Scheduling the Tasks of Two Agents with a Central Selection Mechanism

Gaia Nicosia
Dipartimento di Ingegneria
Università degli Studi Roma Tre, Italy, nicosia@dia.uniroma3.it

Andrea Pacifici
Dipartimento di Ingegneria Civile e Ingegneria Informatica
Università degli Studi di Roma "Tor Vergata", Italy, pacifici@disp.uniroma2.it

Ulrich Pferschy
Department of Statistics and Operations Research
University of Graz, Austria, pferschy@uni-graz.at

Abstract

We address a class of deterministic scheduling problems in which two agents compete for the usage of a single machine. The agents have their own objective functions and submit in each round an arbitrary, unprocessed task from their buffer for possible selection. In each round the smaller of the two submitted tasks is chosen and processed on the machine.

We consider the problems under two distinct perspectives: First, we look at them from a centralized point of view as bicriteria optimization problems and try to characterize the set of Pareto optimal solutions. Then, the problems are viewed under the perspective of a single agent. In particular, we measure the performance of simple priority rules suggesting to an agent how to sequence its own tasks in order to optimize its own objective function. Then we consider *minimax* strategies, i.e. algorithms optimizing the objective of one agent in the worst case.

Keywords: scheduling, multi-agent optimization, bicriteria optimization.

1 Introduction

In *multi-agent scheduling* problems a set of *tasks* has to be processed on some processing *resource* and each task belongs to one decider (agent). Each agent is interested in optimizing its own performance measure which only depends on its tasks completion times. Although these problems can be viewed as a special case of multi-objective scheduling problems [14], their specific properties and applications have spurred a considerable amount of research since the seminal works by Agnetis et al. [1] and Baker and Smith [5]. Most of the recent literature on multi-agent scheduling problems falls into two main streams of research: one focuses on the problem in a multi-objective optimization perspective (see, for example, [9, 16]); the other is from the algorithmic game theory point of view. In the latter context, for instance, *mechanism design* has received considerable attention in the recent literature (see, e.g. [4, 8, 12]). The goal is to design system-wide rules which, given the selfish decisions of the users, maximize the total social welfare. The degree to which these rules approximate the social welfare in a worst-case equilibrium is known as the price of anarchy of the mechanism.

In this work we address a multi-agent scheduling problem introduced in [2]: Two agents, each owning a set of nonpreemptive tasks (or jobs), require a single (commonly used) machine to process their tasks. Each agent pursues the minimization of a given objective function, such as makespan, total completion time or total weighted completion time. Additionally, a coordination mechanism, aiming at the maximization of the number of processed tasks per time unit, regulates access of agents' tasks to the machine as follows: Each agent submits one task, if available, for possible processing. The shortest among the two submitted tasks is selected and scheduled at the end of the current schedule, which is initially empty. When all tasks of one agent have been processed, the remaining tasks of the other agents are appended thereafter in the order they are submitted. In the following we refer to the above steps as *rounds*. We also say that the selected task in one round (and the corresponding agent) is the *winner*, while the other task (and agent) is the *loser*.

We look at the problem in two different settings. In a *centralized perspective* we aim at characterizing the set of Pareto optimal schedules in terms of size and computational complexity, as in a bicriteria optimization problem. Then, we consider the problem from a single *agent perspective* and search for a *strategy*, i.e., an algorithm that suggests to the agent which task to submit in each round, taking into account its own objective function. In this context we study the effectiveness of natural priority rules, e.g., when the agent submits its tasks in SPT or WSPT order. Moreover, we consider the case in which one agent wants to select a strategy that minimizes its solution cost in the worst possible case, i.e., for any strategy adopted by the opponent. This corresponds to what is usually called *minimax strategy* in game theory.

This twofold approach has also been adopted in [2, 3, 10, 11]. In particular, in [2] the authors introduce a class of multi-agent scheduling problems in which the decision process is organized in rounds and provide some preliminary results for different shop configurations. A detailed analysis of the so-called *linear conveyor* shop configuration is carried out in [3], where a number of properties and solution algorithms are presented taking into account both centralized and single-agent perspectives. The shop configuration of [3] refers to a manufacturing application in which two linear conveyor belts, one for each agent, transport parts to the machine. So, each agent sequences the parts on the conveyor, implying that, at each round, one of the two candidate tasks is the loser of the preceding round. In other words, each task is submitted for possible processing, in the given order, until it wins.

In this paper we consider a different, more general configuration (denoted as *flexible processing* in [2]) in which there are no queues at the machine and at each round any part from the two agents' buffers can be picked up and submitted for possible processing. Hence, in this case, the agents are free to choose any available task for submission at each round, independently from the outcome of the previous round. In particular, the two shop configurations (linear and flexible) produce different sets of feasible schedules. Note that in [2] part of the results discussed in this paper have been already mentioned.

The paper is organized as follows. In Section 2 we formally introduce the problem, present the notation, and summarize all the results of the paper in two tables. Section 3 addresses the problem of characterizing the set $\mathcal{P}$ of Pareto optimal solutions for various objective functions. In particular, we investigate the complexity of finding Pareto optima and determine their number. In Section 4 we consider a single agent perspective and provide results on the worst-case performance of various heuristic strategies. Section 5 deals with the problem of devising minimax strategies for one agent. Finally, some conclusions are drawn in Section 6.

2 Notation and summary of results

Let A and B denote the two agents. Each agent owns a set of n nonpreemptive tasks to be performed on a single machine which can process only one task at a time. Tasks have nonnegative deterministic processing times $a_1 \leq a_2 \leq \ldots \leq a_n$ for agent A and $b_1 \leq b_2 \leq \ldots \leq b_n$ for agent B. For convenience, we will often refer to tasks by their processing times. Sometimes each task also has a *weight* indicating its importance. We will only need explicit weight values for agent B and thus define a weight w_j for each task b_j, $j = 1, \ldots, n$. All data are known by both agents and all tasks are available at the beginning of the planning process.

Each agent wants to optimize its own objective function, which only depends on the completion times of its tasks: $f^A = f^A(C_1^A, \ldots, C_n^A)$ and $f^B = f^B(C_1^B, \ldots, C_n^B)$, where C_j^X is the completion time of task j of agent X ($j = 1, \ldots, n$, $X = A, B$). In this paper we consider the minimization of (i) makespan $f^X = \max\{C_1^X, \ldots, C_n^X\}$, ($ii$) total completion time $f^X = \sum_{j=1}^n C_j^X$, and (iii) total weighted completion time, e.g. $f^B = \sum_{j=1}^n w_j C_j^B$. We denote by (f^A, f^B) the problem where f^A and f^B are the two agents' objective functions.

