)$ (P2), $\alpha/rej/\varepsilon(F_2/F_1)$ (P3), and $\alpha/rej/\#(F_1, F_2)$ (P4). In this work we investigate the $1/rej/\#(\sum_j T_j, RC)$ and $1/rej, \bar{d}_j/\#(\sum_j T_j, RC)$ problems, respectively. For the former problem, we introduce a pseudo-polynomial algorithm based on Dynamic Programming (DP), which runs in $\mathcal{O}(n^4P^3E^2)$ time, where $P = \sum_{j\in J} p_j$ is the total processing time and $E = \sum_{j\in J} e_j$ is the total rejection cost of the jobs. This clarifies the complexity status of the problem, which was previously described as open in [22]. A similar algorithm, with the same complexity, solves the case with *compatible* deadlines and processing times. Concerning the problem with general deadlines, we propose a branch-and-bound (B&B) algorithm, based on the computation of lower bounds on the two objectives, in order to build a Lower Bound (LB) set, combined with the computation of heuristic solutions, in order to build an Upper Bound (UB) set. This algorithm is an evolution of the B&B proposed in [5] for the single-objective variation P1 to the full bi-objective enumeration problem P4. Besides introducing an effective management of the LB and UB sets, a specialized branching rule and tailored visit strategies, the new algorithm takes into account the fact that some useful properties enjoyed by P1 cannot be exploited when solving P4. For example, all jobs whose tardiness is necessarily larger than the rejection cost can be safely discarded when minimizing

the sum of the two objectives, but not when enumerating the non-dominated solutions. We compare the results of the B&B with those of an ε-constrained approach using a Mixed Integer Linear Problem (MILP) commercial solver.

In the following we first provide a brief review of the relevant literature (Section 2). We then formally define a MILP model of the problem (Section 3), we provide a pseudo-polynomial DP algorithm for the problem without deadlines or with compatible deadlines and processing times (Section 4), we describe the B&B algorithm (Section 5) and we compare its results with those obtained with the ε-constrained approach (Section 6).

2 Brief literature review

The majority of the papers on scheduling with rejection deals with variation P1, with different scheduling criteria F_1. Very few papers actually deal with the full bi-objective P4 variation, even though scheduling with rejection is intrinsically a bi-objective problem. Examples of such works are [23], which describes a pseudo-polynomial algorithm for $1/rej/\#(\sum_j C_j, RC)$, or [11, 17, 18] which provide metaheuristic algorithms such as Simulated Annealing and Genetic Algorithms for the weighted problem $1/rej/\#(\sum_j w_j C_j, RC)$. A B&B algorithm developed in [27] solves a two machine flowshop problem with due date assignment, denoted as $F2/d_i = d, unknown\ d/\varepsilon(d/\overline{\mathcal{U}})$, which is equivalent to the $F2/rej/\varepsilon(C_{\max}/RC)$ rejection problem. The $F2/rej/\#(C_{\max}, RC)$ problem is also tackled heuristically in [24]. The work of [28] studies a problem in which the processing times can be controlled by allocating extra resources or a special rate-modifying activity; the scheduling criterion F_1 is the sum of the total completion time $\sum_j C_j$ and the resource consumption cost. The model is either solved polynomially in its P1 single-objective version or by solving a series of mathematical programs for the bi-objective P4 version. A certain number of pseudo-polynomial DP algorithms for some P2 problems exist in the literature (see e.g. [29]). While these are easily adapted to the P4 variation, there is almost no enumeration algorithm for the P4 variation of the hardest single-machine problems.

Focusing on total tardiness as the objective competing with the total rejection cost, Steiner and Zhang [25] prove the weak NP-hardness of $1/rej/\sum_j T_j + RC$. They provide a generalization of the DP algorithm of Lawler [14] which runs in $\mathcal{O}(n^4P^3)$ time. The complexity status of $1/\bar{d}_j/\sum_j T_j$, on the contrary, is partially open: the problem is NP-hard, but it is not known whether weakly or strongly. In particular, no pseudo-polynomial algorithm has been found. Consequently, also the status of $1/rej, \bar{d}_j/\sum_j T_j + RC$ is unclear. A B&B algorithm for this problem has been proposed in [5]. Other B&B algorithms for the weighted version of the problem without deadlines were proposed in [19], while the extension of that problem to sequence-dependent setup times was tackled with the help of heuristic approaches in [3] and [20].

3 An ε-constrained approach

Let J be a set of jobs of cardinality $n = |J|$. Some data are associated with each job $j \in J$, namely a processing time p_j, a due date d_j, a deadline $\bar{d}_j$ and a rejection cost e_j, which are all integer parameters. As already stated, $P = \sum_{j\in J} p_j$ and $E = \sum_{j\in J} e_j$ denote the total processing time and the total rejection cost of the jobs. The problem requires to select a subset of jobs $A \subseteq J$ for being processed on a machine, and to reject the complementary subset of jobs $R = J \setminus A$. The selected jobs must be scheduled on the machine respecting their deadlines. The

tardiness T_j of each job j is the difference between the completion time of the job and its due date, if the former exceeds the latter; otherwise, it is zero. The objective is to find all strongly non-dominated solutions with respect to the Total Tardiness (TT) of the jobs in A and the total Rejection Cost (RC) of the jobs in R.

Hereafter we present a *MILP* formulation with positional variables. Other models have been proposed and experimented for the problem at hand, such as a time-indexed formulation, see e.g. [5]. However, as highlighted by the numerical results, the alternative model was too slow to provide useful results for the bi-objective formulation considered here. The model uses the following variables:

- $x_{jh} \in \{0,1\}$ is equal to 1 if and only if job $j \in J$ is in position $h \in \{1,...,n\}$ in the schedule;
- $y_h \in \{0,1\}$ is equal to 1 if and only if the schedule includes a job in position $h \in \{1,...,n\}$;
- C_h is the completion time of the job in position $h \in \{1,...,n\}$, if any;
- $T_h \geq 0$ is the tardiness of the job in position $h \in \{1,...,n\}$, if any.

The MILP model is as follows.

$$\min \sum_{h=1}^{n} T_h \tag{1a}$$

$$\min \sum_{j \in J} e_j \left(1 - \sum_{h=1}^{n} x_{jh}\right) \tag{1b}$$

$$y_h = \sum_{j \in J} x_{jh} \qquad h \in \{1,...,n\}, \tag{1c}$$

$$\sum_{h=1}^{n} x_{jh} \leq 1 \qquad j \in J, \tag{1d}$$

$$y_h \geq y_{h+1} \qquad h \in \{1,...,n-1\}, \tag{1e}$$

$$C_h \geq \sum_{h'=1}^{h} \sum_{j \in J} p_j x_{jh'} - M(1 - y_h) \qquad h \in \{1,...,n\}, \tag{1f}$$

$$T_h \geq C_h - \sum_{j \in J} d_j x_{jh} \qquad h \in \{1,...,n\}, \tag{1g}$$

$$T_h \geq 0 \qquad h \in \{1,...,n\}, \tag{1h}$$

$$T_h \leq \sum_{j \in J} (\bar{d}_j - d_j) x_{jh} \qquad h \in \{1,...,n\}. \tag{1i}$$

Objectives (1a) and (1b) express the TT and the RC, respectively, as functions of the decision variables. Eqs. (1c) relate the x and y variables. Eqs. (1d) ensure that each job is processed at most once, while Eqs. (1e) force the occupied positions in the schedule to be contiguous. Eqs. (1f) define the completion time of the job in position h as the sum of the processing times of all jobs before it; note that it is necessary to introduce a big-M term to allow for a zero completion time for positions where no job is processed. Parameter M needs to be larger than the largest deadline $\max_{j \in J}\{\bar{d}_j\}$. Eqs. (1g), (1h) and (1i) define the tardiness variables and limit their values according to the deadlines.

In order to solve the problem with a MILP solver, we will adopt an ε-constrained approach. This consists in reducing the fully bi-objective P4 variation to the single-objective P3 variation by replacing objective (1a) with constraint $\sum_{h\in\{1,\dots,n\}} T_h \leq \varepsilon$, where ε is a parameter. Then, the parametric problem can be repeatedly solved for all possible values of ε from $\sum_{j\in J}(\bar{d}_j - d_j)$ down to zero, thus tracing the Pareto front of the original problem. However, the resolution process can be greatly sped up in the following way: the threshold ε is initialized by the total tardiness of a feasible solution with the minimum rejection cost, and it is updated, every time the parametric problem is solved, setting it just below the value of the total tardiness computed at the previous step. This avoids finding repeated solutions, but still enumerates the whole Pareto front. Notice that an alternative ε-constrained approach could replace the RC objective with a constraint and consider all possible values of threshold ε from E down to zero. Depending on the specific instance, it could be more efficient to consider one approach or the other: in our benchmark problems, we found the first to be more efficient.

