J Sched (2009) 12: 401–415
DOI 10.1007/s10951-008-0098-0

A Lagrangian approach to single-machine scheduling problems with two competing agents

Alessandro Agnetis · Gianluca de Pascale · Dario Pacciarelli

Received: 9 March 2007 / Accepted: 18 December 2008 / Published online: 29 January 2009

Abstract In this paper, we develop branch-and-bound algorithms for several hard, two-agent scheduling problems, i.e., problems in which each agent has an objective function which depends on the completion times of its jobs only. Our bounding approach is based on the fact that, for all problems considered, the Lagrangian dual gives a good bound and can be solved exactly in strongly polynomial time. The problems addressed here consist in minimizing the total weighted completion time of the jobs of agent A, subject to a bound on the cost function of agent B, which may be: (i) total weighted completion time, (ii) maximum lateness, (iii) maximum completion time. An extensive computational experience shows the effectiveness of the approach.

Keywords Branch-and-bound · Lagrangian relaxation · Multi-agent scheduling · Single-machine scheduling

1 Introduction

Recent developments in the field of distributed decision making have triggered a growing interest towards multi-agent scheduling problems, i.e., in which different agents share a common processing resource, and each agent wants to minimize a cost function depending on its jobs only. These issues arise in different application contexts, including real-time systems, integrated service networks (Peha 1995), industrial districts (Albino et al. 2006), telecommunication systems (Arbib et al. 2004). Such situations can be addressed by typical approaches of multi-criteria optimization problems, such as combining the agents' criteria into a single objective function, minimizing one agent's cost function with a bound on the other agents' cost, or generating all Pareto-optimal solutions.

A number of papers investigate the two-agent setting. Baker and Cole Smith (2003) analyze the computational complexity of combining the agents' criteria into a single objective function in a scenario with two agents and various criteria (maximum completion time, total weighted completion time and maximum lateness). Agnetis et al. (2000, 2004) address both the problems of finding a single Pareto-optimal solution and enumerating the whole set of Pareto-optimal solutions for different machine settings, under three different criteria: maximum of a regular function (such as makespan or lateness), weighted total completion time, and number of tardy jobs. They study the nine problems arising for the different criteria combinations for the two agents, and state complexity results for most of the resulting cases. Cheng et al. (2006, 2008) provide further complexity and approximation results for some special cases, namely when the two agents wish to minimize the weighted number of tardy jobs and the case in which they both hold max-form objective functions. Ng et al. (2006) address the case in which one agent minimizes the total completion time of its jobs with a bound on the number of the other agent's tardy jobs. Peha (1995) investigates the problem of lexicographically minimizing the weighted number of tardy jobs of one agent and then the other agent's total weighted completion, providing polynomial time algorithms for the case in which jobs have unit length.

In this paper, we consider the situation in which two agents (called A and B) share a single machine for process-

A. Agnetis (✉) · G. de Pascale
Dipartimento di Ingegneria dell'Informazione, Università di Siena, Siena, Italy
e-mail: agnetis@dii.unisi.it

D. Pacciarelli
Dipartimento di Informatica e Automazione, Università Roma Tre, Roma, Italy

ing their respective non-preemptive jobs. In particular, we describe exact algorithms for the following NP-hard scheduling problems (Agnetis et al. 2004):

Problem WCWC. Given n_A jobs of the agent A (the A-jobs), n_B jobs of the agent B (the B-jobs) and an integer Q, find a schedule which minimizes the weighted sum of the completion times of the A-jobs such that the maximum value of the weighted sum of the completion times of the B-jobs does not exceed Q.

Problem WCL$_{\max}$. Given n_A jobs of the agent A (the A-jobs), n_B jobs of the agent B (the B-jobs) and an integer Q, each having a due-date, find a schedule which minimizes the weighted sum of the completion times of the A-jobs such that the maximum lateness of the B-jobs does not exceed Q.

Problem WCC$_{\max}$. Given n_A jobs of the agent A (the A-jobs), n_B jobs of the agent B (the B-jobs) and an integer Q, find a schedule which minimizes the weighted sum of the completion times of the A-jobs such that the maximum completion time of the B-jobs does not exceed Q.

For all three problems, we propose a branch-and-bound approach using a Lagrangian relaxation for the computation of the bound. The Lagrangian dual is solved in polynomial time in all three cases by *ad hoc* methods.

The plan of the paper is as follows. In Sect. 2, we introduce the notation and the terminology used throughout the paper. The above three problems are addressed in Sects. 3, 4 and 5, respectively. Section 6 displays the computational results. Finally, in Sect. 7, some conclusions are drawn. Most of the proofs for results in this paper are given in the Appendix.

2 Problems formulation and notation

In this section, we introduce the notation and terminology we use throughout the paper. There are two competing agents, called *agent A* and *agent B*. Each of them has a set of non-preemptive *jobs* to be processed on a common machine. The agent A has to execute the job set $J^A = \{J_1^A, J_2^A, \dots, J_{n_A}^A\}$, whereas the agent B has to execute the job set $J^B = \{J_1^B, J_2^B, \dots, J_{n_B}^B\}$. We call *A-jobs* and *B-jobs* the jobs of the two sets. The processing time of job J_i^A (J_j^B) will be denoted by p_i^A (p_j^B). Also, let $P_A = \sum_{i=1}^{n_A} p_i^A$ and $P_B = \sum_{j=1}^{n_B} p_j^B$. For A-jobs we will also consider weights w_i^A, and define $\delta_i^A = w_i^A/p_i^A$ as the *density* of such jobs. We suppose that the A-jobs are numbered according to non-increasing density; hence, δ_1^A and $\delta_{n_A}^A$ is the highest and the lowest density, respectively.

Each of the two agents will have to schedule its jobs on the machine complying with the presence of the other agent's jobs. Let σ indicate a schedule of the $n = n_A + n_B$ jobs, i.e., an assignment of starting times to the jobs of both agents. The completion times of jobs J_i^A and J_j^B in σ will be denoted as $C_i^A(\sigma)$ and $C_j^B(\sigma)$, respectively. We use the notation J_h, p_h, $C_h(\sigma)$ when referring to any job in the set $J^A \cup J^B$. Each agent has a certain objective function which depends on the completion times of its jobs only. We indicate by $f^A(\sigma)$ and $f^B(\sigma)$ the two functions. In this paper, we consider the three following scenarios:

$$\begin{aligned}
\mathit{WCWC}:\quad & f^A(\sigma) = \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma),\\
& f^B(\sigma) = \sum_{j=1}^{n_B} w_j^B C_j^B(\sigma);\\
\mathit{WCL}_{\max}:\quad & f^A(\sigma) = \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma),\\
& f^B(\sigma) = L_{\max}^B = \max_{j\in J^B}\{L_j^B(\sigma)\}\\
& \qquad = \max_{j\in J^B}\{C_j^B(\sigma) - d_j^B\};\\
\mathit{WCC}_{\max}:\quad & f^A(\sigma) = \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma),\\
& f^B(\sigma) = C_{\max}^B = \max_{j\in J^B}\{C_j^B(\sigma)\}.
\end{aligned}$$

Since in all cases the objective functions are regular (i.e., non-decreasing in the completion times), there is no convenience in keeping the machine idle, and therefore we can restrict ourselves to consider the set of active schedules, denoted by $\mathcal{S}$. According to Graham's notation, the problems addressed in this paper are denoted as:

- $1|\sum w_j^B C_j^B \le Q|\sum w_i^A C_i^A$;
- $1|L_{\max}^B \le Q|\sum w_i^A C_i^A$;
- $1|C_{\max}^B \le Q|\sum w_i^A C_i^A$.

Note that if one is interested in finding all the Pareto-optimal solutions of such problems, this can be achieved by solving several instances of the problem, varying the value of the constant Q.

3 Problem *WCWC*

In this section, we address the problem in which A wants to minimize the weighted sum of its jobs' completion times, while B requires that the weighted sum of its jobs' completion times does not exceed a certain quantity Q. The problem can be formulated as:

$$z^* = \min_{\sigma\in\mathcal{S}}\left\{\sum_{i=1}^{n_A} w_i^A C_i^A(\sigma) : \sum_{j=1}^{n_B} w_j^B C_j^B(\sigma) \le Q\right\}. \tag{1}$$

In Agnetis et al. (2004), it is proved that problem (1) is NP-hard even when all the weights for both agents are equal to one. Throughout Sect. 3, we indicate with $\delta_j^B = w_j^B/p_j^B$ the *density* of the jth B-job, similarly as we do for the A-jobs. We also suppose that the B-jobs are numbered according to non-increasing density; hence, δ_1^B and $\delta_{n_B}^B$ indicate the highest and the lowest values assumed by the B-jobs density.

3.1 Lagrangian relaxation and Lagrangian dual

In this section, we introduce the Lagrangian relaxation of problem (1) and give an efficient algorithm to solve the Lagrangian dual. Relaxing the constraints on B-jobs in problem (1) we get the Lagrangian problem:

$$L(\lambda) = \min_{\sigma\in\mathcal{S}} \left\{ \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma) + \lambda \left(\sum_{j=1}^{n_B} w_j^B C_j^B(\sigma) - Q \right) \right\}. \tag{2}$$

Note that for each value of $\lambda \geq 0$, problem (2) is in the format of a classical, single-agent problem $1||\sum \bar{w}_j C_j$, in which the weights are defined as follows:

$$\bar{w}_h = \begin{cases} w_h & \text{if } J_h \in J^A, \\ \lambda w_h & \text{if } J_h \in J^B. \end{cases}$$

As pointed out by Baker and Cole Smith (2003), the optimal schedule $\sigma(\lambda)$ for this problem can be found by simply applying the Smith's rule (Smith 1956), i.e., scheduling the jobs in non-increasing order of their ratio $\bar{w}_h/p_h$. (As a consequence, the order in which the A-jobs are scheduled and that in which the B-jobs are scheduled remain the same for all values of λ.) Each solution of (2) is a lower bound for (1). The Lagrangian dual of problem (1) consists in finding the highest of such bounds, i.e., computing

$$L(\lambda^*) = \max_{\lambda\geq 0} \left[\min_{\sigma\in\mathcal{S}} \left\{ \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma) + \lambda \left(\sum_{j=1}^{n_B} w_j^B C_j^B(\sigma) - Q \right) \right\} \right]. \tag{3}$$

It is well known that $L(\lambda)$ is a concave, piecewise linear function (Fig. 1). Calling *breakpoint* the values of λ in which the slope of $L(\lambda)$ changes, we note that λ^* is always a breakpoint. General methods for solving the Lagrangian dual (such as the subgradient method) generate a sequence of points asymptotically converging to λ^*. In our case, also exploiting the fact that λ is indeed a scalar, we are able to exactly compute λ^* in a polynomial number of computational steps.