The decision process is divided into $2n$ rounds each consisting of the following two steps.

1. Each agent submits, if possible, one of its unscheduled tasks.
2. The *shortest*[1] among the two submitted tasks, or the single submitted task, is selected and scheduled at the end of the current schedule.

Note that not all sequences of tasks can be the output of the above described process. In particular, a schedule with task x in position i is *feasible* as long as there is a task of the opponent agent with processing time larger than x which is unscheduled until round i.

In the following tables we give an overview of our results presented in Sections 3 and 4. In particular, in Table 1 the results concerning the centralized perspective are given: in the second column (Size of PO set) we report, for each addressed problem, an estimate on the number of Pareto optima. In the third column (Complexity of recognition problem) we indicate the complexity of the problem of deciding whether a feasible schedule with given bounds on the agents objectives exists.

(f^A, f^B)	Size of PO set	Complexity of recognition problem
$(C_{\max}^A, C_{\max}^B)$	1	trivial (Section 3.1)
$(C_{\max}^A, \sum_j C_j^B)$	$O(n)$	trivial (Section 3.1)
$(C_{\max}^A, \sum_j w_j C_j^B)$	exponential	$\mathcal{NP}$-complete (Section 3.1)
$(\sum_j C_j^A, \sum_j C_j^B)$	exponential	$\mathcal{NP}$-complete (Section 3.2)

Table 1: Summary of results for the centralized perspective (Section 3).

Table 2 presents the worst case performance ratios of SPT or WSPT algorithms when used by agent B against any strategy of agent A. Note that the term "QRO" indicates that agent A is restricted to a certain set of "reasonable" strategies, see Section 4.3, Definition 1.

Regarding the results on minimax strategies, in Section 5 we show that SPT is a minimax strategy for agent B when its objective is the minimization of the makespan or total completion time. On the other hand, finding a minimax strategy when $f^B = \sum_j w_j C_j^B$ turns out to be non trivial but can be done in polynomial time.

[1]Without loss of generality, we may assume that, in case of ties, agent A wins.

(f^A, f^B)	H	$\rho(H)$
$(f^A, C^B_{\max})$	SPT	$\rho(SPT) = n$ (Section 4.1)
$(f^A, \sum_j C^B_j)$	SPT	$\rho(SPT) = n$ (Section 4.2)
$(\text{QRO}, \sum_j C^B_j)$	SPT	$\frac{\sqrt{2}}{2}\sqrt{n} \le \rho(SPT) \le \frac{\sqrt{2}}{2}\sqrt{n} + 1$ (Section 4.3)
$(f^A, \sum_j w_j C^B_j)$	SPT	$\rho(SPT) = 2n - 1$ (Section 4.4)
$(f^A, \sum_j w_j C^B_j)$	WSPT	$\rho(WSPT) = n$ (Section 4.5)

Table 2: Performance ratios for standard sequencing rules (Section 4).

Throughout the paper we will frequently put our results in perspective to the results derived by Agnetis et al. [3] for the more restricted variant of this problem, where each task is submitted for possible processing in a fixed order until it wins.

3 Centralized perspective

In this section, we address the problem of characterizing the set $\mathcal{P}$ of Pareto optimal solutions for various objective functions. The results in the remainder of this section bear some resemblance to their counterparts of [3] where the linear conveyor case is addressed.

3.1 Problems with makespan minimization

First assume that both agents want to minimize their makespan, i.e., consider the pair of objective functions $(C^A_{\max}, C^B_{\max})$. Since the largest task can never win against any opponent's task, the owner of the largest task will always have a makespan of $\sum_j a_j + \sum_j b_j$. Hence, we have the following:

Proposition 1 *For problem $(C^A_{\max}, C^B_{\max})$ there is only one Pareto optimal schedule.*

In particular, this single Pareto optimum is obtained when the agent who does not own the largest task wins the first n rounds (regardless of its submission sequence). The resulting schedule, when A owns the largest task, has the following structure:

$$\langle b_{i1}, \ldots, b_{in}, a_{i_1}, a_{i_2}, \ldots, a_{i_n} \rangle.$$

If agent A wants to minimize its makespan, while agent B has the objective of minimizing the total completion time, the problem is still easy and we have the following:

Proposition 2 *For problem $(C^A_{\max}, \sum_j C^B_j)$ there are at most n Pareto optimal schedules.*

Proof. Since A's objective is the makespan minimization and therefore depends only on the completion time of the last task, there is a Pareto optimal solution having all A-tasks consecutively scheduled, i.e. in a single block.

Consider first the case in which $a_n > b_n$. In this case A cannot avoid to reach $C^A_{\max} = \sum_j a_j + \sum_j b_j$ since all B-tasks will be processed before a_n. Hence, there is only *one* Pareto optimal solution, as above, e.g. obtained when A submits a_n until all B-tasks are scheduled in SPT order and then submits its tasks in arbitrary order. The solution is clearly optimal for B since the B-tasks are scheduled in SPT order.

If, on the other hand, $a_n \le b_n$, let $t := \min\{j \mid b_j \ge a_n\}$. Hence, B-tasks $b_t, \ldots, b_n$ are always scheduled after all A-tasks. So, for any $k = 0, 1, \ldots, t-1$, we obtain a Pareto optimal schedule by scheduling the first k B-tasks before the A-block and all remaining B-tasks at the end of the schedule. Hence, a PO solutions has the following structure (see Figure 1 for an example):

$$\langle b_1, \dots, b_k, a_{i_1}, a_{i_2}, \dots, a_{i_n} b_{k+1}, \dots, b_{t-1}, b_t, \dots, b_n \rangle$$

Clearly, any deviation of B from the SPT ordering would only worsen its objective, as can be shown by a simple exchange argument. All together, there are exactly $t \leq n$ Pareto optimal schedules. $\square$

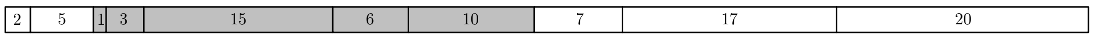

Figure 1: Example of a Pareto optimal solution for $(C^A_{\max}, \sum_j C^B_j)$ with $k = 2$ and $t = 4$, shaded tasks belong to A.

When agent B wants to minimize its total weighted completion time while A's objective is still makespan minimization we have the following results. We omit the proofs here, since they are fairly similar to those introduced in [3]. For the reader's benefit the proofs are reported in the Appendix.

Proposition 3 *There can be exponentially many PO solutions for problem* $(C^A_{\max}, \sum w_j C^B_j)$.

Proposition 4 *Given two integers* Q^A *and* Q^B*, it is* $\mathcal{NP}$*-complete to decide if there exists a feasible schedule* σ *of* $(C^A_{\max}, \sum w_j C^B_j)$ *with* $C^A_{\max}(\sigma) \leq Q^A$ *and* $\sum w_j C^B_j(\sigma) \leq Q^B$.