4 Pseudo-polynomial algorithms for constrained deadlines

In this section we will provide pseudo-polynomial algorithms to solve two special cases of $1/rej, \bar{d}_j/\#(\sum_j T_j, RC)$. The first subsection considers the case in which all deadlines are large: $\bar{d}_j \geq P$ for all $j \in J$. This is equivalent to the problem without deadlines, $1/rej/\#(\sum_j T_j, RC)$. The second subsection considers the case in which all deadlines are *compatible* with the corresponding processing times, that is for any two jobs $i, j \in J$ such that $p_i < p_j$, we have that $\bar{d}_i \leq \bar{d}_j$. This case is denoted as $1/rej, (p_j, \bar{d}_j)/\#(\sum_j T_j, RC)$. Combining the results of [25] and [26], we prove that also this case can be solved pseudo-polynomially. Since both cases generalize $1//\sum_j T_j$, which is NP-hard [6], they are also NP-hard in the weak sense.

4.1 The case without deadlines

A pseudo-polynomial time algorithm was provided in [25] for the P1 variation of the total tardiness minimization with rejection and no deadlines, $1/rej/\sum_j T_j + RC$. This DP algorithm generalizes the one given in [14] for $1//\sum_j T_j$ and runs in $\mathcal{O}\left(n^4P^3\right)$ time. We will now extend it to $1/rej/\varepsilon(\sum_j T_j/RC)$, where the tardiness is minimized with a constraint $e = \sum_{j\in R} e_j \leq \varepsilon$ on the total rejection cost, and to $1/rej/\#(\sum_j T_j, RC)$. In the following, we assume knowledge of the DP algorithm of Lawler [14] for $1//\sum_j T_j$. Its principal characteristic is based on the following powerful property. Suppose that the jobs are renumbered so that they are ordered in Early Due Date (EDD) sequence ($d_1 \leq \ldots \leq d_n$), and let k be the job with the largest processing time (ties are broken introducing an arbitrary order): there exists an optimal sequence in which job k is scheduled in the $(k+\delta)$-th position, for a suitable integer value of $\delta \in \{0\ldots, n-k\}$, preceded by jobs $\{1,\ldots,k-1,k+1,\ldots,k+\delta\}$. Intuitively, longer jobs tend to be scheduled later, possibly incurring some tardiness, in order to avoid pushing multiple other jobs into tardiness. Such a property effectively divides the set of jobs into two subsets separated by job k. Using it, we can define specially structured subsets $J(i,l,k) = \{j : i \leq j \leq l,\, p_j < p_k\}$, which collect the jobs whose due date is between those of i and l and whose processing time is smaller than that of k. By applying recursively Lawler's property, the algorithm creates smaller and smaller subsets $J(i,l,k)$, until the problem can be trivially solved. Once the minimum tardiness of the elementary subsets is known, we can work our way backwards to J and obtain its optimal

schedule. Notice that, in order to simplify the notation, the recursion equations assume the elements of each subset to be renumbered consecutively.

As in [25], we define $v(J(i,l,k),s,t,e)$ as the minimum tardiness cost of scheduling any subset of $J(i,l,k)$, starting at time s, ending at time t and with a total rejection cost equal to e. By definition, $t-s=\sum_{j\in A(i,l,k)} p_j$ and $e=\sum_{j\in R(i,l,k)} e_j$, where $A(i,l,k)$ and $R(i,l,k)$ are, respectively, the unknown subsets of scheduled and rejected jobs in $J(i,l,k)$. Dynamic programming allows to compute the optimal solution of this problem for each given starting time, ending time and total rejection cost. The crucial observation is that the scheduled jobs obey Lawler's property as in problem $1//\sum_j T_j$, so that a similar decomposition theorem applies. The scheduled subset $A(i,l,k)$ can be divided into two subsets, $A(i,k'+\delta,k)$ and $A(k'+\delta+1,l,k)$, separated by the job k' with largest processing time inside $J(i,l,k)$: δ is a suitable unknown integer with $0\le\delta\le m-h$, where m is the cardinality $|J(i,l,k)|$ and h is the position of k' in $J(i,l,k)$ when sorted in EDD order. Now, we denote as τ the unknown time at which the second subset starts, as $\tilde{v}(J(i,l,k),s,t,e,\delta)$ the minimum tardiness obtained placing job k' in position $k'+\delta$ and as $\bar{v}(J(i,l,k),s,t,e,k')$ the minimum tardiness obtained rejecting k'.

The following recursion equations are defined for all possible job sets $J(i,l,k)$, all pairs of scheduling times $s,t\in\{0,\dots,P\}$ and all rejection costs $e\in\{0,\dots,E\}$:

$$
\begin{aligned}
v(\{\emptyset\},s,t,0) &= 0 \text{ for all } s=t\in\{0,\dots,P\}, && \text{(2a)}\\
v(J(i,l,k),s,t,e) &= \min\{\bar{v}(J(i,l,k),s,t,e,k'), \min_{0\le\delta\le m-h}\{\tilde{v}(J(i,l,k),s,t,e,\delta)\}\}, && \text{(2b)}\\
\tilde{v}(J(i,l,k),s,t,e,\delta) &= \min_{s\le\tau\le t,\,0\le e'\le e}\big\{v(J(i,k'+\delta,k'),s,\tau-p_{k'},e')+\max\{\tau-d_{k'},0\}+\\
&\quad +v(J(k'+\delta+1,l,k'),\tau,t,e-e')\big\}, && \text{(2c)}\\
\bar{v}(J(i,l,k),s,t,e,k') &= v(J(i,l,k)\setminus\{k'\},s,t,e+e_{k'}), && \text{(2d)}
\end{aligned}
$$

Eqs. (2a) provide the basic conditions, which hold for empty subsets. Eqs. (2b) state that job k' is either rejected or scheduled in $k'+\delta$-th position and that the optimal solution is the best out of these possible cases. Eqs. (2c) apply Lawler's decomposition: they express the tardiness incurred scheduling k' in position $k'+\delta$ as the sum of the total tardiness of the previous jobs, the tardiness of job k' and the total tardiness of the following jobs for all possible ending times and rejection costs of the first subset, which implicitly determine the starting time and the rejection cost for the second subset. Finally, Eqs. (2d) relate the corresponding values of functions $\bar{v}$ and v when job k' is rejected. The algorithm aims to compute $v(\{1,\dots,n\},0,t,e)$ for all values of $e\in\{0,\dots,E\}$ and $t\in\{0,\dots,P\}$ and recursively splits the current set of jobs until it obtains empty subsets $J(i,l,k)=\emptyset$, for which the boundary conditions provide the optimal solution. Then, the equations can be used backward to obtain all values of the above defined functions, along with the corresponding subsets of scheduled jobs. The optimum of the problem is given by $v^*=\min_{0\le e\le\varepsilon,\,0\le t\le P}\{v(\{1,\dots,n\},0,t,e)\}$.