Let $\lambda_{\min} = \delta_{n_A}^A/\delta_1^B$ and $\lambda_{\max} = \delta_1^A/\delta_{n_B}^B$. Notice that if $\lambda \leq \lambda_{\min}$, $\sigma(\lambda)$ is the schedule in which all the A-jobs are processed before all the B-jobs. Conversely, if $\lambda \geq \lambda_{\max}$, $\sigma(\lambda)$ is the schedule in which all the B-jobs are processed before all the A-jobs. (Clearly, in the latter case, if $\sigma(\lambda)$ is infeasible, so is the whole problem (1)). As λ monotonically decreases from $\lambda_{\max}$ to $\lambda_{\min}$, in the optimal schedule $\sigma(\lambda)$ the B-jobs migrate from the head to the tail of the schedule (Fig. 2). Each intermediate schedule is optimal for a certain range of values of λ. If $\bar{\lambda}$ is not a breakpoint, the slope of $L(\lambda)$ in $\bar{\lambda}$ is

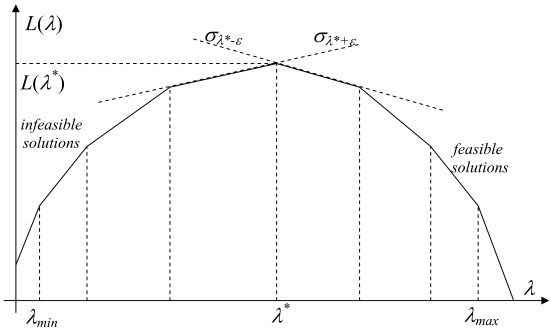

Fig. 1 Shape of Lagrangian function

$$\sum_{j=1}^{n_B} w_j^B C_j^B\big(\sigma(\bar{\lambda})\big) - Q,$$

which represents the violation of the constraint for schedule $\sigma(\bar{\lambda})$. If $\bar{\lambda}$ is a breakpoint, then, for a sufficiently small $\epsilon > 0$, the schedules $\sigma(\bar{\lambda}-\epsilon)$ and $\sigma(\bar{\lambda}+\epsilon)$ are obtained one from the other simply by swapping the jobs of all adjacent pairs (J_i^A, J_j^B) for which $\delta_i^A = \bar{\lambda}\delta_j^B$. (Note that there is at least one such pair.) Both these schedules are optimal for problem (2) with $\lambda = \bar{\lambda}$.

The maximum $L(\lambda^*)$ is attained at the breakpoint λ^* where the slope of $L(\lambda)$ switches from non-negative to non-positive, and therefore the schedule $\sigma(\lambda^*-\epsilon)$ is feasible. The breakpoint λ^* can be efficiently found as follows. First, notice that the total number of breakpoints cannot exceed $n_A n_B$ since going from $\sigma(\lambda_{\min}-\epsilon)$ to $\sigma(\lambda_{\max}+\epsilon)$ each B-job overtakes each A-job exactly once, and that all breakpoints can be generated in $O(n_A n_B)$, by computing the values $\lambda_{ij} = \delta_i^A/\delta_j^B$, $i = 1,\dots,n_A$, $j = 1,\dots,n_B$. Second, the schedule $\sigma(\lambda_{ij}-\epsilon)$ can be generated from $\sigma(\lambda_{ij}+\epsilon)$ by simply swapping all job pairs J_p^A, J_q^B such that $\delta_p^A/\delta_q^B = \lambda_{ij}$. If λ_{ij} and λ_{uv} are two consecutive breakpoints, with $\lambda_{ij} < \lambda_{uv}$, $\sigma(\lambda_{ij}+\epsilon) \equiv \sigma(\lambda_{uv}-\epsilon)$. Hence, all breakpoints can be a priori ordered, which requires $O(n_A n_B \log(n_A n_B))$. Thereafter, $L(\lambda^*)$ can be attained in $O(n_A n_B)$. In conclusion, the complexity is therefore dominated by the ordering of the breakpoints, and the following theorem holds.

Theorem 3.1 *Problem* (3) *can be solved in* $O(n_A n_B \log(n_A n_B))$.

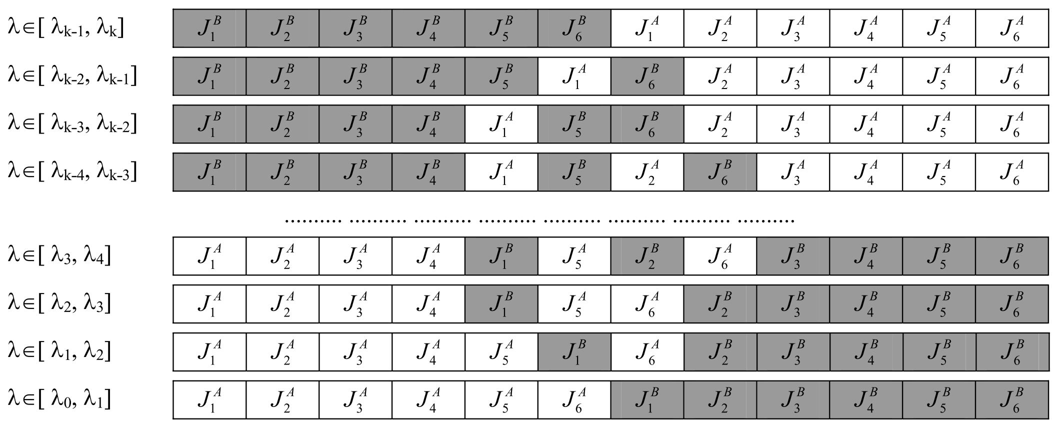
λ∈[λk-1, λk] J_1^B J_2^B J_3^B J_4^B J_5^B J_6^B J_1^A J_2^A J_3^A J_4^A J_5^A J_6^A
λ∈[λk-2, λk-1] J_1^B J_2^B J_3^B J_4^B J_5^B J_1^A J_6^B J_2^A J_3^A J_4^A J_5^A J_6^A
λ∈[λk-3, λk-2] J_1^B J_2^B J_3^B J_4^B J_1^A J_5^B J_6^B J_2^A J_3^A J_4^A J_5^A J_6^A
λ∈[λk-4, λk-3] J_1^B J_2^B J_3^B J_4^B J_1^A J_5^B J_2^A J_6^B J_3^A J_4^A J_5^A J_6^A
λ∈[λ3, λ4] J_1^A J_2^A J_3^A J_4^A J_1^B J_5^A J_2^B J_6^A J_3^B J_4^B J_5^B J_6^B
λ∈[λ2, λ3] J_1^A J_2^A J_3^A J_4^A J_1^B J_5^A J_6^A J_2^B J_3^B J_4^B J_5^B J_6^B
λ∈[λ1, λ2] J_1^A J_2^A J_3^A J_4^A J_5^A J_1^B J_6^A J_2^B J_3^B J_4^B J_5^B J_6^B
λ∈[λ0, λ1] J_1^A J_2^A J_3^A J_4^A J_5^A J_6^A J_1^B J_2^B J_3^B J_4^B J_5^B J_6^B

Fig. 2 Optimal schedules for problem (2) for decreasing values of λ, where $\lambda_0 = \lambda_{\min}$ and $\lambda_k = \lambda_{\max}$

3.2 Extension to the bicriteria problem

Here we want to briefly illustrate how to extend the approach described for problem (1) can be extended to address the more general bicriteria problem in which two weights w_h and v_h are given for each job J_h, and a single agent wants to minimize total weighted completion time using weights w_h, with a bound on total weighted completion time using weights v_h:

$$z^* = \min_{\sigma\in\mathcal{S}}\left\{\sum_{h=1}^{n} w_h C_h(\sigma) : \sum_{h=1}^{n} v_h C_h(\sigma) \le Q\right\}.$$

The Lagrangian dual problem becomes

$$L(\lambda^*) = \max_{\lambda\ge 0} L(\lambda)$$
$$= \max_{\lambda\ge 0}\min_{\sigma\in\mathcal{S}}\left\{\sum_{h=1}^{n}(w_h + \lambda v_h)C_h(\sigma) - \lambda Q\right\}. \qquad (4)$$

We can associate with each job J_h two densities $\delta_h^W = w_h/p_h$ and $\delta_h^V = v_h/p_h$. For $\lambda = 0$, the Lagrangian problem is solved by sorting the jobs according to densities δ_h^W (W-ordering), while for sufficiently large λ, the jobs are sorted according to densities δ_h^V (V-ordering). As λ increases, the slope of $L(\lambda)$ changes at a breakpoint $\bar{\lambda}$, such that two adjacent jobs J_i and J_j swap their positions:

$$\bar{\lambda} = \frac{\delta_i^W - \delta_j^W}{\delta_j^V - \delta_i^V}$$

for $0 \le \lambda \le \bar{\lambda}$, the two jobs follow the W-ordering, while for $\lambda \ge \bar{\lambda}$ they follow the V-ordering. (Note that if $\delta_i^W \ge \delta_j^W$ and $\delta_i^V \ge \delta_j^V$, J_i precedes J_j for all λ.) Since two jobs can overtake each other at most once, there are at most $O(n^2)$ breakpoints, and in conclusion the following theorem holds.

Theorem 3.2 *Problem* (4) *can be solved in* $O(n^2 \log n)$.

3.3 Enumeration

The algorithm described in the previous section has been embedded in a branch-and-bound scheme. The adopted branching rule simply consists in fixing the first l jobs in the schedule. In doing so, a simple dominance rule is enforced, i.e., two adjacent jobs belonging to the same agent must respect Smith's rule.