3.2 Problems minimizing the sum of completion times

In this section we consider the case where both agents want to minimize their sum of completion times.

Theorem 5 *There can be exponentially many Pareto optimal solutions for* $(\sum C^A_j, \sum C^B_j)$.

Proof. Consider the following instance POexp. Let M be a large enough number. Each agent owns n tasks, with the following values $a_i = M^i$ and $b_i = 1 + M^i$ for $i = 1, \dots, n$. Clearly, b_n is the largest task and hence it will be scheduled last.

We call *doubleton* D_i the pair of tasks a_i, b_i, and consider only schedules in which the tasks of D_i are all scheduled before the tasks of D_{i+1}, for all $i = 1, \dots, n-1$. Let $\mathcal{D}$ indicate the set of such feasible schedules and obviously, $|\mathcal{D}| = 2^{n-1}$ since in D_n task a_n must precede b_n. We now show that all these schedules are nondominated.

Consider a schedule $\sigma \in \mathcal{D}$. Let $S \subseteq \{1, \dots, n\}$ be the index set of doubletons in which a_i precedes b_i and $\bar{S} = \{1, 2, \dots, n\} \setminus S$. We denote by P_i the starting time of doubleton D_i, i.e. $P_i = \sum_{j=1}^{i-1}(a_j + b_j)$. The objective function values for the two agents are:

$$f^A(\sigma) = \sum_{i \in S}(P_i + a_i) + \sum_{i \in \bar{S}}(P_i + a_i + b_i) = \sum_{i=1}^{n}(P_i + a_i) + \sum_{i \in \bar{S}} b_i \tag{1}$$

$$f^B(\sigma) = \sum_{i \in \bar{S}}(P_i + b_i) + \sum_{i \in S}(P_i + a_i + b_i) = \sum_{i=1}^{n}(P_i + b_i) + \sum_{i \in S} a_i \tag{2}$$

First we show that schedules of $\mathcal{D}$ are not dominated by any other schedule not in $\mathcal{D}$. To do so we compare the total cost of schedules, i.e. $f^A + f^B$. Clearly, the optimal total cost f^* is obtained by scheduling all tasks by SPT, i.e. when $\bar{S} = \emptyset$.

Now we bound the difference between f^* and the total cost of the worst possible schedule in $\mathcal{D}$, call it $\hat{\sigma}$, which is obtained by letting b_i precede a_i in all D_i. A trivial calculation yields $f^A(\hat{\sigma}) + f^B(\hat{\sigma}) - f^* \leq n$. On the other hand considering the combined sequence of tasks of both agents any schedule not in $\mathcal{D}$ must have at least one position where a task from some doubleton D_{i+p} is followed by a task from a doubleton D_i with $p \geq 1$. Calculating the difference of this configuration from an optimal SPT schedule with respect to the total completion times we get in the best case for the schedule not in $\mathcal{D}$, i.e. $p = 1$, a deviation of $2(M^{i+1} - M^i)$ which is strictly larger than n for M sufficiently large. As a consequence, no schedule of $\mathcal{D}$ is dominated by any other schedule not in $\mathcal{D}$.

Hereafter we show that no two schedules in $\mathcal{D}$ dominate each other. Consider two *distinct* schedules σ_1, σ_2 in $\mathcal{D}$. Let k be the *largest* index of a doubleton such that D_k is processed in different modes in σ_1 and σ_2, and suppose w.l.o.g. that in σ_1 a_k precedes b_k (and viceversa in σ_2). From (1) and (2), even if b_i precedes a_i in all doubletons of σ_1 preceding D_k, and viceversa for σ_2, there is:

$$f^A(\sigma_2) - f^A(\sigma_1) \geq b_k - \sum_{i=1}^{k-1} b_i \geq M^k - \frac{M^k - 1}{M - 1} - k + 2$$

$$f^B(\sigma_1) - f^B(\sigma_2) \geq a_k - \sum_{i=1}^{k-1} a_i \geq M^k - \frac{M^k - 1}{M - 1}$$

Hence, switching the ordering of a_k and b_k, i.e. moving from σ_1 to σ_2, implies $f^B(\sigma_2) < f^B(\sigma_1)$ and $f^A(\sigma_2) > f^A(\sigma_1)$ for large enough M. This shows that any two schedules in $\mathcal{D}$ do not dominate each other. The thesis follows. $\square$

This result can be trivially extended to show that there are also exponentially many Pareto optimal solutions when one or both of the two agents minimizes its total *weighted* completion times.

The following theorem shows that it is hard to determine whether a feasible solution σ of $(\sum_j C_j^A, \sum_j C_j^B)$ is Pareto-optimal. We prove it by showing that the problem of finding another feasible solution σ' that dominates σ (i.e., with $f^A(\sigma') \leq f^A(\sigma)$, $f^B(\sigma') \leq f^B(\sigma)$ with at least one strict inequality) is an $\mathcal{NP}$-complete problem.

Theorem 6 *Given two nonnegative integers Q^A and Q^B, it is $\mathcal{NP}$-complete to decide whether a feasible solution σ of $(\sum_j C_j^A, \sum_j C_j^B)$ exists such that $\sum_j C_j^A(\sigma) \leq Q^A$ and $\sum_j C_j^B(\sigma) \leq Q^B$.*

Proof. We proceed similarly to Theorem 9.2 of [1] and apply a reduction from the classical $\mathcal{NP}$-complete Partitioning problem.

PARTITION
Instance: A set of n integers, $I = \{a_1, a_2, \ldots, a_n\}$.
Question: Is there a bipartition $(S, \bar{S})$ of I such that $\sum_{i \in S} a_i = \sum_{i \in \bar{S}} a_i$?

Given an instance I of PARTITION where, with no loss of generality $a_1 \leq a_2 \leq \ldots \leq a_n$, we define an instance I' of $(\sum_j C_j{}^A, \sum_j C_j{}^B)$ as follows. A and B have $n + 1$ tasks each and the processing times $a_1, \ldots, a_n$ of the first n A-tasks are equal to the n items of PARTITION while

$a_{n+1} = M$ is a very large value. B-tasks have processing times equal to their A counterparts: $b_i = a_i$. We let $P = \sum_{i=1}^{n} a_i$ and choose $Q_A = 2(\sum_{i=1}^{n}(n-i)a_i) + \frac{7}{2}P + M$, $Q_B = Q_A + M$.