Property 4.1. *The $1/rej/\varepsilon(\sum_j T_j/RC)$ problem is weakly NP-hard and the dynamic program (2) provides its optimal solution in pseudo-polynomial time, $\mathcal{O}(n^4P^3E^2)$.*

Proof. There at most n^3 subsets $J(i,l,k)$ to consider, so that we need to evaluate at most $\mathcal{O}(n^3P^2E)$ values of function v. Each of them is obtained by discriminating between $\mathcal{O}(n)$ values of functions $\tilde{v}$, which are in turn obtained by minimizing over at most $\mathcal{O}(PE)$ function values, so that we need to solve $\mathcal{O}(n^4P^3E^2)$ equations. Finally, we need an additional $\mathcal{O}(PE)$

time to choose the smallest value of $v(\{1,\dots,n\},0,t,e)$ for $t \in \{0,\dots,P\}$ and $e \in \{0,\dots,\varepsilon\}$, and an additional $\mathcal{O}(n)$ time to retrieve the optimal solution. Therefore, the total time complexity of the algorithm is in $\mathcal{O}(n^4P^3E^2)$. For small values of the threshold (i. e., $\varepsilon < e_j$ for all $j \in J$), no job can be rejected and the problem reduces to $1//\sum_j T_j$ which is NP-hard, *ergo* $1/rej/\varepsilon(\sum_j T_j/RC)$ is weakly NP-hard. □

Corollary 4.1. *The bi-objective problem* $1/rej/\#(\sum_j T_j, RC)$ *is weakly NP-hard and can be solved in* $\mathcal{O}(n^4P^3E^2)$ *time.*

Proof. As underlined in [22], once it is proven that any of the P2, P3 or P4 variation of a scheduling problem with rejection is weakly NP-hard, it follows that also the other two variations are NP-hard. In fact, the DP algorithm computes the minimum tardiness for each possible value of the rejection cost $e \in \{0,\dots,E\}$. Instead of merely selecting the value of $e \leq \varepsilon$ which yields the smallest total tardiness, we scan all possible values $e \in \{0,\dots,E\}$ to obtain a set of points $(\sum_j T_j, e)$ in the objective space, from which we extract the Pareto front in a total time bounded again by $\mathcal{O}(n^4P^3E^2)$. □

4.2 The case with compatible deadlines and processing times

When the deadlines are compatible with the processing times, i.e. $\bar{d}_i \leq \bar{d}_j$ for any two jobs $i,j \in J$ such that $p_i < p_j$, the Early Deadline and Shortest Processing Time orders coincide. We observe that this case also contains the situation of a unique deadline $\bar{d}_j = \bar{d}$ for all $j \in J$.

Here Lawler's property applies to the jobs in A which are effectively scheduled and it has been proven in [26] that the $1/\bar{d}_j/\sum_j T_j$ problem can be decomposed similarly to the case without deadlines. The difference is that the deadlines make some states infeasible, so that the DP procedure can exclude them. We can then state the following property.

Property 4.2. *The bi-objective problem* $1/rej,\bar{d}_j,(p_j,\bar{d}_j)/\#(\sum_j T_j, RC)$ *is weakly NP-hard and pseudo-polynomially solvable in* $\mathcal{O}(n^4P^3E^2)$ *time.*

The same extension from a problem with no deadlines to a problem with deadlines compatible with the processing times holds for the DP algorithm proposed in [25]: once again, the main difference is that some states become infeasible, allowing the DP procedure to remove them. Therefore, problem $1/rej,(p_j,\bar{d}_j)/\sum_j T_j + RC$ can be solved with the same complexity as $1/rej/\sum_j T_j + RC$.

Property 4.3. *The single objective problem* $1/rej,(p_j,\bar{d}_j)/\sum_j T_j + RC$ *is weakly NP-hard and pseudo-polynomially solvable in* $\mathcal{O}(n^4P^3)$ *time.*

5 Branch-and-bound

In this section we tackle the general problem with hard deadlines, i.e. $1/rej,\bar{d}_j/\#(\sum_j T_j, RC)$. Since no pseudo-polynomial algorithm currently exists to solve this problem, we provide a B&B algorithm. This implicitly enumerates all solutions branching on the jobs: each node ν of the branching tree is characterized by a subset $A_\nu \subseteq J$ of scheduled jobs and a subset R_ν of rejected jobs; the branching mechanism selects a free job from $J \setminus (A_\nu \cup R_\nu)$ and generates an A-node by forcing the job in the schedule and an R-node by rejecting the job. Such a mechanism is

equivalent to introducing in Formulation (1) one auxiliary binary variable $f_j = \sum_{h=1}^{n} x_{jh}$ for each job $j \in J$ and fixing it either to 1 (A-node) or to 0 (R-node).

The modern approach to enumeration algorithms for multi-objective problems focuses on building and refining a *LB set* and an *UB set*, which generalize the classical lower and upper bounds used for single-objective problems (see [21] for a review). We adopt the definition of [7], in which at every point of the computation, each strictly non-dominated solution is coincident with or dominated by a point in the LB set. The UB set includes feasible solutions found during the computation, and the points of the UB set that are not dominated by the LB set are guaranteed to correspond to strictly non-dominated solutions. When the algorithm terminates, the two sets coincide with each other and provide the Pareto front.

5.1 Computation of the LB set

For each open and feasible node ν of the branching tree, the algorithm determines the minimum tardiness T_ν^* and the minimum rejection cost e_ν^* that are independently achievable in the node. These two values identify an *ideal Pareto point* (T_ν^*, e_ν^*). The LB set maintained by the B&B collects all these ideal points, together with all known feasible solutions that are not dominated by any other point in the LB set. Notice that it has been proposed to compute the LB set of multi-objective problems by generating convex combinations of the objectives and solving the resulting single-objective auxiliary problems. Such an approach would generate several elements of the LB set which in general would dominate the ideal point [21]. However, we consider ideal points because it is possible to compute them rather efficiently, whereas no algorithm is available to optimize combinations of the single objectives, or even to compute strong lower bounds on them.

5.1.1 Minimization of the total tardiness

In order to obtain the minimum total tardiness for node ν, we consider only the jobs in A_ν, ignoring all the other ones. For those jobs, we compute the minimum tardiness with the B&B algorithm of Tadei et al. [26]. The use of a B&B algorithm to process a single sub-problem is justified by its velocity. The algorithm applies a best-first strategy. At each branching node, it computes an earliest ending time a_j and a latest ending time b_j for each job to be scheduled (the former also allows to compute a lower bound on the total tardiness). Then, it makes intensive use of the following two properties to limit the possible orders of the jobs:

1. Let $i, j \in A$ such that $p_i < p_j$:
   (a) if $d_i \leq \max\{a_j, d_j\}$ and $b_i \leq \bar{d}_j$, then i precedes j;
   (b) if $d_i > \max\{a_j, d_j\}$, $\bar{d}_i \geq l_j$ and $d_i + p_i > b_j$, i follows j.
2. Suppose that the jobs are ordered by increasing due date value (Early Due Date order) and that job k has the largest processing time. If $\bar{d}_k \geq \sum_{i=1}^{n} p_i$, then the optimal schedule sets job k in some position $r = k, ..., n$, preceded by jobs $\{1, 2, ..., k-1, k+1, ..., r\}$ and followed by jobs $\{r+1, ..., n\}$.

These two rules are a generalization to the problem with deadlines of the rules proposed by Lawler [14] and Emmons [8] for the problem with no deadlines. The above properties allow to fix some jobs in the schedule so that the remaining jobs form blocks, and these blocks are further restricted by a suitable branching mechanism.

5.1.2 Minimization of the total rejection cost

Consider the following nested knapsack problem:

$$\max \sum_{j \in J} e_j f_j \tag{3a}$$

$$\sum_{j \in J : \bar{d}_j \leq \bar{d}_i} p_j f_j \leq \bar{d}_i \qquad i \in J \tag{3b}$$

$$f_j = 1 \qquad j \in A_\nu \tag{3c}$$

$$f_j = 0 \qquad j \in R_\nu \tag{3d}$$

$$f_j \in \{0, 1\} \qquad j \in J \tag{3e}$$

where $f_j = 1$ indicates that job j is processed and $f_j = 0$ that is is rejected. The optimum of this problem, subtracted from the sum of all rejection costs $E = \sum_{j \in J} e_j$, provides the minimum rejection cost achievable in node ν. Problem (3) can be seen as a relaxation of Formulation (1) where the TT objective (1a) has been removed. Therefore, all elements of the formulation besides the deadline constraints and the rejection costs can be neglected. The above problem can be solved by a DP algorithm much similar to the classical algorithm for the knapsack problem, with a time complexity in $\mathcal{O}(n \max_j \bar{d}_j)$ [5]. We first sort the jobs in Early Deadline (EDL) order and define the recursion function $m_{j,t}$ as the maximum recoverable rejection cost from selecting jobs $i \leq j$, such that the total processing time of the selected jobs is equal to t. For a given job $j \in J$ and a given time $t \leq \min\{\bar{d}_j, \bar{d}_{j-1} + p_j\}$, we have the following recursion relations:

$$m_{0,t} = 0, \tag{4a}$$

$$m_{j,0} = 0, \tag{4b}$$

$$m_{j,t} = m_{j-1,t}, \qquad j \in R_\nu \tag{4c}$$

$$m_{j,t} = m_{j-1,t}, \qquad j \in J \setminus (A_\nu \cup R_\nu) \text{ and } t < p_j \text{ or } t > \bar{d}_{j-1} \tag{4d}$$

$$m_{j,t} = m_{j-1,t-p_j} + e_j, \qquad j \in A_\nu \text{ and } p_j \leq t \leq \bar{d}_{j-1}, \tag{4e}$$

$$m_{j,t} = m_{j-1,t-p_j} + e_j, \qquad j \in J \setminus (A_\nu \cup R_\nu) \text{ and } t \geq \max(\bar{d}_{j-1}, p_j), \tag{4f}$$

$$m_{j,t} = \max\{m_{j-1,t}, m_{j-1,t-p_j} + e_j\}, \qquad j \in J \setminus (A_\nu \cup R_\nu) \text{ and } p_j \leq t \leq \bar{d}_{j-1}, \tag{4g}$$

where Eq. (4a) and (4b) set the basic conditions, Eqs. (4c) and (4d) define the recursion relations for the rejected or infeasible jobs, Eqs. (4e) and (4f) do the same for the mandatory jobs and Eq. (4g) considers the free jobs. Processing the jobs by increasing deadlines ensures that the jobs selected up to job j can be feasibly scheduled as such, so that we do consider all the relevant feasible solutions in the solution space.