At each node of level l, the algorithm solves the Lagrangian dual of a problem having the same structure of the root problem, but only $n - l$ jobs to schedule. Notice that the breakpoints are ordered once for all at the root node, so that solving the Lagrangian dual at each node indeed requires only $O(n_A n_B)$ time. If either a subproblem is infeasible or it is trivially solved by scheduling all the B-jobs at the end, then the subproblem is fathomed.

The open subproblems are kept in an ordered dynamic list. At each step, the subproblem having the lowest bound is chosen from the list, and all of its subproblems are processed.

As shown in Sect. 3.1, one property of the optimal solution to the Lagrangian dual is that it is always attained for a schedule $\tilde{\sigma}$ which is feasible for the original problem. Such a schedule, that we get at no extra computational cost, can be exploited to compute an initial upper bound as well as to update the incumbent. Note that $\tilde{\sigma}$ is the schedule having largest $\sum w_j^B C_j^B$ among all feasible schedules encountered in the solution of the Lagrangian dual, so the quantity $|\sum w_j^B C_j^B(\tilde{\sigma}) - Q|$ is usually quite small (recall that since $\tilde{\sigma}$ is feasible, $\sum w_j^B C_j^B(\tilde{\sigma}) \le Q$). Since the difference between $L(\lambda^*)$ and the value of the feasible schedule $\sum w_i^A C_i^A(\tilde{\sigma})$ is $\lambda^*(\sum w_j^B C_j^B(\tilde{\sigma}) - Q)$, and since it turns out in most experiments that λ^* is typically close to 1, chances are that $\tilde{\sigma}$ is also a good schedule. Moreover, note that if $\sum w_j^B C_j^B(\tilde{\sigma}) - Q = 0$, then $\tilde{\sigma}$ is optimal for that subproblem, which can therefore be fathomed.

4 Problem $WCL_{\max}$

In this section, we address the problem in which A wants to minimize the weighted sum of its jobs' completion times, while B wants to complete each of its jobs J_j^B within Q from its due date d_j^B. This problem has been shown to be strongly NP-hard by Cheng et al. (2008).

We suppose that the B-jobs are numbered according to non-decreasing due dates (EDD order). We let $\bar{d}_{n_B} = d_{n_B}$ and, for $j = (n_B - 1), \dots, 1$, $\bar{d}_j = \min\{d_j, \bar{d}_{j+1} - p_{j+1}^B\}$. For each B-job we define a *shifted due date* $Q_j = \bar{d}_j + Q$, for which the following holds

$$Q_j \le Q_{j+1} - p_{j+1}^B, \quad j = 1, \dots, n_B - 1. \tag{5}$$

This simple preprocessing plays a key role to speed up the solution algorithm. Note that, with no loss of generality, we can always assume that the B-jobs are scheduled in EDD order in an optimal solution, and hence we can use shifted due dates in the formulation of the problem:

$$z^* = \min_{\sigma} \left\{ \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma) \right\}$$

$$\text{s.t. } C_j^B(\sigma) \le Q_j, \quad j = 1, \dots, n_B, \tag{6}$$

$$\sigma \in \mathcal{S}.$$

We note that $WCL_{\max}$ is a special case of the scheduling problem with deadlines $1|\tilde{d}_j|\sum w_j C_j$, in which jobs having weight $w_j > 0$ have infinite deadline. This problem has received some attention in the literature. In particular, Pan (2003) proposes an enumeration scheme which significantly enhances the results obtained using a bounding scheme previously proposed by Posner (1985). Our approach exploits the special structure of our problem. After introducing an algorithm for solving the Lagrangian dual (Sect. 4.2) and proving its correctness (Sect. 4.2.1), we will show that our bound is tighter than Posner's (Sect. 4.3). Also, we will discuss how we embedded Pan's ideas in an enumeration scheme for our problem (Sect. 4.4).

4.1 Lagrangian relaxation

Relaxing the n_B constraints on B-jobs' completion times in (6) we get the *Lagrangian problem*:

$$L(\lambda) = \min_{\sigma \in \mathcal{S}} \left\{ \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma) + \sum_{j=1}^{n_B} \lambda_j (C_j^B(\sigma) - Q_j) \right\} \tag{7}$$

```
Algorithm LDA
$i = 1$, $j = 1$, $T = 0$
while($i \le n_A + 1$)
{
  while($T + p_i^A > Q_j - p_j^B$ and $j \le n_B$)
  {
    let $\delta_j^B = \delta_i^A$
    schedule $J_j^B$, update $T \leftarrow T + p_j^B$
    update $j \leftarrow j + 1$
  }
  schedule $J_i^A$, update $T \leftarrow T + p_i^A$
  update $i \leftarrow i + 1$
}
```

Fig. 3 The Lagrangian dual algorithm

and the corresponding *Lagrangian Dual*:

$$L(\lambda^*) = \max_{\lambda \ge 0} \left[\min_{\sigma \in \mathcal{S}} \left\{ \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma) + \sum_{j=1}^{n_B} \lambda_j (C_j^B(\sigma) - Q_j) \right\} \right]. \tag{8}$$

We let $\delta_j^B = \lambda_j^B / p_j^B$, and denote with δ^A, δ^B and δ the vectors $(\delta_1^A, \dots, \delta_{n_A}^A)$, $(\delta_1^B, \dots, \delta_{n_B}^B)$ and $(\delta_1^A, \dots, \delta_{n_A}^A, \delta_1^B, \dots, \delta_{n_B}^B)$, respectively. The Lagrangian problem (8) then becomes

$$L(\delta^B) = \min_{\sigma \in \mathcal{S}} \left\{ \sum_{i=1}^{n_A} \delta_i^A p_i^A C_i^A(\sigma) + \sum_{j=1}^{n_B} \delta_j^B p_j^B (C_j^B(\sigma) - Q_j) \right\}. \tag{9}$$

Note that in (9) only the δ_j^B are variables, whereas the δ_i^A are given. Let us restate the dual problem (8) as

$$L(\delta^{*B}) = \max_{\delta^B \ge 0} \{ L(\delta^B) \}. \tag{10}$$

4.2 Algorithm

For convenience of exposition, we introduce a dummy A-job, $J_{n_A+1}^A$, such that $p_{n_A+1}^A = Q_{n_B}$ and $w_{n_A+1}^A = 0$. Clearly, such a job will be always scheduled last in an optimal solution to (6) and (9). We next consider the *Lagrangian Dual Algorithm* (*LDA*) for (10) reported in Fig. 3.

We denote with σ_{LDA}^* and δ_{LDA}^B the schedule and, respectively, the densities of the B-jobs given by LDA. Note that in LDA each B-job's δ_j^B is set equal to the value δ_i^A of some A-job J_i^A. We denote with $\mathcal{J}_i^B$ the set of the B-jobs whose δ_j^B was set equal to δ_i^A, and call *cluster* $\mathcal{J}_i$ the set $\mathcal{J}_i^B \cup \{J_i^A\}$. Note that the B-jobs belonging to the same cluster are scheduled consecutively, followed by the corresponding A-job at the end of the cluster.

4.2.1 Correctness of the Lagrangian dual algorithm

In this section, we show that LDA optimally solves problem (10). To this aim, we first need to establish some preliminary results.

Proposition 4.1 *If* (6) *is feasible then, for each* J_j^B, (a) $Q_j - C_j^B(\sigma^*_{\text{LDA}}) \geq 0$ *and* (b) *letting* $\mathcal{J}_i$ *be the cluster* J_j^B *belongs to,* $Q_j - C_j^B(\sigma^*_{\text{LDA}}) < p_i^A$.

Proof For simplicity, in this proof we omit σ^*_{LDA} from the completion times notation (so we write C_j^B for $C_j^B(\sigma^*_{\text{LDA}})$). We prove the thesis by induction. From the feasibility of (6), it immediately follows that $Q_1 - p_1^B \geq 0$. Then J_1^B will be scheduled starting at time $T = \sum_{k=1}^{i-1} p_k^A$, where i such that $T \leq Q_1 - p_1^B < T + p_i^A$ (or $T = 0$ if $i = 1$). Hence, $C_1^B = T + p_1^B \leq Q_1^B$, proving (a) for $j = 1$, and $Q_1 - C_1^B = Q_1 - T - p_1^B < p_i^A$, proving (b) for $j = 1$.

Now, suppose that (a) and (b) hold for job J_j^B. Note that, after J_j^B has been scheduled, T is set equal to C_j^B. Two cases can occur:

(i) $C_j^B + p_i^A > Q_{j+1} - p_{j+1}^B$. In this case, J_{j+1}^B is scheduled immediately after the end of J_j^B, and

$$\begin{aligned} C_{j+1}^B &= C_j^B + p_{j+1}^B \\ &\leq Q_j + p_{j+1}^B \qquad \text{by the induction hypothesis} \\ &\leq Q_{j+1} \qquad \text{for (5)} \end{aligned}$$

proving (a). Moreover, $Q_{j+1} - C_{j+1}^B = Q_{j+1} - C_j^B - p_{j+1}^B < p_i^A$, proving (b). (Note that, in particular, for cluster $\mathcal{J}_{n_A+1}$ this holds because $p_{n_A+1}^A = Q_{n_B}$.)

(ii) $C_j^B + p_i^A \leq Q_{j+1} - p_{j+1}^B$. In this case, after the end of job J_j^B, the A-jobs $J_i^A, \dots, J_{i'}^A$ will be consecutively scheduled, where i' is such that

$$C_j^B + \sum_{k=i}^{i'-1} p_k^A \leq Q_{j+1} - p_{j+1}^B < C_j^B + \sum_{k=i}^{i'} p_k^A.$$

Then J_{j+1}^B is scheduled to start at the end of $J_{i'-1}^B$, and so $C_{j+1}^B = C_j^B + \sum_{k=i}^{i'-1} p_k^A + p_{j+1}^B \leq Q_{j+1}$, proving (a). Moreover, $Q_{j+1} - C_{j+1}^B = Q_{j+1} - C_j^B - \sum_{k=i}^{i'-1} p_k^A - p_j^B < p_{i'}^A$, proving (b). (Again, for $\mathcal{J}_{n_A+1}$ this holds because $p_{n_A+1}^A = Q_{n_B}$.) □

Note that schedule σ^*_{LDA} is feasible as a consequence of (a).