Suppose I is a yes-instance of Partition and $S \subset \{1, \ldots, n\}$ such that

$$\sum_{i \in S}^{n} a_i = \sum_{i \in \bar{S}}^{n} a_i = \frac{1}{2}P. \tag{3}$$

Consider a schedule σ of I' organized in $n+1$ doubletons as in Theorem 5 (i.e., tasks a_i, b_i must precede a_j, b_j for all $1 \leq i < j \leq n+1$). In σ, a_i precedes b_i for all $i \in S$ and, conversely, b_i precedes a_i for all $i \in \bar{S}$. Obviously, since in case of a tie A is the winner, in the last doubleton a_{n+1} must precede b_{n+1}. Let us now compute the objective values for the two agents. From equations (1) and (2), recalling that $b_i = a_i$ and thus $P_i = 2\sum_{j=1}^{i-1} a_j$, we obtain:

$$f^A(\sigma) = \sum_{i=1}^{n}(P_i + a_i) + \sum_{i \in \bar{S}} a_i + (P_{n+1} + a_{n+1}) = 2\left(\sum_{i=1}^{n}(n-i)a_i\right) + P + \sum_{i \in \bar{S}}^{n} a_i + (2P + M) \tag{4}$$

$$f^B(\sigma) = \sum_{i=1}^{n}(P_i + a_i) + \sum_{i \in S} a_i + (P_{n+1} + 2a_{n+1}) = 2\left(\sum_{i=1}^{n}(n-i)a_i\right) + P + \sum_{i \in S}^{n} a_i + (2P + 2M) \tag{5}$$

Note the latter terms of (4) and (5) referring to the completion times of the tasks in the last doubleton. The schedule σ is feasible and, since I is a yes-instance and equation (3) holds, then $f^A(\sigma) = Q^A$, $f^B(\sigma) = Q^B$, i.e., there exists a yes-instance I' of $(\sum_j {C_j}^A, \sum_j {C_j}^B)$.

We now show that any schedule $\bar{\sigma}$ satisfying $\sum_j C_j^A(\bar{\sigma}) \leq Q^A$ and $\sum_j C_j^B(\bar{\sigma}) \leq Q^B$ must be organized in doubletons. First note that if a schedule σ minimizes the sum of the agents' objectives $f^A(\sigma) + f^B(\sigma)$, then tasks in σ are sequenced according to an SPT order, that is, σ is arranged in doubletons[2] and $f^A(\sigma) + f^B(\sigma) = Q^A + Q^B$. Suppose that $\bar{\sigma}$ is not arranged in doubletons. Then, there must be at least two consecutive tasks in $\bar{\sigma}$, x preceding y, such that $x > y$. Now if we swap the two tasks, we obtain a new schedule such that the *overall* total completion time has decreased by a strictly positive amount $x - y$. By repeatedly applying the above swaps, we eventually find a solution σ^* arranged in doubletons, such that $\sum_j C_j^A(\sigma^*) + \sum_j C_j^B(\sigma^*) < \sum_j C_j^A(\bar{\sigma}) + \sum_j C_j^B(\bar{\sigma})$. Hence, we reach the contradiction $Q^A + Q^B = \sum_j C_j^A(\sigma^*) + \sum_j C_j^B(\sigma^*) < \sum_j C_j^A(\bar{\sigma}) + \sum_j C_j^B(\bar{\sigma}) \leq Q^A + Q^B$, by definition of $\bar{\sigma}$.

Therefore only schedules arranged in doubletons can be feasible. This implies that (4) and (5) hold and, in particular, that $f^A(\sigma) = Q^A$ and $f^B(\sigma) = Q^B$. This in turn implies that $\sum_{i \in S}^{n} a_i = \sum_{i \in \bar{S}}^{n} a_i = \frac{1}{2}P$, i.e. I is a yes instance of Partition. Membership of $(\sum_j C_j^A, \sum_j C_j^B)$ in NP is trivial and this completes the proof. $\square$

Clearly, the latter result implies also the $\mathcal{NP}$-completeness of the same problem when at least one of the two agents wants to minimize its total *weighted* completion times.

4 Single agent perspective

Differently from the centralized perspective, most of the results for the linear conveyor architecture presented in [3] do not extend to the shop configuration considered in this paper. In particular, hereafter, we evaluate the quality of the schedule B attains, when its tasks are sequenced applying a certain standard, single-agent heuristic rule H, compared to the optimal

[2]We avoid the easy but tedious details concerning the case in which $a_i = a_{i+1}$ for some i.

schedule for agent B, when B's objective is the minimization of makespan, total completion time, and total weighted completion time.

Of course, the outcome of a heuristic of B and the optimal schedule of B both depend crucially on the strategy pursued by A. In this setting, we use the term *strategy* as it is known in game theory. It refers to an algorithm that outputs in any round a task to be submitted depending on all submissions of both agents in all previous rounds. We will study the case where agent B follows a simple heuristic rule H to determine its strategy, while A can use any strategy.

In this setting, let Ω^A (respectively Ω^B) be the set of all possible strategies for agent A (respectively B). For every pair of strategies $s^A \in \Omega^A, s^B \in \Omega^B$, let $\sigma(s^A, s^B)$ denote the schedule obtained when agent X, $X = A, B$, adopts a strategy s^X. In the following we analyze the value of the performance ratio

$$\rho(H) \leq \max_{s^A \in \Omega^A} \frac{f^B(\sigma(s^A, H))}{\min_{s^B \in \Omega^B}\{f^B(\sigma(s^A, s^B))\}}. \tag{6}$$

As usual, we call a performance bound $\rho(H)$ *tight*, if no larger value than $\rho(H)$ exists which fulfills (6). In the next sections we provide bounds for $\rho(H)$ when H is either the shortest processing time first (SPT) heuristic (losing tasks are submitted until they win as in the linear conveyor case) or the weighted shortest processing time (WSPT) rule.

In [3] it was shown that, in the linear conveyor setting where both agents submit their tasks in a fixed order, SPT is an optimal strategy for B when minimizing makespan or total completion time. In our setting where each agent is allowed to possibly withdraw a losing task and submit any unprocessed task, such flexibility of the opponent agent A considerably worsens the performance of SPT heuristic for B.

4.1 Performance of SPT for problem $(f^A, C^B_{\max})$

We analyze the performance of the SPT heuristic for agent B against an arbitrary strategy of A, when its objective is to minimize its makespan.

Theorem 7 *For the minimization of $C^B_{\max}$ the SPT heuristic has the performance bound*

$$\rho(SPT) \leq n$$

and the bound is tight.

Proof. First we show that for any instance the performance for agent B when submitting tasks in SPT order against an arbitrary adaptive strategy of A is bounded by n. Then, we provide an instance to prove the tightness of the bound.