We slightly improve the running time of the algorithm by using the following trick. It is not actually necessary to compute optimal values of $m_{j,t}$ for all time steps. Indeed, since the jobs are processed in a "no-wait" fashion, the only relevant values of t are those which can be obtained as sums of processing times. For example, the only relevant values for the first job are $m_{1,0}$ and m_{1,p_1}. Consequently, for any job j, we only need to consider the set of values $\{m_{j-1,t}\}$ which were relevant for the previous job and the additional set of values $\{m_{j-1,t+p_j}\}$ obtained summing p_j to each relevant value of the previous job. This experimentally reduces the running time of the algorithm by 10 to 15%.

5.2 Pruning, branching and visit strategy

A branching node ν is pruned when it proves infeasible or when its ideal point (T^*_ν, e^*_ν) is dominated by a known feasible solution (for example, when it corresponds itself to a feasible solution). If a node is pruned by dominance, its ideal point is removed from the LB set, unless it corresponds to a feasible solution (in this case, in fact, it belongs to the Pareto front and must be saved). If a branching node ν is not pruned, it must be branched. To do that, we select from the optimal solution of the RC minimization sub-problem the free jobs $J \setminus (A_\nu \cup R_\nu)$. These jobs necessarily do not belong to the optimal solution of the TT minimization sub-problem, which only considers the jobs forced in the scheduling. Among the selected jobs, we choose j^* as the job with the largest e_j/p_j ratio. Then, we generate two nodes by moving j^* either to A_ν or to R_ν.

Both nodes generated from ν must be processed to update their LB and UB sets. Notice that the A-node has a RC sub-problem equivalent to that of ν. Therefore, the A-node simply inherits the value of e^*_ν. The new branching constraint, by forcing a new job in the schedule, might increase the minimum tardiness value, which must therefore be recomputed. Hopefully, this should refine the LB set. On the other hand, the R-node has a TT sub-problem equivalent to that of its father node. Therefore, the R-node inherits the value of T^*_ν. The new branching constraint, by rejecting a new job, might increase the minimum rejection cost value, which must therefore be recomputed, hopefully refining the LB set.

The branch-and-bound tree is explored with several different strategies. Besides the classical depth-first and breadth-first, we investigate three strategies inspired by the best-first strategy for single-objective problems: a lexicographic one extracting the node with the lowest tardiness T^*_ν and, in case of ties, the lowest rejection cost e^*_ν; a lexicographic one extracting the node with the lowest rejection cost and, in case of ties, the lowest tardiness; finally, a strategy which extracts the node with the lowest product $(T^*_\nu - T^*_0)\,(e^*_\nu - e^*_0)$, where T^*_0 and e^*_0 are the minimum tardiness and rejection costs at the root node. The rationale of this criterion is to replace the single-objective lower bound used in the classical best-first strategy with a combination of lower bounds for the two objectives. Notice that, if no job is mandatory, the minimum tardiness at the root node is $T^*_0 = 0$. The criterion admits a geometrical interpretation, as it corresponds to the area of the rectangle identified by the ideal points of the root node and the current node.

5.3 Computation of the UB set

The UB set is populated by exploiting the following heuristic procedure at each branching node. First, the optimal solution of the TT sub-problem provides a feasible schedule for the jobs in A_ν. Then, the jobs from $J \setminus (R_\nu \cup A_\nu)$ are inserted in the current schedule, one by one, in non-increasing order of e_j/p_j (rejection cost to processing time ratio), so as to improve the rejection cost as much as possible while consuming the available working time as little as possible. Each new job is inserted in the position of the schedule where it implies the minimum increase in tardiness. After each job has been inserted, the current feasible solution is tested for inclusion in the UB set: if no known solution dominates it, it is included in the UB set, possibly removing previous solutions now dominated by it. This procedure aims to produce a sequence of effective compromises between the two objectives. A pseudo-code is shown in Algorithm 1.

Algorithm 1 *Heuristic*$(J, e, p, d, \bar{d}, A, R)$

Compute the minimum tardiness solution x for A using the algorithm of Tadei et al. [26];
$J := J \setminus (R \cup A)$;
Sort J by non-increasing order of e_j/p_j;
while $J \neq \emptyset$ **do**
  $j^* := \arg\max_{j \in J} \frac{e_j}{p_j}$;
  Compute the best feasible position p^* for j^* in x with respect to the total tardiness;
  if p^* exists **then**
    Insert j^* in position p^* in x;
    Update the UB set with x;
  end if
  $J := J \setminus \{j^*\}$;
end while
Return x;

6 Computational experiments

In this section we report on the outcome of computational tests concerning the ε-constrained approach illustrated in Section 3 and the B&B bound algorithm described in Section 5. The computations were performed on an eight-thread Intel(R) Core(TM) i7-3770 CPU with 3.40 GHz and 8 GB RAM. The MILP formulations were solved with the C++ library of `CPLEX 12.63`, while the B&B algorithm was coded in C++ and compiled with `g++ 4.7.2`.

6.1 Instances

In order to build a benchmark, we considered the instances of the total weighted tardiness minimization problem from the OR library [1]. These are divided into three classes with a total number of 40, 50 and 100 jobs, respectively. They feature important variations of the distribution of the due dates along the time horizon. In detail, the processing times are integers uniformly generated from the interval $[1, 100]$. The due dates are then generated with uniform probability inside interval $[(1 - TF - RDD/2)P, (1 - TF + RDD/2)P]$, where parameters TF and RDD take 5 possible values: 0.2, 0.4, 0.6, 0.8, and 1.0. To avoid negative due dates, that would force a minimum tardiness, these are modified so that $d_i = p_i$. Each possible pair of parameter values yields 5 instances, so that the overall benchmark consists of $3 \cdot 5 \cdot 5 \cdot 5 = 375$ instances. The tardiness factor TF controls the average value of the due dates: as TF increases, they tend to become tighter. The due date range RDD controls the width of the interval from which the due dates are extracted: as RDD increases, they tend to spread on a wider range. In order to convert the OR library instances into instances of the problem at hand, we have removed the job weights and generated random deadlines such that the value of $\bar{d}_j - d_j$ for each job j is between 1 and 20% of the average due dates for the instance, with a uniform probability distribution. Finally, we have generated rejection costs with a uniform probability distribution in the same interval, in order to make them comparable to the potential tardiness.

A different scheduling problem with rejection and total tardiness has been addressed in [20]. We have therefore generated a second benchmark of instances along the lines followed in that work. The number of jobs is the same as in the first benchmark, i.e. n =40, 50 or 100. The

processing times and rejection costs are uniformly generated inside the range $[1, 20]$. The due dates are once again inside the interval $[(1 - TF - RDD/2)P, (1 - TF + RDD/2)P]$, with the smallest values raised up to p_i, but TF and RDD take five different values: 0.1, 0.3, 0.5, 0.7 and 0.9. The deadlines are fixed as $\bar{d}_j = d_j + RDDp_j$. We therefore generate 25 instances for each number of jobs, hence a total of 75 instances. All instances are available at the following URL address: `http://optlab.di.unimi.it/PCSMSP/`.