The following is a well known general property that holds for all scheduling problems, and can be easily proved by a simple pairwise interchange argument.

Proposition 4.2 *Let* σ *be a schedule of the n jobs* $\{J_1, \dots, J_n\}$ *with durations* $\{p_1, \dots, p_n\}$, *and let* $\mathcal{I} \subseteq \{J_1, \dots, J_n\}$ *be a subset of consecutively scheduled jobs. The quantity*

$$\sum_{J_i \in \mathcal{I}} p_i C_i(\sigma) \tag{11}$$

does not depend on the ordering of the jobs in $\mathcal{J}$.

Lemma 4.3 *Given* σ^*_{LDA} *and any cluster* $\mathcal{J}_i$, *let* $\tilde{\sigma}$ *be any schedule obtained from* σ^*_{LDA} *by arbitrarily reordering the jobs within cluster* $\mathcal{J}_i$. *Also, denote with* $J_{\pi_1}^B, \dots, J_{\pi_{\ell-1}}^B, J_{\pi_\ell}^A, J_{\pi_{\ell+1}}^B, \dots, J_{\pi_{|\mathcal{J}_i|}}^B$ *the jobs belonging to* $\mathcal{J}_i$, *ordered according to their position in the schedule* $\tilde{\sigma}$. *Then*

$$\sum_{k=1}^{h} p_{\pi_k}^B \left(C_{\pi_k}^B(\tilde{\sigma}) - Q_{\pi_k}\right) \leq 0 \quad \forall h = 1, \dots, \ell - 1, \tag{12}$$

$$\sum_{k=h}^{|\mathcal{J}_i|} p_{\pi_k}^B \left(C_{\pi_k}^B(\tilde{\sigma}) - Q_{\pi_k}\right) > 0 \quad \forall h = \ell + 1, \dots, |\mathcal{J}_i|. \tag{13}$$

Proof Let $\bar{\sigma}$ be the schedule obtained reordering the jobs of σ^*_{LDA} within cluster $\mathcal{J}_i$ as follows: schedule in EDD order the B-jobs $J_{\pi_1}^B, \dots, J_{\pi_{\ell-1}}^B$, then schedule J_i^A, then schedule in EDD order the B-jobs $J_{\pi_{\ell+1}}^B, \dots, J_{\pi_{|\mathcal{J}_i|}}^B$. Note that $C_{\pi_k}^B(\bar{\sigma}) \leq C_{\pi_k}^B(\sigma^*_{\text{LDA}})$ for any $k < \ell$. Then, from Proposition 4.1(a), it follows that $C_{\pi_k}^B(\bar{\sigma}) - Q_{\pi_k} \leq C_{\pi_k}^B(\sigma^*_{\text{LDA}}) - Q_{\pi_k} \leq 0$ for any $k < \ell$. Also, note that $C_{\pi_k}^B(\bar{\sigma}) \geq C_{\pi_k}^B(\sigma^*_{\text{LDA}}) + p_i^A$ for any $k > \ell$, then from Proposition 4.1(b) it follows that $C_{\pi_k}^B(\bar{\sigma}) - Q_{\pi_k} \geq C_{\pi_k}^B(\sigma^*_{\text{LDA}}) - Q_{\pi_k} + p_i^A > 0$ for any $k > \ell$. Hence, for schedule $\bar{\sigma}$ it holds

$$\sum_{k=1}^{h} p_{\pi_k}^B \left(C_{\pi_k}^B(\bar{\sigma}) - Q_{\pi_k}\right) \leq 0 \quad \forall h = 1, \dots, \ell - 1, \tag{14}$$

$$\sum_{k=h}^{|\mathcal{J}_i|} p_{\pi_k}^B \left(C_{\pi_k}^B(\bar{\sigma}) - Q_{\pi_k}\right) > 0 \quad \forall h = \ell + 1, \dots, |\mathcal{J}_i|. \tag{15}$$

Schedule $\tilde{\sigma}$ differs from $\bar{\sigma}$ only by the order of the jobs within the two blocks of consecutively scheduled B-jobs, before and after J_i^A, respectively. Then, in view of Proposition 4.2, the thesis holds. □

We are now in the position of proving the main result of this section.

Theorem 4.4 *Algorithm LDA correctly solves problem* (10) *in* $O(n)$.

Proof We first show that vector δ_{LDA}^B is optimal for (10). From the concavity of $L(\delta^B)$, it is sufficient to prove that

δ^B_{LDA} is locally optimal. To this aim, let us consider an arbitrary perturbed vector $\delta^B_{\text{LDA}} + \epsilon\Delta$, where $\epsilon > 0$ is a sufficiently small scalar so that if $\delta^B_h > \delta^B_k$, then $\delta^B_h + \epsilon\Delta_h > \delta^B_k + \epsilon\Delta_k$. By resequencing all jobs by the Smith's rule, we obtain an optimal schedule $\tilde{\sigma}$ for Problem (9) in which the jobs that belong to the same cluster in σ^*_{LDA} are still sequenced consecutively (in nonincreasing order of Δ_j).

The value of the perturbed Lagrangian function can be expressed as a summation over all clusters:

$$L\big(\delta^B_{\text{LDA}} + \epsilon\Delta\big) = \sum_{i=1}^{n_A}\Bigg[\delta^A_i p^A_i C^A_i(\tilde{\sigma}) + \sum_{J^B_j \in \mathcal{J}^B_i} \big(\delta^B_j + \epsilon\Delta_j\big) p^B_j \big(C^B_j(\tilde{\sigma}) - Q_j\big)\Bigg] \tag{16}$$

$$= \underbrace{\sum_{i=1}^{n_A}\delta^A_i\Bigg[p^A_i C^A_i(\tilde{\sigma}) + \sum_{J^B_j \in \mathcal{J}^B_i} p^B_j\big(C^B_j(\tilde{\sigma}) - Q_j\big)\Bigg]}_{\#1} + \epsilon \underbrace{\sum_{i=1}^{n_A}\Bigg[\sum_{J^B_j \in \mathcal{J}^B_i} \Delta_j p^B_j\big(C^B_j(\tilde{\sigma}) - Q_j\big)\Bigg]}_{\#2}. \tag{17}$$

From Proposition 4.2, it follows that the term #1 is equal to $L(\delta^B_{\text{LDA}})$. Then the variation of the Lagrangian function is given by the term #2. We next show that all terms in square brackets in the summation #2 are non-positive. Each of such terms can be split into two parts, taking into account the B-jobs for which $\Delta_j > 0$ and $\Delta_j < 0$, respectively. Note that the B-jobs for which Δ_j is strictly positive are scheduled before J^A_i in $\tilde{\sigma}$ and the B-jobs for which Δ_j is strictly negative are scheduled after J^A_i in $\tilde{\sigma}$, so

$$L\big(\delta^B_{\text{LDA}} + \epsilon\Delta\big) - L\big(\delta^B_{\text{LDA}}\big) = \epsilon\sum_{i=1}^{n_A}\Bigg[\sum_{J^B_j \in J^B_i}^{\Delta_j > 0} \Delta_j p^B_j\big(C^B_j(\tilde{\sigma}) - Q_j\big) + \sum_{J^B_j \in J^B_i}^{\Delta_j < 0} \Delta_j p^B_j\big(C^B_j(\tilde{\sigma}) - Q_j\big)\Bigg]. \tag{18}$$

For any cluster $\mathcal{J}_i$, denote with $J^B_{\pi_1}, \ldots, J^A_{\pi_\ell}, \ldots, J^B_{\pi_{|\mathcal{J}_i|}}$ the jobs belonging to $\mathcal{J}_i$, ordered according to their position in the schedule $\tilde{\sigma}$ (i.e., by non-increasing values of Δ_j). Letting $\Delta_{\pi_\ell} = 0$ for uniformity of notation, the summations for $\Delta_j > 0$ and $\Delta_j < 0$ in (18) can be respectively rewritten as

$$\sum_{k=1}^{\ell-1} \Delta_{\pi_k} p^B_{\pi_k}\big(C^B_{\pi_k}(\tilde{\sigma}) - Q_{\pi_k}\big) = \sum_{h=1}^{\ell-1} \underbrace{(\Delta_{\pi_h} - \Delta_{\pi_{h+1}})}_{\#1} \underbrace{\sum_{k=1}^{h} p^B_{\pi_k}\big(C^B_{\pi_k}(\tilde{\sigma}) - Q_{\pi_k}\big)}_{\#2}, \tag{19}$$

$$\sum_{k=\ell+1}^{|\mathcal{J}_i|} \Delta_{\pi_k} p^B_{\pi_k}\big(C^B_{\pi_k}(\tilde{\sigma}) - Q_{\pi_k}\big) = \sum_{h=\ell+1}^{|\mathcal{J}_i|} \underbrace{(\Delta_{\pi_h} - \Delta_{\pi_{h-1}})}_{\#1} \underbrace{\sum_{k=h}^{|\mathcal{J}_i|} p^B_{\pi_k}\big(C^B_{\pi_k}(\tilde{\sigma}) - Q_{\pi_k}\big)}_{\#2}. \tag{20}$$

Since $\Delta_{\pi_h} \geq \Delta_{\pi_{h+1}}$, all the coefficients #1 in (19) are non-negative, and from Lemma 4.3, all the terms #2 in (19) are non-positive. Similarly, all the coefficients #1 in (20) are non-positive, while from Lemma 4.3 all the terms #2 in (20) are strictly positive. Then $L(\delta^B_{\text{LDA}} + \epsilon\Delta) - L(\delta_{\text{LDA}})$, being the sum of non-positive terms, is non-positive, which implies that $L(\delta^B_{\text{LDA}})$ is a local maximum.