We can clearly assume that $b_n < a_n$ (otherwise any strategy of B is optimal). In the worst case we might have $z^B_{SPT} = \sum_{k=1}^n b_k + \sum_{k=1}^{n-1} a_k$, if all $a_k < b_n$ for $k = 1, \ldots, n-1$. A lower bound on the optimum is trivially given by $z^B \geq \sum_{k=1}^n b_k$. Hence,

$$\rho(SPT) \leq \frac{\sum_{k=1}^n b_k + \sum_{k=1}^{n-1} a_k}{\sum_{k=1}^n b_k} \leq 1 + \frac{(n-1)b_n}{b_n} = n.$$

The lower bound n on the performance of SPT can be shown by the following instance, for some large value M:

$$a_i = \begin{cases} M-1 & i = 1, \ldots, n-1 \\ M+1 & i = n \end{cases} \qquad b_i = \begin{cases} 1 & i = 1 \\ 2 & i = 2, \ldots, n-1 \\ M & i = n \end{cases}$$

Assume that A starts with a_1. The SPT strategy of B submits b_1 and wins. As a reaction the adaptive strategy of A sticks to submitting a_1 and loses also the next rounds allowing B to schedule also its tasks 2 to $n-1$. Then A wins with a_1 against b_n and continues to win thereby scheduling its tasks 1 to $n-1$. Finally, B wins with b_n against a_n. Altogether we get

$$z^B_{SPT} = \sum_{k=1}^{n-1} a_k + \sum_{k=1}^{n} b_k = (n-1)(M-1) + 1 + (n-2)2 + M = nM + n - 2\,.$$

An optimal and omniscient strategy of B wins the first round with b_2. Then A adaptively pursues a different (although quite self-destructive) strategy and continues to submit a_n. In each such round B wins and thus can schedule all its tasks with

$$z^B = \sum_{k=1}^{n} b_k = 1 + (n-2)2 + M\,.$$

Thus we have

$$\rho(SPT) \geq \frac{nM+n-2}{M+2n-3} \overset{M\to\infty}{\longrightarrow} n.$$

□

4.2 Performance of SPT for problem $(f^A, \sum_j C_j^B)$

As in the previous section we analyze the performance of the SPT heuristic for agent B when its objective is the minimization of total completion times.

Theorem 8 *For the minimization of $\sum_j C_j^B$ the SPT heuristic has the performance bound*

$$\rho(SPT) \leq n$$

and the bound is tight.

Proof. We start by proving an upper bound on the performance ratio of SPT. The performance of the algorithm depends on who owns the largest tasks. Denote by b_ℓ the largest task of B which is smaller than the largest task of A, i.e. $b_\ell := \max\{b_i \mid b_i < a_n\}$. This means that no strategy of B can avoid to lose with all tasks $b_{\ell+1}, \dots, b_n$ against any submission of A. All these $n-\ell$ tasks can only be processed after all tasks of A are finished. Clearly, this consideration becomes redundant if $\ell = n$, i.e. $b_n < a_n$.

The optimal solution for B cannot do better than scheduling in SPT order tasks $b_1, \dots, b_\ell$ before any task of A is processed. However, all $n-\ell$ tasks $b_{\ell+1}, \dots, b_n$ are delayed in any feasible solution by the sum of processing times of A-tasks. Thus, we have that the optimal solution value z^B can be bounded as follows:

$$z^B \geq \sum_{k=1}^{n} (n-k+1)\, b_k + (n-\ell) \sum_{k=1}^{n} a_k. \tag{7}$$

In a worst-case scenario for the SPT strategy of B, each submitted task b_j is scheduled only after all shorter tasks of A with $a_i \leq b_j$ were processed. Clearly, no longer task of A can precede b_j on the machine. For $j = 1, \dots, \ell$ there can be at most $n-1$ tasks of A preceding b_j, since

$a_n > b_j$ for $j \leq \ell$. Each such task of A can be upper bounded by b_j. For $j = \ell + 1, \ldots, n$ all tasks of A will precede b_j. Hence, we can bound the objective function of the SPT strategy by

$$z^B_{SPT} \leq \sum_{k=1}^{\ell} (n-k+1)\left((n-1)b_k + b_k\right) + \sum_{k=\ell+1}^{n} (n-k+1) \left(\sum_{j=1}^{n} a_j + b_k \right). \tag{8}$$

In order to show that $z^B_{SPT} \leq n \cdot z^B$, we can combine (7) and (8) so that we are left to prove that

$$\sum_{k=\ell+1}^{n} (n-k+1) \left(\sum_{j=1}^{n} a_j \right) \leq n(n-\ell) \left(\sum_{j=1}^{n} a_j \right)$$

which in turn is equivalent to proving that

$$\sum_{k=\ell+1}^{n} (n-k+1) \leq n(n-\ell).$$

The latter inequality can be verified by elementary calculations for all $\ell \leq n$.

For estimating the lower bound on $\rho(SPT)$, we can use exactly the same instance and submissions applied in the proof of Theorem 7. In this case, since B's objective is the sum of completion times, we get that the value of the solution found by SPT is

$$z^B_{SPT} = 1 + \sum_{k=1}^{n-2} (1+2k) + 1 + (n-2) \cdot 2 + (n-1)(M-1) + M = nM + n^2 - n - 1,$$

while the optimum is

$$z^B = 2 + 3 + \sum_{k=1}^{n-3} (5+2k) + 1 + (n-2)2 + M = M + n^2 + 2n - 7.$$

Similarly as before we get

$$\rho(SPT) \geq \frac{nM + n^2 - n - 1}{M + n^2 + 2n - 7} \overset{M\to\infty}{\longrightarrow} n.$$

□

It is possible to show that the above proof could be considerably strengthened if $b_n \geq a_n$ (i.e. ℓ strictly smaller than n). In particular, using the same arguments presented in the following section, one can reach an upper bound for $\rho(SPT)$ of order $\sqrt{n}$ (see proof of Theorem 9).

4.3 Performance of SPT against a "quasi-rational" opponent for problem $(f^A, \sum_j C^B_j)$

The strategy of agent A pursued in the proof of Theorem 8 clearly corresponds to a very strange behavior. On the other hand, the analysis of an algorithm for B may not assume a totally rational and purposeful strategy of A. So, in order to get rid of completely erratic conducts of A, a highly reasonable but mild assumption on the strategy of A is the following.

Definition 1 *Let $b_{\max}$ denote the largest not yet scheduled task of B. We say that agent A is a* Quasi-Rational Opponent (QRO)*, if A is not allowed to submit a task with length greater than $b_{\max}$, unless for any unscheduled task $a_i \in A$, $a_i > b_{\max}$.*

Quasi-rationality implies that, in each round, A does not submit a task which would lose against *any* submission of B, if it can avoid to do so. Note that if A is a QRO, it is not required to win in any case when it would be possible. In the following, we give an asymptotically tight worst-case performance ratio for this case.