6.2 Effectiveness of different visit strategies

First, we compare the performance of the B&B algorithm with five different visit strategies: the classical depth-first and breadth-first stategies (respectively, *DFS* and *BFS*), a lexicographic strategy selecting the node with minimum tardiness and (secondarily) minimum rejection cost (*Lex-TT*), a lexicographic strategy selecting the node with minimum rejection cost and, secondarily, minimum tardiness (*Lex-RC*) and, finally, a strategy selecting the node ν with the smallest value of product $(T^*_\nu - T^*_0)\,(e^*_\nu - e^*_0)$, which we denote as *Smallest-Area-First Strategy* (*SAFS*), because the product corresponds to the area of the rectangle identified by the ideal points of node ν and the root node of the branching tree (see Figure 1, where the node selected is the one with ideal point coordinates $(100, 3\,700)$).

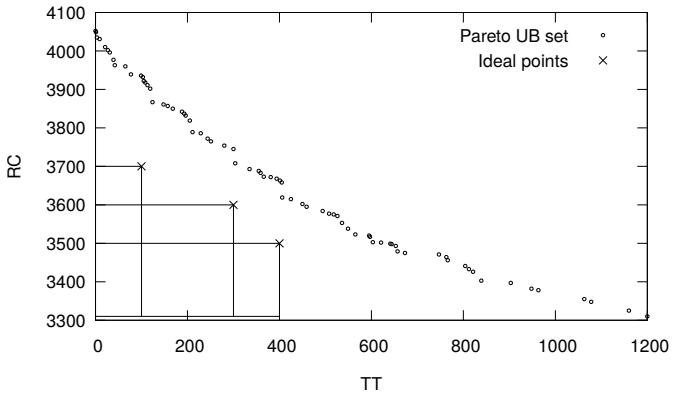

Figure 1: The *SAFS* visit strategy selects the node ν whose ideal point $(T^*_\nu, e^*_\nu) = (100, 3\,700)$ identifies, together with the ideal point of the root node, $(T^*_0, e^*_0) = (0, 3\,320)$, the rectangle of smallest area $(T^*_\nu - T^*_0)\,(e^*_\nu - e^*_0)$.

Table 1 reports the results of the comparison on the first benchmark: the first column provides the size of the three classes of instances and the experimental quantities considered, while the last five columns refer to the strategies applied. The first row displays the number of instances closed within a time limit of 1 000 seconds (out of the 125 instances of each size class). The second row shows the average computational time in seconds. The third row reports the average number of elements belonging to both the UB set and the LB set at termination (labelled as $|LB \cap UB|$): these are the provably non-dominated solutions provided in output. The fourth row contains the average number of elements of the UB set at termination (labelled as $|UB|$): these are the solutions provided in output. If the instance is closed, both the LB set and the UB set coincide with the full Pareto front; otherwise, their intersection is a subset

of the Pareto front, while the UB set is a superset of that subset (therefore a set of solutions with no clearly definite property, besides being feasible, but probably representative and useful for the DM). Estimating the quality of the UB set would require a quantitative measure of its distance from the LB set. To the best of our knowledge, there is no consensus on how to provide such a measure. We propose a possible measure, which we denote as *relative area* δ, and report its values in the fifth row of Table 1. The definition of the relative area is inspired by the classical relative gap used to compare the lower and the upper bound in a single-objective optimization problem. First, we find the ideal point of the problem (T_0^*, e_0^*). Then, we define a classic stair-like dominance region in the objective space for both the LB set and the UB set. This is defined by the points which are dominated by the ideal point of the root node (T_0^*, e_0^*), but not dominated by any point of the LB set or the UB set, respectively. In Figure 2, the two regions are identified by dashed and continuous lines, respectively. The areas of these regions are denoted A_{LB} and A_{UB}. By construction, we have that $A_{LB} \leq A_{UB}$. The relative area is defined as the ratio $\delta = (A_{UB} - A_{LB})/A_{UB}$. Let us observe that $0 \leq \delta \leq 1$, and that δ is zero when the LB set and the UB set coincide. Moreover, this quantity has the advantage of being independent from the unit of measure of the objective functions. Dividing the gap by A_{LB} would be of course also reasonable, but on a minority of instances in which the LB set reduces to the ideal point of the root node it would imply $A_{LB} = 0$, and therefore an infinite gap. This indicator is similar to the hypervolume indicator introduced in [30], which refers however to a set of points dominated by the UB set, and is therefore conceptually unrelated.

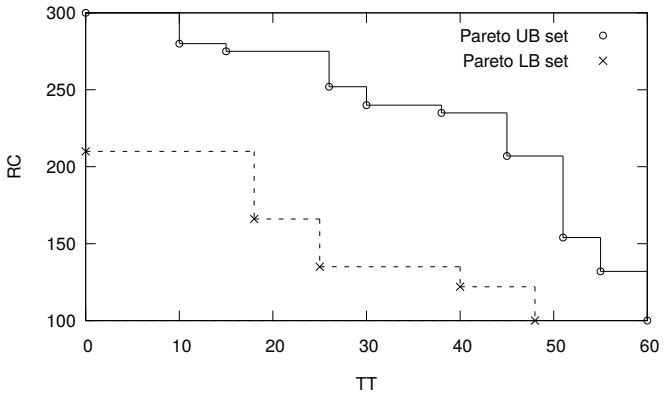

Figure 2: Definition of the dominance regions associated to the LB set (under the dashed curve) and the UB set (under the continuous curve); the ideal point of the root node is $(0, 100)$.

The best value in each row of Table 1 is bolded. The results clearly suggest that the classical *BFS* performs much worse than the other strategies, which solve nearly all instances with up to 50 jobs and many more instances with 100 jobs. They also require an average computational time one order of magnitude smaller, though for the larger instances the difference tends to fade out because all unsolved instances take $1,000$ seconds. When the algorithm is prematurely terminated, the number of solutions found is larger for *DFS* and *Lex-RC*. The performance of *Lex-RC* is particularly good with respect to the number of provably non-dominated solutions: for the instances with 100 jobs, it is more than twice as large as that of *DFS* and *SAFS*, and more than three times as large as that of *BFS* and *Lex-TT*. Overall, Table 1 indicates a

40 Jobs					
	DFS	*BFS*	*Lex-TT*	*Lex-RC*	*SAFS*
# closed instances	**125**	**125**	**125**	**125**	**125**
avg. time	1.04	45.59	**1.03**	**1.03**	1.04
avg. # non-dom. sol. ($\|LB \cap UB\|$)	**9.40**	**9.40**	**9.40**	**9.40**	**9.40**
avg. # sol. ($\|UB\|$)	**9.40**	**9.40**	**9.40**	**9.40**	**9.40**
avg. relative area (δ)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
50 Jobs					
	DFS	*BFS*	*Lex-TT*	*Lex-RC*	*SAFS*
# closed instances	**125**	109	123	123	123
avg. time	**25.05**	203.22	31.94	32.07	31.82
avg. # non-dom. sol. ($\|LB \cap UB\|$)	**13.42**	10.86	12.96	13.38	12.98
avg. # sol. ($\|UB\|$)	**13.42**	12.86	13.34	**13.42**	13.35
avg. relative area (δ)	**0.00**	0.12	0.02	<0.01	0.01
100 Jobs					
	DFS	*BFS*	*Lex-TT*	*Lex-RC*	*SAFS*
# closed instances	**61**	48	**61**	**61**	**61**
avg. time	565.90	657.06	563.96	564.44	**564.30**
avg. # non-dom. sol. ($\|LB \cap UB\|$	10.89	4.20	8.66	**25.21**	9.26
avg. # sol. ($\|UB\|$)	43.42	21.90	27.87	**44.47**	27.58
avg. relative area (δ)	0.44	0.60	0.51	**0.21**	0.47

Table 1: Numerical results for different branching strategies of the B&B algorithm over the instances of the OR Library with a timeout of 1 000 seconds.

predominance of the *Lex-RC* strategy (with the partial exception of the instances with 50 jobs, where its performance is however not far from the best: only two instances are incompletely solved and hit the time limit). Consequently, this visit strategy has been used in the following computational experiments.

6.3 Comparison between methods

We hereby compare the numerical results obtained by the B&B and the ε-constrained approach based on Formulation (1). For both approaches we first determine the minimum possible rejection cost with the DP algorithm described in Section 5.1.2. Then, we schedule the jobs thus selected minimizing their total tardiness with the B&B presented in Section 5.1.1. This provides the starting value of the threshold on the total tardiness used in the ε-constrained approach and a feasible solution to initialize the UB set of the B&B.