As for the complexity of LDA, we observe that each job is considered exactly once throughout the algorithm. Assuming that the A-jobs have preliminarily been ordered according to WSPT and B-jobs according to EDD, the thesis follows. □

4.3 Comparison with Posner's lower bound

In this section, we show that our bounding scheme is strictly better that the best known bound in the literature. Let us first briefly review Posner's bounding scheme (PBS), when applied to problem (6). PBS works in two phases and builds a preemptive schedule from left to right as follows. (1) Each B-job J^B_j is scheduled to start at its latest start time, $Q_j - p^B_j$ (recall that because of (5) this is always possible with no overlap among B-jobs). (2) The A-jobs are scheduled in WSPT order, filling the gaps left by the B-jobs. Whenever an A-job does not fit one interval, it is split into two smaller jobs, each having the same density of the original job. To be more precise, suppose that we have built the schedule up to time T and J^A_i is the next A-job to be scheduled, in the interval $\mathcal{G} = [T, Q_j - p^B_j]$. If J^A_i fits the interval, it is scheduled to start at T, and T is updated to $T + p^A_i$. Else, J^A_i is split into two smaller *split jobs*, say J' and J''. J' has length p' equal to the size of $\mathcal{G}$, while J'' has length $p^A_i - p'$. The weights of J' and J'' are such that both split jobs have the same density of J^A_i. Then, J' is scheduled to fill the interval $\mathcal{G}$ while J'' will be scheduled in the next gap.

Posner (1985) shows that the total weighted completion time of such a schedule (including all newly generated split jobs) gives a lower bound for (6). We next show that the bound provided by PBS is always not tighter than the Lagrangian dual bound provided by LDA, and the two bounds coincide only when both bounding schemes attain the optimal solution.

Theorem 4.5 *Let L_{PBS} be the value of Posner's lower bound for* (6). *Then $L_{\mathrm{PBS}} \le L(\delta^{*B})$, and $L_{\mathrm{PBS}} = L(\delta^{*B})$ if and only if PBS solves* (6) *to optimality.*

Proof See Appendix. □

4.4 Enumeration

The algorithm described in the previous sections has been embedded in a branch-and-bound scheme. The branching rule consists in fixing *the first A-jobs* in the schedule. Notice that we need not consider the B-jobs in the branching rule. In fact, if the sequence of A-jobs is fixed, the position of each B-job J_j^B is consequently determined as the latest position which still allows J_j^B to complete within Q_j.

Open subproblems are kept in a dynamic list ordered by the value of the bound. From such a list, the subproblem with the most promising bound is extracted and branching is performed. For each generated child, the following processing steps are carried out.

(i) A fast heuristic is applied to find a (possibly good) solution for the corresponding subproblem to be compared with the actual incumbent, or to check the infeasibility of the current node. The heuristic simply schedules each B-job as late as possible (from the last to the first), and then fills the space between B-jobs with the A-jobs (from the first to the last, in WSPT order).

(ii) A dominance test is checked, similarly to what Pan (2003) proposed for $1|\tilde{d}_j|\sum w_jC_j$. In particular, at level h of the enumeration tree, the first h A-jobs are fixed. Then, among all possible permutations of the last k fixed A-jobs, we search for one that strictly improves the cost function. If such a permutation exists, we can then fathom the current node. In our experiments, we set such a *retrospect depth* k to 5, which gave the best results on some preliminary tests.

(iii) If the node has not been fathomed, the Lagrangian bound is evaluated and the node is inserted in the list of open nodes.

5 Problem *WCC*$_{\max}$

In this section, we address the problem in which A wants to minimize the weighted sum of its jobs' completion times, while B wants to complete all its jobs within Q. Of course, this can be viewed as a special case of $1|L^B_{\max} \le Q|\sum w_j^A C_j^A$, obtained when $d_j^B = 0$ for all B-jobs. We exploit the peculiarity of this special case to devise a specific, more efficient solution algorithm. The problem is still NP-hard (Agnetis et al. 2004), but can be solved in pseudo-polynomial time (Sect. 5.3).

A simple but relevant observation (Baker and Cole Smith 2003) is that in an optimal solution to $1|C^B_{\max} \le Q|\sum w_j^A C_j^A$, all the B-jobs are scheduled consecutively. (If a B-job exists which is not the last and is followed by an A-job, we can always swap the B-job and the A-job getting a schedule which is strictly better for A and equivalent for B.) Hence, from now on, in this section we assume that J^B consists of a single job J_B, of length p^B, and therefore denote by $C^B(\sigma)$ its completion time in σ (which equals $C^B_{\max}(\sigma)$). This observation implies that $1|C^B_{\max} \le Q|\sum w_j^A C_j^A$ is also a special case of *WCWC*, in which B holds a single job of weight 1.

$$z^* = \min_{\sigma}\left\{\sum_{i=1}^{n_A} w_i^A C_i^A(\sigma)\right\}$$
$$\text{s.t. } C^B(\sigma) \le Q, \tag{21}$$
$$\sigma \in \mathcal{S}.$$

By the above observation, the structure of an optimal solution to problem (21) is

$$\{J^A_{\mathrm{prec}}\}\{J_B\}\{J^A_{\mathrm{succ}}\}, \tag{22}$$

where $J^A_{\mathrm{prec}} \cup J^A_{\mathrm{succ}} = J^A$ and $J^A_{\mathrm{prec}} \cap J^A_{\mathrm{succ}} = \emptyset$. Another property is stated in the following proposition which can be easily established by a simple pairwise interchange argument (Smith's rule).

Proposition 5.1 *In an optimal solution σ^* to $1|C^B_{\max} \le Q|\sum w_j^A C_j^A$, jobs in J^A_{prec} and J^A_{succ} are ordered by nonincreasing values of the ratio w_i^A/p_i^A.*

Due to this proposition, the optimal solution is completely defined by the partition $(J^A_{\mathrm{prec}}, J^A_{\mathrm{succ}})$ of the set J^A.

5.1 Lagrangian relaxation and Lagrangian dual

In the following sections, we first address the solution to the Lagrangian dual of problem (21) at the root node of the enumeration tree, i.e., with no branching constraints. The structure of its solution will be exploited to devise a branching rule. Finally, we give an algorithm to solve the Lagrangian dual of problem (21) at non-root nodes.

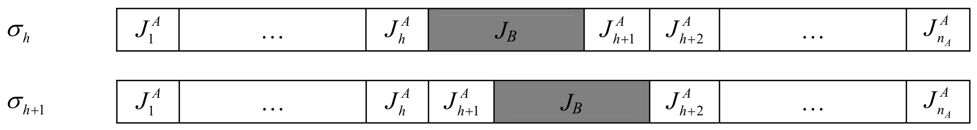

Fig. 4 A pair of critical schedules

5.1.1 *Solving the root node*

Relaxing the constraint on C^B in Problem (21), we get the Lagrangian problem:

$$L(\lambda) = \min_{\sigma\in\mathcal{S}} \left\{ \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma) + \lambda\big(C^B(\sigma) - Q\big) \right\}. \tag{23}$$

As in the previous sections, we address the Lagrangian dual of problem (21)

$$L(\lambda^*) = \max_{\lambda\geq 0}\left[\min_{\sigma\in\mathcal{S}} \left\{ \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma) + \lambda\big(C^B(\sigma) - Q\big) \right\} \right]. \tag{24}$$

In the following, we indicate with $J^A(\delta)$ the set of the A-jobs whose densities are greater or equal to δ. For each value of $\lambda \geq 0$, the solution to problem (23) can be simply found by first scheduling the A-jobs belonging to set $J^A(\lambda/p^B)$ ordered according to Smith's rule, then the B-job, and finally the remaining A-jobs ordered according to Smith's rule.

We next show that as λ varies from 0 to $+\infty$, there are $n_A + 1$ different schedules, which are optimal for problem (23). If $\lambda/p^B \geq \delta_1^A$, the optimal schedule for problem (23) (call it σ_0) is obtained by scheduling the B-job before all A-jobs. (Such a schedule is feasible for the original problem, unless problem (21) is infeasible.) Varying λ/p^B from δ_1^A down to $\delta_{n_A}^A$, we get a sequence $\mathcal{S} = \{\sigma_1, \dots, \sigma_i, \dots \sigma_{n_A-1}\}$ of $n_A - 1$ schedules, with σ_i optimal for $\lambda \in [\delta_{i+1}^A p^B, \delta_i^A p^B]$ and in which J_B is scheduled immediately after job J_i^A. Finally, for $\lambda/p^B \leq \delta_{n_A}^A$, the optimal schedule for problem (23) (call it σ_{n_A}) is obtained by scheduling the B-job after all A-jobs. (Note that such a schedule is infeasible for the original problem, unless the problem itself is trivial.)

In the sequence $\mathcal{S}$, there is an index h such that $\sigma_0, \sigma_1, \dots, \sigma_h$ are feasible, and $\sigma_{h+1}, \dots, \sigma_{n_A}$ are infeasible. The value of λ (call it *critical*) for which the optimal schedule switches from σ_h to σ_{h+1} is $(w_{h+1}^A/p_{h+1}^A)p^B$. Note that the two *critical* schedules σ_h and σ_{h+1} are both optimal when $\lambda = \delta_{h+1}^A p^B$, and they are obtained one from the other simply swapping J^B with the *critical* job J_{h+1}^A.

Whether any of the schedules in the sequence $\mathcal{S}$ is feasible or not for the original problem can be easily checked by the following condition:

$$C^B = \sum_{i\in J(\lambda/p^B)} p_i^A + p^B \leq Q. \tag{25}$$

Hence, we can find the last feasible schedule and the first infeasible schedule in the sequence $\mathcal{S}$ by simply adding one by one the durations of the A-jobs until their total duration exceeds the quantity $(Q - p^B)$. The critical job J_{h+1}^A is the last added job when infeasibility arises, i.e., such that

$$\sum_{i=1}^{h} p_i^A \leq Q - p^B < \sum_{i=1}^{h+1} p_i^A. \tag{26}$$

Hence, to obtain the optimal value λ^* for the Lagrangian dual we only need to build the two critical schedules σ_h and σ_{h+1} by (26), and then compute the value of λ such that:

$$\sum_{i=1}^{n_A} w_i^A C_i^A(\sigma_h) + \lambda\big(C^B(\sigma_h) - Q\big) = \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma_{h+1}) + \lambda\big(C^B(\sigma_{h+1}) - Q\big). \tag{27}$$

Figure 4 illustrates the structure of two critical schedules σ_h and σ_{h+1}. From (27), and since $C^B(\sigma_h) = C^B(\sigma_{h+1}) + p_{h+1}^A$, $C_{h+1}^A(\sigma_h) = C_{h+1}^A(\sigma_{h+1}) - p^B$, while the completion time of all other A-jobs is unchanged, one gets $\lambda^* = \delta_{h+1}^A p^B$.