Theorem 9 *For the minimization of $\sum_j C_j^B$ the SPT heuristic has the following performance bound against a quasi-rational agent A:*

$$\frac{\sqrt{2}}{2}\sqrt{n} \;\leq\; \rho(SPT) \;\leq\; \frac{\sqrt{2}}{2}\sqrt{n}\,+1$$

Proof. In order to prove the upper bound on $\rho(SPT)$, without loss of generality we can assume that $a_n \leq b_n$. Otherwise, since A is a QRO, it would be forced to submit its tasks of length greater than b_n only after all tasks of B are scheduled. Hence, these tasks would not affect the solution of B, but would only reduce the number of tasks available for A against B. In the remainder of the proof, quasi-rationality of A is thereby automatically satisfied.

Bounding the optimal solution value requires a more careful argument than the one applied in the proof of Theorem 8. We assume that B manages to schedule all its tasks $b_1, \dots, b_{n-1}$ before any task of A is processed and thus reaches a best possible solution by scheduling its tasks by SPT. Only for the largest task b_n we take into account that no strategy of B can avoid to lose with b_n against any submission of A. Thus, all tasks of A are scheduled before b_n in any case and we have

$$z^B \geq \sum_{k=1}^{n-1}(n-k+1)\,b_k + \sum_{k=1}^{n} a_k + b_n. \tag{9}$$

In a worst-case scenario for the SPT strategy of B, each submitted task b_j is scheduled only after all shorter tasks of A with $a_i \leq b_j$ were processed. Clearly, no longer task of A can precede b_j on the machine. In the worst case, all tasks $a_i \in (b_{j-1}, b_j]$ (if any exist in this interval) are scheduled immediately before b_j and thus maximize their impact on the objective function of B since they contribute to the completion time of all tasks with $b_j \geq a_i$. Hence, we can bound the objective function of the SPT strategy as follows:

$$z^B_{SPT} \leq \sum_{k=1}^{n}(n-k+1)\left(b_k + \sum_{i:a_i\in(b_{k-1},b_k]} a_i\right) \tag{10}$$

Our goal is to prove $z^B_{SPT} \leq r(n)\, z^B$ for some function $r(n)$ as small as possible. Combining (9) and (10), after some algebra, we are left to show that for any given instance

$$\sum_{k=1}^{n}\left(\sum_{i:a_i\in(b_{k-1},b_k]} a_i\right)(n-k+1-r(n)) \leq \sum_{k=1}^{n}(n-k+1)(r(n)-1)b_k. \tag{11}$$

Since (11) must hold for any set of tasks of A, we consider the worst-case of the left-hand side: For all k where the coefficient $(n-k+1-r(n))$ is positive, we increase all tasks $a_i \in (b_{k-1}, b_k]$ and set them equal to b_k. In the worst case, all tasks of A are concentrated into a single interval, where they generate the largest contribution to the sum on the left-hand side of (11). Thus, we define $k^* := \arg\max\{(n-k+1-r(n))b_k \mid k = 1, \dots, n\}$ and assume that A contains exactly n tasks of length b_{k^*}.

Applying the worst-case view to the right-hand side of (11) we note that the coefficients $(n-k+1)(r(n)-1)$ are always positive. The smallest possible choice of processing times for B

would be $b_1 = \ldots = b_{k^*-1} = 0$ and $b_{k^*} = b_{k^*+1} = \ldots = b_n$. Thus, b_{k^*} cancels out, and it suffices to prove the following strengthening of (11):

$$n(n - k^* + 1 - r(n)) \leq (r(n) - 1) \sum_{k=k^*}^{n} (n - k + 1) = (r(n) - 1) \sum_{k=1}^{n-k^*+1} k \tag{12}$$

Plugging in the claimed bound $r(n) := \frac{\sqrt{2}}{2}\sqrt{n} + 1$ and substituting for simplicity $s := n - k^* + 1$ we now have to prove:

$$n(s - \frac{\sqrt{2}}{2}\sqrt{n} - 1) \leq \frac{\sqrt{2}}{4}\sqrt{n}\, s(s + 1) \tag{13}$$

This is equivalent to showing that the following quadratic, convex function $g(s)$ is positive for all s:

$$g(s) := \frac{\sqrt{2}}{2} s^2 + (\frac{\sqrt{2}}{2} - 2\sqrt{n})s + \sqrt{2}\, n + 2\sqrt{n} \tag{14}$$

Thus, we compute the minimum of $g(s)$ attained at

$$\bar{s} = \sqrt{2n} - \frac{1}{2}. \tag{15}$$

Inserting this minimizer in (14) yields after an elementary calculation

$$g(s) \geq g(\bar{s}) = 3\sqrt{n} - \frac{\sqrt{2}}{8}, \tag{16}$$

which is positive for all n. Thus we have shown that for $r(n) = \frac{\sqrt{2}}{2}\sqrt{n} + 1$ inequality (12) holds for any value of k^*. This proves the upper bound stated in the Theorem.

The lower bound on $\rho(SPT)$ can be shown by considering the following instance in which the task lengths depend on some large value M and a parameter f to be determined later:

$$a_i = \begin{cases} M - 1 & i = 1, \ldots, n - 1 \\ M + 1 & i = n \end{cases} \qquad b_i = \begin{cases} 1 & i = 1, \ldots, n - f - 1 \\ M & i = n - f, \ldots, n - 1 \\ M + 2 & i = n \end{cases}$$

Note that B owns the largest task $b_n = b_{\max}$ and this ensures that A is a QRO. Assume that A starts with a_n. By the SPT strategy, B starts by submitting b_1 and will win the first $n - f - 1$ rounds with its smallest tasks, no matter what A does. A reacts by submitting its tasks in SPT order as well and wins the following $n - 1$ rounds with tasks a_1 to a_{n-1}. Then B wins with tasks b_{n-f} up to b_{n-1}. Finally, A schedules a_n and the very last task to be executed is b_n. We get:

$$z^B_{SPT} = \sum_{k=1}^{n-f-1} (n - k + 1) \cdot 1 + (M - 1)(n - 1)(f + 1) + \sum_{k=n-f}^{n-1} (n - k + 1) \cdot M + M + 1 + M + 2$$

In the analysis, we consider M tending to infinity. Thus, we can neglect all terms not containing M. Formally, we have

$$z^B_{SPT} = M \left(nf + n - f - 1 + \frac{1}{2}(f^2 + 3f) + 2 + o(M) \right)$$