A fair comparison beetween the ε-constrained approach and the B&B is delicate since both algorithms provide the Pareto front in the end, but they proceed somewhat differently, and provide quite different outcomes when terminated prematurely. The former approach systematically accumulates a sequence of non-dominated points, gradually enlarging a specific region of the Pareto set. In fact, it repeatedly optimizes the RC with a threshold on the value of the TT, updating at each step the value of the threshold to forbid the solution found at the previous step. Each new solution thus generated is at least weakly non-dominated, since no feasible solution can be better for both objectives, but solutions with the same RC and a smaller TT could exist. However, as the constraint on the TT becomes tighter and tighter, every new solutions found

Nb. of jobs		40	50	100
ε-constrained ILP	# closed instances	61	40	23
	avg. time	561.42	707.74	820.61
	avg. # non-dom. sol. ($\|LB \cap UB\|$)	3.65	2.30	2.49
	avg. # sol. ($\|UB\|$)	5.81	5.75	8.06
	avg. rel. gap (δ)	0.43	0.64	0.69
B&B (*Lex-RC*)	# closed instances	125	123	61
	avg. time	1.02	32.10	564.37
	avg. # non-dom. sol. ($\|LB \cap UB\|$)	9.40	13.38	25.20
	avg. # sol. ($\|UB\|$)	9.40	13.42	99.92
	avg. rel. gap (δ)	0.00	<0.01	0.21

Table 2: Numerical results for the ε-constrained approach and the B&B algorithm on instances of the OR Library with a timeout of 1 000 seconds.

either has the same rejection cost as the previous one or a larger one. In the former case, the new solution dominates the previous one, and replaces it in the UB set; in the latter case, it proves that the previous solution was strongly non-dominated. In this way, all solutions found, except for the last one, are guaranteed to belong to the Pareto front. The B&B approach has a rather different behaviour, as it regularly updates an UB and a LB set, exploring more uniformly the criterion plane, but possibly missing the actual Pareto front until late phases of the search. In order to mitigate the difference in their behaviour, we have decided to feed both approaches with a starting UB set, generated by Algorithm 1 applied at the root node, and let them evolve for 1 000 seconds.

Table 2 displays the results on the first benchmark. The first column identifies the approach, the second one the experimental quantities reported. The last three columns refer to the three classes of instances, divided according to the number of jobs. The upper block of rows refers to the ε-constrained approach, the lower one to the B&B. The first row of each block reports the number of instances closed within the time limit, the second one the average computational time in seconds. The third row displays the average number of strongly non-dominated solutions provided in output ($|LB \cap UB|$), while the following one displays the average number of solutions generated ($|UB|$). The last row provides the relative area gap. All values apart from the number of closed instances are averaged over the 125 instances of the corresponding class.

Rather expectedly, both approaches close less instances and take larger computational times as the number of jobs to schedule increases. The B&B, however, closes at least twice as many instances in the allowed time, and much faster than the ε-constrained approach. The ratio of the running time between the two algoritms decreases with $|J|$ because the 1 000 seconds time limit is hit more often by both algorithms as the size increases. The number of solutions in the output set increases with $|J|$ for the B&B, whereas it decreases for the ε-constrained approach. This is a hint of the difficulty the latter encounters with larger instances, where the running time for each iteration of the ε-constrained method increases sharply, allowing fewer and fewer solutions to be recovered by the algorithm. It should be noted that the fraction of strictly non-dominated solutions found by the B&B decreases with $|J|$, but up to $|J| = 100$ the B&B still finds more strongly non-dominated solutions than the general feasible solutions found by the ε-constrained approach (which also include those provided by the starting heuristic). Further

light on the behaviour of the two algorithms can be shed by analysing in more detail the time at which the solutions in the output set were found. The main difference in this regard is that the ε-constrained approach first finds a few solutions very quickly, but then enters a region of the objective space where it becomes very time consuming to find new solutions solving the mathematical formulation. The MILP solver then remains stuck looking for a new solution for hundreds of seconds until it runs out of time. On the other hand, the B&B takes more and more time to find solutions when the number of jobs increases, but the ratio with respect to the total computational time decreases. All in all, the B&B is more effective in providing a set of solutions that span the objective space better, that are more often guaranteed to belong to the Pareto front and that furnish in a short amount of time interesting alternative options to the DM.

Nb. of jobs		40	50	100
ε-constrained ILP	# closed instances	21	18	14
	avg. time	174.12	311.20	456.20
	avg. # non-dom. sol. ($\|LB \cap UB\|$)	1.84	1.84	1.52
	avg. # sol. ($\|UB\|$)	2.56	3.20	2.44
	avg. rel. gap (δ)	0.16	0.28	0.37
B&B (*Lex-RC*)	# closed instances	25	25	25
	avg. time	0.04	0.12	7.48
	avg. # non-dom. sol. ($\|LB \cap UB\|$)	3.00	4.04	3.56
	avg. # sol. ($\|UB\|$)	3.00	4.04	3.56
	avg. rel. gap (δ)	0.00	0.00	0.00

Table 3: Numerical results for the ε-constrained approach and the B&B algorithm on instances generated as in [20] with a timeout of 1 000 seconds.

In Table 3, results similar to Table 2 are displayed for the second benchmark of instances, generated as in [20]. The superiority of the B&B over the solver-based ε-constraint method is once again obvious as it closes the instances three or four orders of magnitude faster. The instances, however, are easier to solve than those from the OR Library, as emphasized by the fact that all instances are closed by the B&B within the time limit. This might come from the fact that such instances have a much tighter range for the tardiness of each job, and therefore have fewer solutions. We can see for example that the Pareto fronts contain relatively few solutions, between 3 and 4.

6.4 Analysis of specific classes and instances

We now analyse in more detail the performance of the B&B on specific classes of instances, in order to characterize the features which make an instance harder or easier to solve. Table 4 reports the number of closed instances, the average computational time, the average number of strongly non-dominated solutions and the average relative area for the instances with 100 jobs of the first benchmark, divided according to the values of parameters RDD and TF. We remind that RDD controls the width of the range of due dates (larger values correspond to more widespread due dates), while TF controls their tightness (larger values correspond to tighter due dates). Each average value refers to the 5 instances with the same structure. The results

show that the instances tend to be much easier and quicker to solve when TF is small or large. Increasing RDD also tends to make the instances easier, though the effect is weaker. Focusing on the closed instances, it appears that the Pareto front tends to be smaller when RDD is larger and TF is smaller. These trends seems reasonable, since small values of TF correspond to large due dates (and deadlines), which allow to schedule most of the jobs, reducing the range of values for the rejection cost. On the other hand, large values of TF correspond to small due dates (and deadlines), so that only few jobs can be feasibly scheduled and the combinatorial possibilities for the selection of jobs are severely limited. Secondarily, when RDD increases, the jobs have more varied due dates and it is easier to find the best ordering among the jobs.

		# closed instances		avg. time		avg. $\lvert LB \cap UB \rvert$		avg. relative area	
RDD	TF	B&B	ILP	B&B	ILP	B&B	ILP	B&B	ILP
0.2	0.2	5	0	165.6	1000.0	27.6	4.6	0.00	0.75
	0.4	0	0	1000.0	1000.0	29.8	1.0	0.33	1.00
	0.6	0	0	1000.0	1000.0	23.8	1.0	0.47	1.00
	0.8	0	0	1000.0	1000.0	20.6	1.0	0.26	1.00
	1.0	5	1	0.4	896.2	10.2	2.6	0.00	0.79
0.4	0.2	5	4	11.6	214.4	3.2	2.4	0.00	0.03
	0.4	1	0	977.2	1000.0	44.2	1.0	0.10	1.00
	0.6	0	0	1000.0	1000.0	26.4	1.0	0.33	1.00
	0.8	0	0	1000.0	1000.0	27.8	1.0	0.16	1.00
	1.0	5	0	4.20	1000.0	19.4	1.0	0.00	1.00
0.6	0.2	5	5	10.8	0.0	1.0	1.0	0.00	0.00
	0.4	4	0	311.8	1000.0	38.0	1.0	0.00	1.00
	0.6	0	0	1000.0	1000.0	32.0	2.8	0.15	0.60
	0.8	0	0	1000.0	1000.0	36.6	4.0	0.54	0.80
	1.0	5	0	363.4	1000.0	52.4	1.8	0.00	0.97
0.8	0.2	5	5	10.6	0.0	1.0	1.0	0.00	0.00
	0.4	5	0	10.6	1000.0	5.0	1.4	0.00	0.71
	0.6	0	0	1000.0	1000.0	40.4	7.4	0.32	0.35
	0.8	0	0	1000.0	1000.0	47.2	6.4	0.75	0.76
	1.0	5	0	291.4	1000.0	34.0	1.8	0.00	0.97
1.0	0.2	5	5	10.8	0.0	1.0	1.0	0.00	0.00
	0.4	5	3	11.4	400.2	1.8	1.2	0.00	0.25
	0.6	0	0	1000.0	1000.0	32.4	8.6	0.57	0.36
	0.8	0	0	1000.0	1000.0	32.0	3.2	0.78	0.92
	1.0	1	0	929.4	1000.0	42.2	3.0	0.47	0.95

Table 4: Performance of the B&B and the ε-constrained approach on the instances of the OR Library with 100 jobs, with a timeout of 1 000 seconds, depending on the distribution of the due dates (RDD) and their tightness (TF).