5.1.2 *Branching strategy*

Due to the structure of the optimal solution to problem (24), i.e., the existence of a critical A-job, we branch at the root node by defining two subproblems. In one subproblem, the critical job is constrained to be scheduled before the B-job, in the other it is constrained to be scheduled after the B-job.

The same branching strategy can be applied at any non-root node, provided that a suitable definition of critical job is given. This leads to a binary enumeration tree in which branching constraints at each node result in precedence constraints between some A-jobs and J_B. Hence, in the general case, we have to deal with precedence constraints in the form of a tripartition $\mathcal{C} = \{P, S, F\}$ of the job set J^A, with P, S and F being, respectively, the set of A-jobs constrained to precede J^B, the set of A-jobs constrained to follow J^B, and the set of unconstrained (free) A-jobs.

$$z^* = \min_{\sigma} \left\{ \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma) \right\}$$
$$\text{s.t. } C^B(\sigma) \leq Q,$$

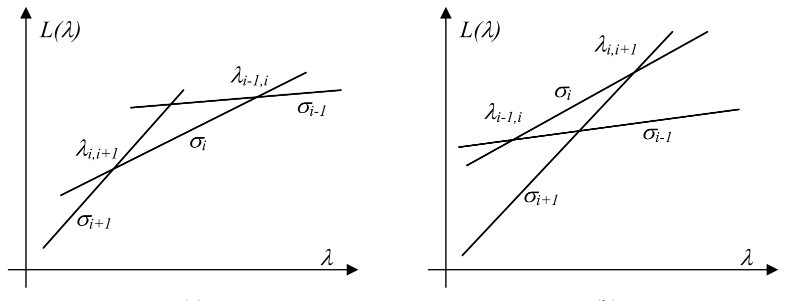

Fig. 5 Breakpoints on the Lagrangian function

$$
\begin{aligned}
&J_i^A \prec J^B \quad \forall i \in P, && (28)\\
&J_i^A \succ J^B \quad \forall i \in S,\\
&\sigma \in \mathcal{S}.
\end{aligned}
$$

We refer to jobs in $P \cup S$ as *constrained*. The structure of an optimal solution to problem (28) is still given by (22), where now $P \subseteq J^A_{\text{prec}}$ and $S \subseteq J^A_{\text{succ}}$. The property stated in Proposition 5.1 still holds, too.

5.1.3 Solving non-root nodes

In this section, we extend the results of Sect. 5.1.1 to solve the Lagrangian dual for a subproblem defined by constraints $\mathcal{C} = \{P, S, F\}$. The Lagrangian relaxation of such a subproblem is:

$$
\begin{aligned}
L(\lambda) = \min_{\sigma\in\mathcal{S}} &\left\{ \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma) + \lambda\big(C^B(\sigma) - Q\big) \right\}\\
\text{s.t. } &J_i^A \prec J^B \quad \forall i \in P, \qquad (29)\\
&J_i^A \succ J^B \quad \forall i \in S.
\end{aligned}
$$

As in the previous section, we address the Lagrangian dual problem. Let $\bar{n} = |F|$, and let $J^F(\delta)$ be the set of free A-jobs whose density w_i^A/p_i^A is greater than or equal to δ. Similar to what is done at the root node, for each $\lambda \geq 0$, the solution to (29) can be found by first scheduling the A-jobs belonging to set $P \cup J^F(\lambda/p^B)$ ordered according to Smith's rule, then the B-job, and finally the remaining A-jobs ordered according to Smith's rule. At each subproblem of the enumeration tree, there is a sequence $\mathcal{S} = \{\sigma_0 \dots \sigma_k \dots \sigma_{\bar{n}}\}$ of $\bar{n}+1$ schedules having such a structure. Each of them corresponds to a line in the $(\lambda, L(\lambda))$-plane (Fig. 5):

$$
\ell_{\sigma_k}(\lambda) = \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma_k) + \lambda\big(C^B(\sigma_k) - Q\big).
$$

Note that in the schedule σ_k the job B is scheduled after the kth and before the $(k+1)$th free A-job. Each schedule of $\mathcal{S}$ is optimal for (29) as λ/p^B varies from 0 to $+\infty$. Whether any of these schedules is feasible or not for the original problem can be easily checked by the following condition which generalizes (25):

$$
\sum_{i\in J^F(\lambda/p^B)} p_i^A + \sum_{i\in P} p_i^A + p^B = C^B \leq Q.
$$

Let $\lambda_{i-1,i}$ be the value for which lines $\ell_{\sigma_{i-1}}(\lambda)$ and $\ell_{\sigma_i}(\lambda)$ intersect. The following result ensures that for each pair of adjacent schedules (σ_{i-1}, σ_i) in the sequence $\mathcal{S}$, the point $\lambda_{i-1,i}$ is a breakpoint of the Lagrangian function, as shown in Fig. 5(a). (In other words, the situation of Fig. 5(b) cannot occur.)

Lemma 5.2 $\lambda_{i-1,i} \geq \lambda_{i,i+1}$.

Proof See Appendix. □

As a consequence of this lemma, the breakpoint corresponding to λ^* is the intersection of the lines corresponding to the last feasible schedule and the first infeasible schedule in the sequence $\mathcal{S}$. We can find such critical schedules by a very similar condition to (26). Adding one by one the durations of *free* A-jobs, the job J^A_{h+1} such that

$$
\sum_{\substack{i\leq h\\ i\in F}} p_i^A \leq Q - \sum_{i\in P} p_i^A - p^B < \sum_{\substack{i\leq h+1\\ i\in F}} p_i^A
$$

is the critical job. As for the root node, we solve the Lagrangian dual by constructing only the two critical schedules, say σ_h and σ_{h+1}, and by computing the value λ^* such that

$$
\begin{aligned}
&\sum_{i=1}^{n_A} w_i^A C_i^A(\sigma_h) + \lambda^*\big(C^B(\sigma_h) - Q\big)\\
&\quad = \sum_{i=1}^{n_A} w_i^A C_i^A(\sigma_{h+1}) + \lambda^*\big(C^B(\sigma_{h+1}) - Q\big).
\end{aligned}
$$

The structure of σ_h and σ_{h+1} is shown in Fig. 6, in which $\tilde{P}$ is the set of A-jobs belonging to P having density less than w^A_{h+1}/p^A_{h+1}, and $\tilde{S}$ is the set of A-jobs belonging to S

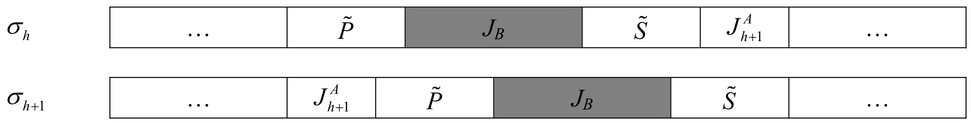

Fig. 6 A pair of critical schedules

having density greater than w^A_{h+1}/p^A_{h+1}. From the structure of the critical schedules, it can be easily obtained

$$\lambda^* = \delta^A_{h+1}\left(p^B + \sum_{i\in\tilde{P}\cup\tilde{S}} p^A_i\right) - \sum_{i\in\tilde{P}\cup\tilde{S}} w^A_i.$$

Concerning the complexity, note that $\tilde{P}$ and $\tilde{S}$ can be built in $O(n)$. Then, finding the critical index h requires $O(n_A)$ additions, as well as the computation of the value λ^*. Hence, the following result holds:

Theorem 5.3 *Problem* (28) *can be solved in* $O(n)$.

5.2 Enumeration

As for the previous problems, the Lagrangian bound has been used in a branch-and-bound scheme. Unlike previous cases, the branching rule adopted is binary, and consists in constraining the A-job to belong to P, or, respectively, S in the two subproblems. Thus, at each node of level l the algorithm solves the Lagrangian dual of problems having the same job sets of the root problem, but l precedence constraints. At each step, the subproblem having lowest bound is extracted from the list.

Note that the critical schedule σ_h found at a node of the enumeration tree is feasible for the original problem (21). Such solutions can be used to obtain an initial upper bound and for effective tree fathoming. In fact, arguments similar to that given in Sect. 3.3 show that the difference between $L(\lambda^*)$ and the value of the feasible schedule $\sum w^A_i C^A_i(\tilde{\sigma})$ is usually small, so $\tilde{\sigma}$ is often a good schedule.

5.3 A pseudo-polynomial algorithm

As pointed out by Sourd (2008), problem $WCC_{\max}$ can, in fact, be solved in pseudo-polynomial time by means of a dynamic programming algorithm.

Let $F(k,t_1,t_2)$ be the optimal value of a subproblem restricted to the first k A-jobs (numbered according to their density) and the B-job, in which the machine continuously works between 0 and t_1, it is idle from t_1 to t_2, and the B-jobs starts at t_2. This means that each A-job is either processed within 0 and t_1, after $t_2 + p^B$. In particular, in an optimal solution to such a restricted problem, either job J^A_k completes at time t_1, or it is the last job in the schedule, hence completing at time $p^B + (t_2 - t_1) + \sum_{i=1}^k p^A_i$. $F(k,t_1,t_2)$ is taken equal to 0 if $k = 0$ and $t_1 = 0$, while it is taken equal to $+\infty$ if no feasible schedule exists for the associated subproblem, i.e.:

$$F(k,t_1,t_2) = +\infty \quad \text{for all } k \text{ and for } t_1 < 0 \text{ or } t_1 > t_2,$$
$$F(0,t_1,t_2) = +\infty \quad \text{for } t_1 > 0.$$

If none of the above boundary conditions hold, the value $F(k,t_1,t_2)$ is given by the following recursive formula:

$$F(k,t_1,t_2) = \min\Biggl\{F\bigl(k-1, t_1 - p^A_k, t_2\bigr) + w^A_1 t_1;\ F(k-1,t_1,t_2) + w^A_1\left(p^B + (t_2 - t_1) + \sum_{i=1}^k p^A_i\right)\Biggr\}.$$

The optimal solution to the problem is given by

$$\min_{t_1}\bigl\{F(n,t_1,t_1)\bigr\}$$

and can therefore be computed in $O(n_A(\sum_{i=1}^{n_A} p^A_i)^2)$.