An optimal (and omniscient) strategy of B wins the first round with task b_2. Reacting to this signal[3], A now pursues a different, self-destructive strategy and continues to submit a_n. In each such round B wins and thus can schedule b_1 and all its tasks b_3 to b_{n-1}. Clearly, the longest task b_n can be scheduled only at the very end. This yields an optimal sum of completion times for B:

$$z^B = \sum_{k=1}^{n-f-1} (n-k+1) \cdot 1 + \sum_{k=n-f}^{n-1} (n-k+1) \cdot M + (M-1)(n-1) + M + 1 + M + 2$$

As above, we concentrate on the multiples of M.

$$z^B = M \left(\frac{1}{2} f(f+3) + (n-1) + 2 + o(M) \right)$$

Putting terms together and letting M go to infinity we have

$$\rho(SPT) \geq \frac{f^2 + (2n+1)f + (2n+2)}{f^2 + 3f + (2n+2)} .$$

Simplifying the maximization of this lower bound over f we concentrate on the terms of highest order of magnitude and set $f := \sqrt{2n}$, which is very close to the actual maximizer (i.e., $\sqrt{2n+2}$). Plugging in this choice of f into the expression above and using elementary arithmetics we get:

$$\rho(SPT) \geq \frac{2n + 2\sqrt{2}n^{3/2} + \sqrt{2n} + (2n+2)}{2n + 3\sqrt{2n} + (2n+2)} \geq \frac{\sqrt{2}}{2}\sqrt{n}.$$

$\square$

Observe that for $n \geq 3$ we can marginally improve the upper bound of Theorem 9 to $\frac{\sqrt{2}}{2}\sqrt{n} + \frac{1}{2}$. It might be noted that one could also try to find the smallest ratio $r(n)$ for which (12) holds. Indeed, finding the minimizer $\bar{s}$ parametrically depending on $r(n)$ and computing the minimal $r(n)$ for which $g(\bar{s}) \geq 0$ holds, is a tedious exercise which yields a slightly better, but "ugly" value of $r(n)$ as a solution of a quadratic equation.

4.4 Performance of SPT for problem $(f^A, \sum_j w_j C_j^B)$

Hereafter we analyze the performance of the SPT heuristic, treated extensively in the previous sections, for the weighted problem $(f^A, \sum_j w_j C_j^B)$. As it might be expected, its worst-case performance ratio is worse than for the unweighted case.

Theorem 10 *For the minimization of $\sum_j w_j C_j^B$ the SPT heuristic has the performance bound*

$$\rho(SPT) \leq 2n - 1$$

and the bound is tight.

Proof. In order to prove that $\rho(SPT) \leq 2n-1$, we want to bound the completion times of B-tasks. Let C_j (respectively $\tilde{C}_j$) indicate the completion time of a task b_j in the solution resulting from an optimal strategy (respectively from an SPT strategy) of B. To bound $\tilde{C}_j$ we distinguish two cases.

[3]If A cannot distinguish between tasks of equal length, we can set $b_1 = 1 - \varepsilon$.

Case 1: $b_j < a_n$. Clearly, in the worst case, the SPT heuristic schedules before b_j all A-tasks and B-tasks with length at most b_j. Since $b_j < a_n$, there can be at most $n-1$ of such A-tasks. In the worst case, the optimal solution might manage to schedule all these A-tasks and also all other $n-1$ B-tasks later than b_j. This yields:

$$\tilde{C}_j \leq C_j + (2n-2)b_j \leq (2n-1)C_j\,. \tag{17}$$

Case 2: $b_j \geq a_n$. Now it suffices to note that also in an optimal solution *all* tasks of A will always precede b_j, no matter how the submissions are performed. Thus, we have a simple bound for this case:

$$C_j \geq b_j + \sum_{i=1}^{n} a_i. \tag{18}$$

A trivial SPT-argument together with (18) shows

$$\tilde{C}_j \leq \sum_{i=1}^{j} b_i + \sum_{i=1}^{n} a_i \leq nb_j + \sum_{i=1}^{n} a_i \leq nC_j\,. \tag{19}$$

Hence, (17) together with (19), implies that for all j $\tilde{C}_j \leq (2n-1)C_j$, so that $\sum_{j=1}^{n} w_j\tilde{C}_j \leq \sum_{j=1}^{n} w_j(2n-1)C_j$ and the thesis follows.

For proving the tightness of the bound, consider the following instance for some large value M and a small constant $\varepsilon > 0$:

$$a_i = \begin{cases} 1-\varepsilon & i = 1,\ldots,n-1 \\ 2 & i = n \end{cases} \qquad b_i = \begin{cases} 1 & i = 1,\ldots,n-1 \\ 1+\varepsilon & i = n \end{cases} \qquad w_i = \begin{cases} 1 & i = 1,\ldots,n-1 \\ M & i = n \end{cases}$$

Assume that A pursues the following strategy (independently from the submissions of B): In the first round it submits a_n and loses in any case. Then it continues to submit all tasks with length $1-\varepsilon$, before submitting a_n again in the last round.

The SPT strategy of B submits b_1 in the first round and wins. Then it loses the next $n-1$ rounds before winning with tasks $b_2,\ldots,b_n$. Since we can choose M arbitrarily large, we will only consider the part of the objective functions contributed by w_n. Thus we have

$$z^B_{SPT} = (1 + (n-1)(1-\varepsilon) + 1\cdot(n-2) + (1+\varepsilon))M + o(M) = (2n-1)M + o(M)$$

An optimal strategy of B would immediately submit b_n in the first round an win. All the following rounds (where A wins the next $n-1$ rounds, then B schedules its remaining $n-1$ tasks) can be neglected since the completion times of the corresponding tasks are not multiplied by w_n. Hence, $z^B \approx (1+\varepsilon)M$ and the lower bound on $\rho(SPT)$ follows. □

Observe that, even if agent A is a QRO, as in Section 4.2, we cannot get a better performance ratio for problem $(f^A, \sum_j w_jC^B_j)$. In fact, it is possible to modify the above lower bound example (proof of Theorem 10) as follows: Simply add a task $n+1$ with $b_{n+1} = 2+\varepsilon$ and $w_{n+1} = \varepsilon$ so that the quasi rationality of the opponent is implied. This task b_{n+1} is placed at the end of the schedule by the SPT heuristic of B and does not contribute to the terms of order M in the objective function. Thus, the above bound remains valid with $n-1$ replacing n.

Note that in [3] it is shown—by a completely different approach—that in a linear conveyor setting, where A is restricted to submit tasks in a given order, the performance bound of the SPT heuristic is n.