In order to illustrate the distribution of the solutions in the criterion space, we show the LB sets and the UB sets returned by the B&B and the ε-constrained approach for a few benchmark instances. In Figure 3 we can see the two sets for an instance with 40 jobs (on top), an instance with 50 jobs (in the middle) and an instance with 100 jobs (at the bottom). The graphs on the left refer to the B&B, while the graphs on the right refer to the ε-constrained approach. The

elements of the LB set are represented as cross marks, the elements of the UB set as circles, the strongly non-dominated elements as circles including crosses. An important feature we can see from these plots is that the curve in criterion space is not convex. This reinforces the need to analyze the problem in a complete bi-objective manner. In fact, minimizing any weighted sum of the criteria would necessarily miss several non-dominated solutions that could be of relevance for the DM.

The B&B approach clearly provides a larger number of solutions and marks most of them as part of both sets, thus guaranteeing that they are actually part of the Pareto front. The ε-constrained approach focuses on the right part of the front, that is on the solutions with larger tardiness and smaller rejection cost. This is because the instance is not completely solved in the available time. Correspondingly, apart from the solutions correctly identified as strongly non-dominated on the right side of the graph, the LB set is limited to a single ideal point in the lower left corner. The instance with 100 jobs exhibits a visible separate LB set also for the B&B, as it was not closed within the time limit. The LB set tends to focus on the right part of the front, as for the ε-constrained approach. This is probably due to the use of the *Lex-RC* visit strategy, which favours the open nodes with small values of the rejection cost over the nodes with small total tardiness. Indeed, the typical behaviour of the B&B with this strategy is that the LB set tends to stay away from the UB set in the region with small tardiness until the last phases of the algorithm, when the LB set is quickly updated to include points with larger rejection costs, converging to the UB set and allowing the process to terminate. Also in these instances, however, the ε-constrained approach is much more biased, as apparent from the figure in the lower right corner, where the LB set is reduced to a single strongly non-dominated solution on the right and an ideal point on the left, quite far away from the UB set.

In order to further investigate the bias introduced by the visit strategy towards generating points of the LB set in different regions of the criterion space, Figure 4 compares the LB and UB sets returned by the B&B with the *Lex-RC* strategy on the left panel and the *SASF* strategy on the right panel for an instance with 100 jobs. As described earlier, we can see that the *Lex-RC* strategy tends to find the strongly non-dominated solutions with lowest RC first and the LB solutions on the left part of the diagram do not have sensibly higher TT cost than the solutions on the right. The *SASF* strategy aims to provide a more evenly distributed LB set, possibly closer on average to the UB set, with respect to the *Lex-RC* strategy. In practice, however, though its LB set contains points with slightly larger RC values, these are farther from the UB set and the latter consists of worse points, much fewer of which have been confirmed as strongly non-dominated. This visual illustration gives a hint as to why the average relative area gap is smaller for the *Lex-RC* strategy than for the *SAFS* strategy.

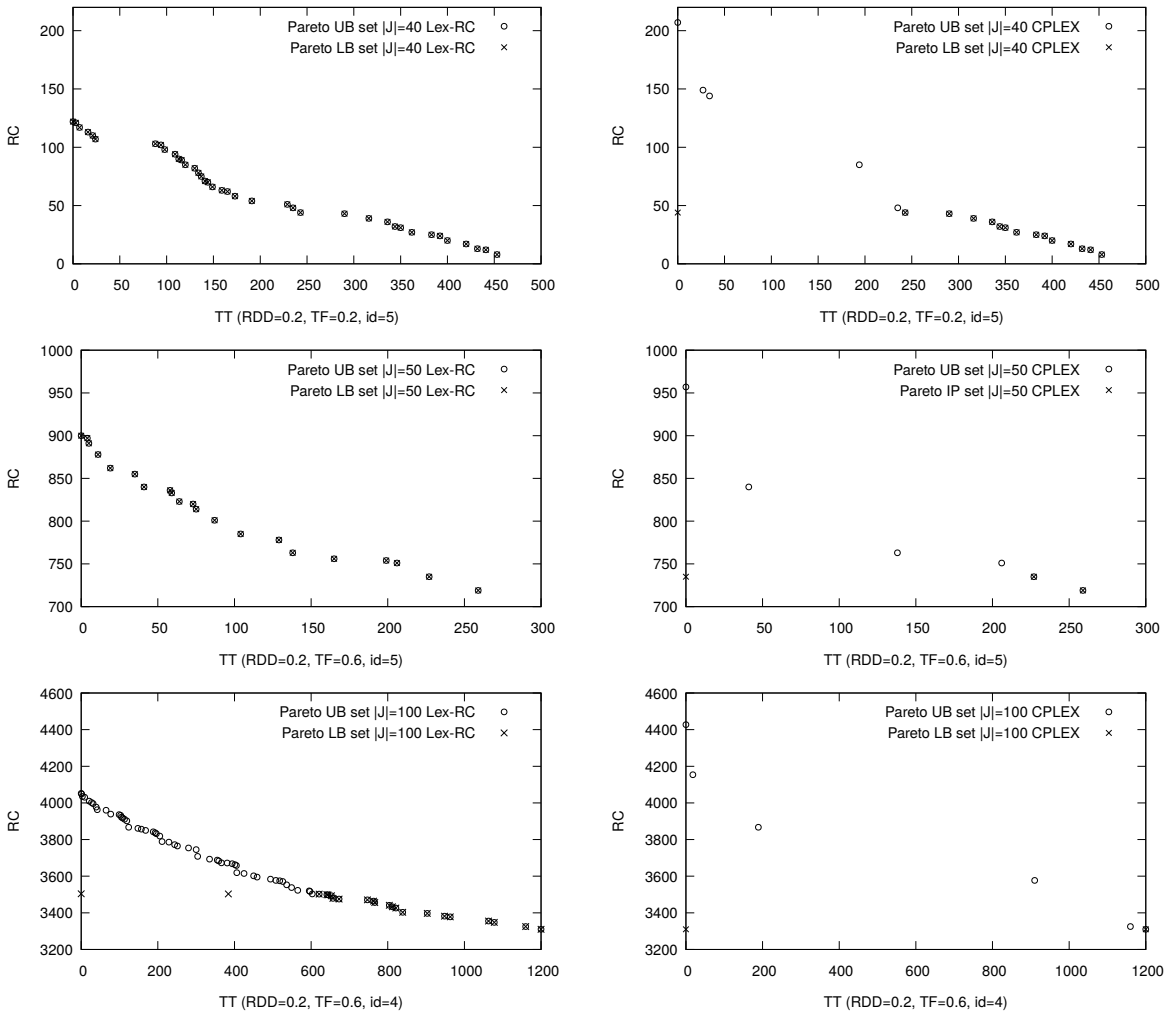

Figure 3: Pareto fronts from the B&B algorithm (on the left panels) and the ε-constrained approach (on the right panels) for three instances, respectively with 40 jobs (on top), 50 jobs (second row) and 100 jobs (last row). The total tardiness is reported on the x axis, the rejection cost on the y axis. The UB set of the B&B is represented by round points and the LB set by x-shaped points. The features of each instance are referenced below each figure.