6 Computational experience

In this section, we present the results of computational experiments on the algorithms for Problems *WCWC*, *WCL*$_{\max}$ and *WCC*$_{\max}$ presented in Sects. 3, 4 and 5. The algorithms have been run on a large set of instances, for various values of n_A and n_B. Each problem and each pair (n_A, n_B) defines a scenario. For each scenario, 50 instances were run, and in Tables 1, 2 and 3 we report:

- the number of instances solved to optimality within the time limit of 1800 seconds;
- the average time required to solve such instances to optimality;
- the average number of nodes of the enumeration tree for these instances;
- the average gap $(UB - LB/LB)$ at the root node (on all instances), where LB is the Lagrangian bound at the root node and UB is the value of the heuristic solution;
- average gap after 1800 seconds (for instances not solved to optimality);
- number of instances (out of 50) in which the gap drops below 0.01 and 0.001, and the average number of enumerated nodes and time elapsed at that point.

The algorithms were implemented in the C++ language, using *Visual C++ 2005*, and run on an Intel Pentium 4 Processor, 2.6 GHz clock frequency, 1 GB RAM and *Windows XP sp2* installed.

Table 1 Computational results for *WCWC*

n_A	n_B	# proved optimal	Elapsed time	Generated nodes	Root gap	Final gap	Gap < 0.01 # inst.	Gap < 0.01 @node	Gap < 0.01 @time	Gap < 0.001 # inst.	Gap < 0.001 @node	Gap < 0.001 @time
10	10	50	0.02	500	0.003	–	50	17	0.009	50	183	0.017
10	20	50	0.03	2253	0.002	–	50	3	0.003	50	212	0.006
10	30	50	0.09	417	< 0.001	–	50	1	0.003	50	105	0.039
10	40	50	0.17	408	< 0.001	–	50	1	0.006	50	10	0.015
20	10	49	0.07	7518	0.003	0.001	50	1	< 0.001	49	413	0.007
20	20	50	0.32	89994	0.002	–	50	1	0.005	50	373	0.013
20	30	49	1.13	9469	0.001	< 0.001	50	1	0.001	50	88	0.013
20	40	48	1.23	14525	< 0.001	< 0.001	50	1	0.001	50	55	0.006
30	10	46	0.22	28564	0.001	< 0.001	50	1	0.000	48	136	0.002
30	20	49	1.73	16273	0.001	< 0.001	50	1	0.002	49	72	0.014
30	30	50	2.12	27534	0.001	–	50	1	0.007	50	46	0.011
30	40	47	2.31	49087	< 0.001	< 0.001	50	1	< 0.001	50	33	0.003
40	10	39	3.39	64908	0.001	< 0.001	50	1	0.001	45	405	0.040
40	20	37	3.7	43903	< 0.001	< 0.001	50	1	0.002	49	16	0.003
40	30	44	5.23	83479	< 0.001	< 0.001	50	1	0.001	50	14	0.003
40	40	50	5.8	583392	< 0.001	–	50	1	0.006	50	5	0.007
50	50	44	43.05	2418292	< 0.001	< 0.001	50	1	0.004	48	8	0.005
60	60	31	119.04	54364647	< 0.001	< 0.001	50	1	0.002	50	3	0.004

6.1 Problem *WCWC*

In Table 1, we report our results for 20 scenarios, generated with all processing times and weights being integers uniformly distributed in $[1\dots25]$. Concerning Q, let $f^B_{\min}$ be the value of $\sum w^B_j C^B_j$ when all B-jobs are scheduled in WSPT order at the beginning of the schedule, and let $f^B_{\max}$ be the value of $\sum w^B_j C^B_j$ when all B-jobs are scheduled in WSPT order at the end of the schedule, after all A-jobs. Clearly, if $Q < f^B_{\min}$ the problem is infeasible, and if $Q \geq f^B_{\max}$ the problem is trivial. Actually, preliminary tests showed that when Q is close to $f^B_{\min}$ or $f^B_{\max}$ the problem is easy to solve, i.e., the hardest instances are those in which Q is close to $(f^B_{\max} - f^B_{\min})/2$. For this reason, in each instance we set $Q = \alpha f^B_{\min} + (1-\alpha) f^B_{\max}$, drawing α randomly between 0.4 and 0.6.

The main observation is that our approach solves to optimality significantly large problems, up to 60 jobs per agent. We note that in all cases considered, the gap at the root node is already extremely small, indicating that the heuristic solution at the root node is optimal or near-optimal. Actually, as the number of jobs increases, the gap drops below 0.1% very soon.

It is worth noticing that, although the problem is symmetric, the fact that the two agents play different roles in the optimization model does have a consequence in terms of computational efficiency. In fact, for the same total number n of jobs, instances in which n_A is smaller appear to be easier. (For instance, notice that solving an instance with $n_A = 40$ and $n_B = 10$ takes 3.39 seconds vs. only 0.17 seconds for the opposite case.) This is probably due to the fact that only the completion times of the A-jobs directly affect the objective function.

6.2 Problem $WCL_{\max}$

In Table 2, we report our results for 11 scenarios, generated for various combinations of n_A and n_B, and again drawing all integer processing times and weights from the uniform distribution on $[1\dots25]$. Since we were able to solve all instances with $n_A \leq 30$ and $n_B \leq 30$, we also launched some experiments for larger values of n_A and n_B. Since in this problem Q is simply an offset on the due date values, we set $Q = 0$, so that $Q_j = \bar{d}^B_j$ for all B-jobs. Clearly, if for some B-job $\bar{d}^B_j < p^B_j$, the problem is infeasible, whereas, if P is the total processing time of all $n_A + n_B$ jobs, if $d^B_j \geq P$, job J^B_j can be removed. For this reason, the due dates were randomly generated from a uniform distribution on $[0.1P, 0.9P]$.

From Table 2, this problem appears more difficult than the previous one. While all instances were solved to optimality up to $n_A = 30$ and $n_B = 30$, the number of certified optimal solutions decreases in larger scenarios. However, in *all* instances the final gap is below 1%, with an average of 0.6% on largest instances. Note that even in largest instances, in most cases (84%) a solution at most 0.1% away

Table 2 Computational results for $WCL_{\max}$

n_A	n_B	# proved optimal	Elapsed time	Generated nodes	Root gap	Final gap	Gap < 0.01 # inst.	Gap < 0.01 @node	Gap < 0.01 @time	Gap < 0.001 # inst.	Gap < 0.001 @node	Gap < 0.001 @time
10	10	50	0.01	367	0.089	–	50	20	0.001	50	298	0.006
10	20	50	0.02	312	0.115	–	50	57	0.003	50	266	0.010
10	30	50	0.03	215	0.117	–	50	46	0.003	50	168	0.012
20	10	50	0.26	8679	0.028	–	50	1	0.000	50	2918	0.090
20	20	50	0.84	19814	0.045	–	50	34	0.001	50	10008	0.407
20	30	50	1.08	26478	0.053	–	50	2	< 0.001	50	14977	0.564
30	10	50	2.68	68657	0.013	–	50	1	< 0.001	50	4311	0.173
30	20	50	7.24	187377	0.022	–	50	1	< 0.001	50	29096	1.159
30	30	50	23.79	615546	0.036	–	50	1	< 0.001	50	169654	6.732
40	40	19	95.22	1930592	0.029	0.006	50	1	< 0.001	42	184994	9.244
50	50	4	113.43	1863063	0.022	0.006	50	1	< 0.001	42	107249	6.498

Table 3 Computational results for $WCC_{\max}$

n_A	n_B	# proved optimal	Elapsed time	Generated nodes	Root gap	Final gap	Gap < 0.01 # inst.	Gap < 0.01 @node	Gap < 0.01 @time	Gap < 0.001 # inst.	Gap < 0.001 @node	Gap < 0.001 @time
10	10	50	0.01	512	0.009	–	50	3	< 0.001	50	10	0.002
20	20	50	0.09	481	0.01	–	50	1	< 0.001	50	22	0.001
30	30	50	0.47	21874	0.004	–	50	1	< 0.001	50	4	0.001
40	40	49	4.54	189464	0.008	<0.001	50	2	0.016	50	121	0.23
50	50	47	7.68	389926	0.004	<0.001	50	1	0.003	50	2	1.002
60	60	39	23.43	11933565	0.006	<0.001	50	1	< 0.001	48	103	12.006

from optimality is available after a few seconds of computation. It is also worth noticing that in all instances (and more so for large n) the gap drops below the 1% threshold very quickly, most often at the root node. This indicates that the Lagrangian bound is indeed strong.

Similarly to the previous problem, the computation time is more sensitive to the growth of n_A than of n_B. In fact, this is more apparent here than for *WCWC*, which is not surprising since the two agents have asymmetric objectives here. It is interesting to note that, on the contrary, the quality of the bound at the root node (slightly) decreases as n_B increases. This is probably due to the fact that the number of multipliers increases with n_B.

6.3 Problem $WCC_{\max}$

For this problem we generated 6 scenarios, with the parameters generated as for the previous problems. Recall that the B-jobs are scheduled consecutively in any optimal schedule. If their total length P^B is very small, the WSPT rule is likely to produce an optimal solution. On the other hand, if P^B is very large, the problem becomes similar to a classical knapsack, since the problem is then to maximize the total weight of the A-jobs scheduled from 0 to $Q - P^B$. We therefore generated "balanced" instances, i.e., with $n_A = n_B$. Concerning Q, we let $Q = \alpha(P^A + P^B) + P^B/2$, where α is drawn randomly from the interval $[0.4, 0.6]$. This choice corresponds to locating the B-jobs roughly in the middle of the schedule, hence dividing the A-jobs into two sets having similar total duration.

From Table 3, we see that our algorithm is able to optimally solve almost all instances up to 50 A-jobs. Moreover, in all instances the final gap is below 0.1%. The quality of the bound at the root node is already very high, which confirms the fact that this problem is indeed easier than $WCL_{\max}$.