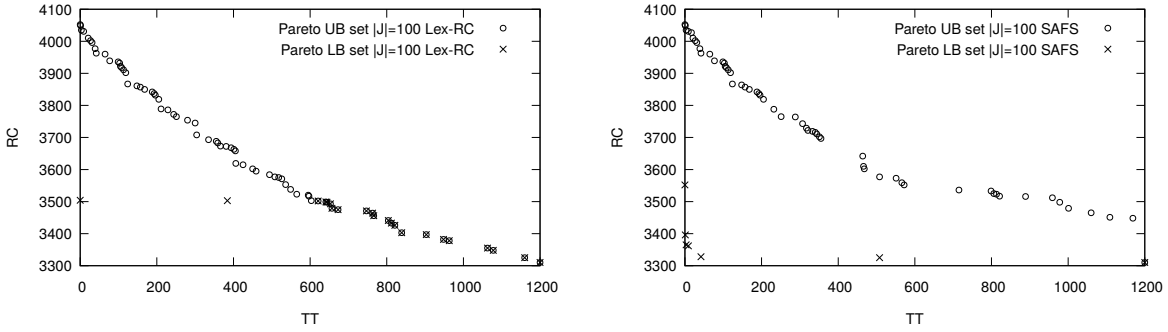

Figure 4: LB and UB sets returned by the B&B with the *Lex-RC* strategy (on the left panel) and with the *SASF* strategy (on the right panel) for the $RDD = 0.2$, $TF = 0.6$ and $id = 4$ instance with 100 jobs. The total tardiness is reported on the x axis, the rejection cost on the y axis. The UB set of the B&B is represented by round points and the LB set by x-shaped points.

7 Conclusions

We have analyzed a full bi-objective model for the single machine scheduling problem with rejection cost and total tardiness minimization, without or with deadlines, i.e. the $1/rej/\#(\sum_j T_j, RC)$ and $1/rej, \bar{d}_j/\#(\sum_j T_j, RC)$ problems. In the case with no deadlines we have provided a pseudo-polynomial algorithm to recover the set of strictly non-dominated solutions, thereby proving that the full bi-objective version of the problem is weakly NP-hard. The algorithm can be simply adapted to the case in which deadlines are given, but are compatible with the processing times of the jobs. When general hard deadlines are imposed on the jobs, we have proposed a B&B and evaluated its performance on two benchmarks of instances generated as in the literature on similar problems. Our experiments prove the B&B to be more efficient than a ε-constrained approach. The results tend to underline the importance to analyze scheduling problems with rejection in a full bi-objective approach, so as to obtain a reasonably complete approximation of the Pareto front, instead of tackling a simple linear combination of the two objectives. While relatively few approaches in the literature do that, we believe it to be an interesting avenue of research to design full bi-objective exact algorithms for scheduling problems with rejection, in order to obtain useful suggestions to the DM. This could be a viable approach even when the optimization of the single objectives is not easy, provided that sufficiently efficient methods are available to evaluate lower bounds on both the scheduling and the rejection costs.

References

[1] J.E. Beasley, OR-Library: distributing test problems by electronic mail, *Journal of the Operational Research Society* 41:11 (1990) 1069–1072.

[2] N. Bianchessi and G. Righini, Planning and scheduling algorithms for the COSMO-SkyMed constellation, *Aerospace Science and Technology* 12 (2008) 535–544.

[3] Z. Bilgintürk, C. Oğuz and S. Salman, Order acceptance and scheduling decisions in make-to-order systems, In *Proceedings of the 5th Multidisciplinary International Scheduling Conference: Theory and Applications*, Paris, France (2007).

[4] B. Chen, C.N. Potts and G.J. Woeginger, A Review of Machine Scheduling: Complexity, Algorithms and Approximability, In D.-Z. Du and P.M. Pardalos (Eds.), *Handbook of Combinatorial Optimization*, Kluwer Academic Publishers (1998).

[5] R. Cordone, P. Hosteins and G. Righini, A branch-and-bound algorithm for the prize-collecting single-machine scheduling problem with deadlines and total tardiness minimization, *INFORMS Journal on Computing,* 30:1 (2018) 168–180.

[6] J. Du, and J. Y. T. Leung, Minimizing total tardiness on one machine is NP-hard, *Mathematics of Operations Research* 15 (1990) 483–495.

[7] M. Ehrgott and X. Gandibleux, Bound sets for biobjective combinatorial optimization problems, *Computers and Operations Research* 34:9 (2007) 2674–2694.

[8] H. Emmons, One-machine sequencing to minimize certain functions of job tardiness, *Operations Research* 17 (1969) 701–715.

[9] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Operations Research* 5 (1979) 287–326.

[10] C. L. Hwang and A. S. M. Masud, Multiple objective decision making methods and applications: a state-of-the-art survey, *Springer Science and Business Media*, vol. 164 (2012).

[11] F. Jolai, M.S. Sangari and M. Babaie, Pareto simulated annealing and colonial competitive algorithm to solve an offline scheduling problem with rejection, *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 224:7 (2010) 1119–1131.

[12] C. Koulamas, The single-machine total tardiness scheduling problem: Review and extensions, *European Journal of Operational Research* 202 (2010) 1–7.

[13] C. Koulamas and G.J. Kyparisis, Single machine scheduling with release times, deadlines and tardiness objectives, *European Journal of Operational Research* 133 (2001) 447–453.

[14] E.L. Lawler, A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness, *Annals of Discrete Mathematics* 1 (1977) 331–342.

[15] J.-B. Lasserre and M. Queyranne, Generic scheduling polyhedral and a new mixed- integer formulation for single-machine scheduling, *Proceedings of the 2nd Integer Programming and Combinatorial Optimization Conference* (1992).

[16] K. Miettinen, F. Ruiz and A. P. Wierzbicki, Introduction to multi-objective optimization: interactive approaches, *In: J. Branke, K. Deb, K. Miettinen R. Słowiński (Eds.), Multiobjective Optimization*, Springer (2008) 27–57.

[17] A. Moghaddam, F. Yalaoui and L. Amodeo, Efficient meta-heuristics based on various dominance criteria for a single-machine bi-critera scheduling problem with rejection, *Journal of Manufacturing Systems*, 34 (2015) 12–22.

[18] A. Moghaddam, L. Amodeo, F. Yalaoui and B. Karimi, Single machine scheduling with rejection: minimizing total weighted completion time and rejection cost, *Applied Evolutionary Computation*, 3:2 (2012) 42–61.

[19] F.T. Nobibon and R. Leus, Exact algorithms for a generalization of the order acceptance and scheduling problem in a single-machine environment, *Computers and Operations Research* 38:10 (2011) 367–378.

[20] C. Oğuz, F.S. Salman and Z.B. Yalçın, Order acceptance and scheduling decisions in make-to-order systems, *International Journal of Production Economics*, 125:1 (2010) 200–211.

[21] A. Przybylski and X. Gandibleux, Multi-objective branch-and-bound, *European Journal of Operational Research* 260 (2017) 856–872.

[22] D. Shabtay, N. Gaspar and M. Kaspi, A survey on offline scheduling with rejection, *Journal of Scheduling* 16 (2013) 3–28.

[23] D. Shabtay, N. Gaspar and L. Yedidsion, A bicriteria approach to scheduling a single machine with rejection and positional penalties, *Journal of Combinatorial Optimization* 23:4 (2012) 395–424.

[24] D. Shabtay and N. Gaspar, Two-machine flow-shop scheduling with rejection, *Computers and Operations Research* 39:5 (2012) 1087–1096.

[25] G. Steiner and R. Zhang, Revised Delivery-Time Quotation in Scheduling with Tardiness Penalties, *Operations Research* 59:6 (2011) 1504–1511.

[26] R. Tadei, A. Grosso, and F. Della Croce, Finding the Pareto-optima for the total and maximum tardiness single machine problem, *Discrete Applied Mathematics* 124 (2002) 117–126.

[27] V. T'kindt and F. Della Croce, Enumeration of Pareto optima for a flowshop scheduling problem with two criteria, *INFORMS Journal of Computing* 19:1 (2007) 64–72.

[28] D.J. Wang, Y. Yin, and M. Liu. Bicriteria scheduling problems involving job rejection, controllable processing times and rate-modifying activity, *International Journal of Production Research*, 54:12 (2016) 3691–3705.

[29] L. Zhang L. Lu, and J. Yuan, Single-machine scheduling under the job rejection constraint, *Theoretical Computer Science* 411 (2010) 1877–1882.

[30] E. Zitzler and L. Thiele, Multiobjective optimization using evolutionary algorithms: A comparative case study, In A. E. Eiben et al., editors, *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature - PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 292–301, Springer, Berlin (1998).