7 Conclusions

In this paper, we addressed the problem of generating meaningful schedules for two agents, A and B, who have to negotiate the usage of a common single machine. In particular, we focused our attention on the problem of finding an optimal schedule for agent A subject to the constraint that the cost for agent B does not exceed a given value. We ad-

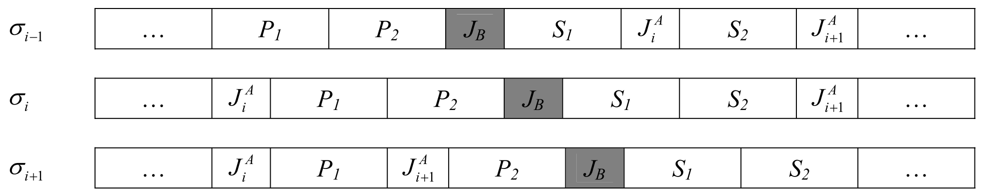

Fig. 7 Structure of three consecutive optimal schedules

dressed three such scenarios, all NP-hard from the complexity viewpoint, different for the cost functions of agent B.

We devised a Lagrangian approach for which, in all cases, we were able to solve the Lagrangian dual very efficiently by ad hoc algorithms. Our approach allows to solve instances of significant size, always providing optimal or near-optimal solutions. The hardest case is when agent B wants to minimize the maximum lateness of its jobs, while the other two cases appear to be easier, also because in these cases the Lagrangian relaxation involves a single multiplier.

Several venues for future research are open, including the following:

- Different bounding schemes for the same two-agent scheduling problems addressed here. This would probably require very different approaches, since other classical bounding schemes are not effective in this context (e.g., preemption does not help).
- Extension to more than two agents. Note that such a generalization is straightforward if A wants to minimize $\sum w_i^A C_i^A$ and the maximum lateness or makespan for *all* other agents is bounded, whereas it is less obvious if some of the other agents also want to minimize total weighted completion time.
- Extension of the Lagrangian approach to devise exact algorithms for bicriteria, single-agent scheduling problems. In particular, computational experiments to test the approach in Sect. 3.2 for the bicriteria version of *WCWC* should be carried out. Also, it may be worth investigating whether the Lagrangian approach of Sect. 4 for $WCL_{\max}$ can be extended to the bicriteria problem $1|\tilde{d}_j|\sum w_j C_j$, either to devise an exact or a heuristic algorithm.

Appendix: Proofs

In this section, we give the proofs for results given in previous sections.

Proof of Theorem 4.5 For the sake of simplicity, we assume that, at any step of PBS, the next two gaps are separated by a single B-job J_j^B, and that the total size of these two gaps exceeds the length of the current job J_i^A to be scheduled. Indicate by $[T, Q_j - p_j^B]$ the next gap. Two cases can occur.

(i) If $Q_j - p_j^B - T \geq p_i^A$, PBS schedules the entire job J_i^A, starting at T. In this case, also LDA schedules J_i^A at the same starting time. So, the contribution to the value of the bound is the same in both schemes.

(ii) If $Q_j - p_j^B - T < p_i^A$, the two schemes give different contributions. PBS splits J_i^A into J' and J'', of total length $p' + p'' = p_i^A$, that will complete respectively at $C' = T + p'$ and $C'' = T + p_i^A + p_j^B$. Hence, the contribution of J' and J'' to Posner's bound is given by

$$\delta_i^A p'C' + \delta_i^A p''C'' = \delta_i^A\left(p_i^A T + p'^2 + p''^2 + p'p'' + p''p_j^B\right). \tag{30}$$

Now turn to LDA. In this case, LDA moves J_j^B backward, to start at T, so that $C_j^B = T + p_j^B$, and assigns to J_j^B the density $\delta_j^B = \delta_i^A$. Thereafter, J_i^A is scheduled to start at $T + p_j^B$, so $C_i^A = T + p_j^B + p_i^A$. Hence, recalling that $T + p' + p_j^B = Q_j$, the contribution of J_i^A and J_j^B to the Lagrangian dual bound is given by

$$\delta_i^A p_i^A\left(T + p_i^A + p_j^B\right) + \delta_i^A p_j^B\left(T + p_j^B - Q_j\right) = \delta_i^A\left(p_i^A T + p'^2 + p''^2 + 2p'p'' + p''p_j^B\right). \tag{31}$$

Subtracting (30) from (31), we get $\delta_i^A p'p'' > 0$, proving the first part of the thesis.

To prove the second part, one only needs to note that the two bounds coincide if and only if case (ii) never occurs, i.e., no A-job is ever split during PBS (and hence each B-job J_j^B completes at Q_j). In this case, PBS finds a feasible and hence optimal solution. Finally, dropping the initial assumptions, very similar arguments apply. □

Proof of Lemma 5.2
From Figs. 5 and 7, in which

- P_1 is the set of A-jobs belonging to P having density in $[w_{i+1}^A/p_{i+1}^A, w_i^A/p_i^A]$,
- P_2 is the set of A-jobs belonging to P having density less than w_{i+1}^A/p_{i+1}^A,
- S_1 is the set of A-jobs belonging to S having density greater than w_{i+1}^A/p_{i+1}^A,
- S_2 is the set of A-jobs belonging to S having density in $[w_{i+1}^A/p_{i+1}^A, w_i^A/p_i^A]$,

we can establish the following expressions for $\lambda_{i-1,i}$ and $\lambda_{i,i+1}$:

$$\lambda_{i-1,i} = \delta_i^A \Bigg(p^B + \sum_{P_1,P_2,S_1} p_k^A \Bigg) - \Bigg(\sum_{P_1,P_2,S_1} w_k^A \Bigg),$$

$$\lambda_{i,i+1} = \delta_{i+1}^A \Bigg(p^B + \sum_{P_2,S_1,S_2} p_k^A \Bigg) - \Bigg(\sum_{P_2,S_1,S_2} w_k^A \Bigg),$$

from which one has:

$$\begin{aligned}
&\lambda_{i-1,i} - \lambda_{i,i+1} \\
&\quad = \delta_i^A \Bigg(p^B + \sum_{P_1,P_2,S_1} p_k^A \Bigg) - \delta_{i+1}^A \Bigg(p^B + \sum_{P_2,S_1,S_2} p_k^A \Bigg) \\
&\qquad + \Bigg(\sum_{P_2,S_1,S_2} w_k^A \Bigg) - \Bigg(\sum_{P_1,P_2,S_1} w_k^A \Bigg) \\
&\quad = \left(\delta_i^A - \delta_{i+1}^A\right) \Bigg(p^B + \sum_{P_2,S_1} p_k^A \Bigg) + \Bigg(\sum_{S_2} w_k^A - \sum_{P_1} w_k^A \Bigg) \\
&\qquad + \Bigg(\delta_i^A \sum_{P_1} p_k^A - \delta_{i+1}^A \sum_{S_2} p_k^A \Bigg).
\end{aligned}$$

Omitting the first term of last member, which is non-negative, we get:

$$\begin{aligned}
&\lambda_{i-1,i} - \lambda_{i,i+1} \\
&\quad \geq \Bigg(\delta_i^A \sum_{P_1} p_k^A - \sum_{P_1} w_k^A \Bigg) - \Bigg(\delta_{i+1}^A \sum_{S_2} p_k^A - \sum_{S_2} w_k^A \Bigg) \\
&\qquad + \sum_{P_1} \left(\delta_{i+1}^A p_k^A - w_k^A\right) - \sum_{S_2} \left(\delta_{i+1}^A p_k^A - w_k^A\right).
\end{aligned}$$

The first sum in the last member is non negative while the second is non positive. In fact, due to the structure of schedules σ_{i-1} and σ_i, the following holds:

$$\delta_i^A \geq \delta_k^A = \frac{w_k^A}{p_k^A} \quad \Rightarrow \quad \delta_i^A p_k^A - w_k^A \geq 0, \quad k \in P_1,$$

$$\delta_i^A \leq \delta_k^A = \frac{w_k^A}{p_k^A} \quad \Rightarrow \quad \delta_i^A p_k^A - w_k^A \leq 0, \quad k \in S_2.$$

Hence, the thesis holds. □

References

Albino, V., Carbonara, N., & Giannoccaro, I. (2006). Innovation in industrial districts: an agent-based simulation model. *International Journal of Production Economics*, *104*, 30–45.

Arbib, C., Servilio, M., & Smriglio, S. (2004). A competitive scheduling problem and its relevance to UMTS channel assignment. *Networks*, *44*(2), 132–141.

Agnetis, A., Mirchandani, P. B., Pacciarelli, D., & Pacifici, A. (2000). Nondominated schedules for a job-shop with two competing agents. *Computational and Mathematical Organization Theory*, *6*(2), 191–217.

Agnetis, A., Mirchandani, P. B., Pacciarelli, D., & Pacifici, A. (2004). Scheduling problems with two competing agents. *Operations Research*, *52*(2), 229–242.

Baker, K. R., & Cole Smith, J. (2003). A multiple-criterion model for machine scheduling. *Journal of Scheduling*, *6*(1), 7–16.

Cheng, T. C. E., Ng, C. T., & Yuan, J. J. (2006). Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs. *Theoretical Computer Science*, *362*, 273–281.

Cheng, T. C. E., Ng, C. T., & Yuan, J. J. (2008). Multi-agent scheduling on a single machine with max-form criteria. *European Journal of Operational Research*, *188*, 603–609.

Ng, C. T., Cheng, T. C. E., & Yuan, J. J. (2006). A note on the complexity of the problem of two-agent scheduling on a single machine. *Journal of Combinatorial Optimization*, *12*(4), 387–394.

Pan, Y. (2003). An improved branch and bound algorithm for single machine scheduling with deadlines to minimize total weighted completion time. *Operations Research Letters*, *31*, 492–496.

Peha, J. M. (1995). Heterogeneous-criteria scheduling: minimizing weighted number of tardy jobs and weighted completion time. *Journal of Computers and Operations Research*, *22*(10), 1089–1100.

Posner, M. E. (1985). Minimizing weighted completion times with deadlines. *Operations Research*, *33*(3), 562–574.

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

Sourd, F. (2008). Private communication.