4.5 Performance of WSPT for problem $(f^A, \sum_j w_j C_j^B)$

It is well-known that the WSPT heuristic solves to optimality the standard single agent scheduling problem when the objective is the minimization of total weighted completion time. We now analyze the performance of WSPT for $(f^A, \sum_j w_j C_j^B)$. Recall that WSPT sorts the tasks in increasing order of processing time to weight ratio. Therefore, we will assume for the remainder of this section that tasks of B are numbered so that

$$\frac{p_1}{w_1} \leq \frac{p_2}{w_2} \leq \ldots \leq \frac{p_n}{w_n}.$$

The following theorem shows that, when B wants to minimize the total weighted completion time of its tasks, the WSPT heuristic actually performs better than SPT.

Theorem 11 *For the minimization of $\sum_j w_j C_j^B$ the WSPT heuristic has the performance bound*

$$\rho(WSPT) \leq n$$

and the bound is tight.

Proof. We first estimate the value z^B of an optimal solution. Let $B^- := \{j \mid b_j < a_n\}$ and $B^+ := \{j \mid b_j \geq a_n\}$. The following inequality holds.

$$z^B \geq \sum_{j \in B^-} w_j \left(\sum_{i=1}^{j} b_i \right) + \sum_{j \in B^+} w_j \left(b_j + \sum_{i=1}^{n} a_i \right) \tag{20}$$

The first summation on B^- arises from considering all B-tasks smaller than a_n as scheduled in WSPT ordering without any tasks of A. The second summation, instead, takes into account that tasks in B^+ must be preceded by *all* tasks of A in any case.

As in the proof of Theorem 10, let $\tilde{C}_j$ be the completion time of task b_j in the solution resulting from the WSPT strategy of B. For an upper bound on $\tilde{C}_j$ we distinguish two cases:
Case 1: $j \in B^-$. Define for any task b_j the longest task of B submitted so far, i.e. $\bar{b}_j := \max\{b_i \mid i \leq j\}$. Now in the worst case, by a rough estimate, every task b_j is preceded by *all* tasks of A with $a_i \leq \bar{b}_j$. These might be at most $n-1$ tasks (all except a_n). Thus,

$$\tilde{C}_j \leq \sum_{i=1}^{j} b_i + \sum_{a_i \leq \bar{b}_j} a_i \leq \sum_{i=1}^{j} b_i + (n-1)\bar{b}_j \leq n \sum_{i=1}^{j} b_i\,. \tag{21}$$

Case 2: $j \in B^+$. As above, b_j is preceded by all tasks of A and we simply get

$$\tilde{C}_j \leq \sum_{i=1}^{j} b_i + \sum_{i=1}^{n} a_i\,. \tag{22}$$

Putting the bounds (21) and (22) together with (20), we obtain

$$z^B_{WSPT} = \sum_{j=1}^{n} w_j \tilde{C}_j \leq \sum_{j \in B^-} w_j \left(n \sum_{i=1}^{j} b_i \right) + \sum_{j \in B^+} w_j \left(\sum_{i=1}^{j} b_i + \sum_{i=1}^{n} a_i \right) \leq n z^B.$$

Note that actually only tasks in B^- cause the factor of n since A-tasks cannot interfere with tasks in B^+ which are therefore scheduled optimally by WSPT.

For a lower bound we can use the same instance applied in the proof of Theorem 7 with weights $w_j = 1$ for each task j of B. Then we follow exactly the same arguments applied in the proof of Theorem 8. Since all weights are identical, SPT and WSPT coincide and also the objective function values remain the same and $\rho(WSPT) \geq n$ follows. $\square$

As in Section 4.4 we can easily show that the above lower bound example can be extended with marginal loss to take quasi-rationality of A into account. We simply add a task $n+1$ with $b_{n+1} = M+2$ and $w_{n+1} = 1/M^2$. This task will be scheduled last among all tasks by the WSPT heuristic of B. The increase in the objective function for the WSPT strategy is of order n/M and thus b_{n+1} does not give a relevant contribution to the performance ratio.

Note that the same WSPT heuristic, in the linear conveyor setting studied in [3], has a performance ratio comprised between $\sqrt{n}/2$ and $n-1$.

5 Minimax strategies

In this section we address the problem faced by agent B when it does not know anything about the strategy played by A, the opponent agent. Recall that Ω^A (respectively Ω^B) are the set of all possible strategies for agent A (respectively B) and that $\sigma(s^A, s^B)$ denotes the schedule resulting from a pair of strategies $s^A \in \Omega^A, s^B \in \Omega^B$. Then, a *minimax strategy* $s^B_* \in \Omega^B$ of agent B is defined as follows:

$$s^B_* := \arg \min_{s^B \in \Omega^B} \left(\max_{s^A \in \Omega^A} f^B(\sigma(s^A, s^B)) \right)$$

So, devising a minimax strategy consists in finding a response algorithm for B that minimizes its costs in the face of the worst possible (for B) scenario caused by agent A's strategy.

Solely for the purpose of analyzing the minimax strategies for agent B presented below, we assume that A behaves as an opponent to B interested only in worsening (i.e., maximizing) the solution of B as much as possible, independently from the consequences on its own objective. We call such behavior as *malevolent* strategy.

In [3] the authors consider a linear conveyor setting in which each agent's strategy can be completely defined by an ordered sequence of n tasks, denoted as *submission sequence*. Each task is submitted for possible processing in the given order until it wins. Denote by Λ^A and Λ^B the sets of all possible submission sequences, i.e. *linear strategies*, for agents A and B. Clearly, $\Lambda^X \subset \Omega^X$, for $X = A, B$. It was shown in [3] that – restricted to linear strategies – the SPT order is a malevolent strategy of A for all three problems $(f^A, C^B_{\max})$, $(f^A, \sum C^B_j)$ and $(f^A, \sum w_j C^B_j)$.

In our case, where the agents are free choose any strategy in Ω^A and Ω^B, it is easy to see that SPT is still a malevolent strategy of agent A for $f^B = C^B_{\max}$ and thus SPT is also a minimax strategy of agent B. However, problems $(f^A, \sum C^B_j)$ and $(f^A, \sum w_j C^B_j)$ turn out to be less trivial. In particular, the following two examples show that SPT is not necessarily a malevolent strategy of A. In fact, A may consider to voluntarily lose some rounds even when potentially winning tasks are available.

Example 12 *Consider an instance of problem $(f^A, \sum C^B_j)$ with the following 3 tasks and some large value M:*

$$\begin{array}{lllllll} a_1 & = & 1-\varepsilon & \quad & b_1 & = & 1 \\ a_2 & = & 1+\varepsilon & & b_2 & = & 1 \\ a_3 & = & M+\varepsilon & & b_3 & = & M \end{array}$$

Assume now that B's (nonlinear) strategy consists in submitting b_3 in the first round, then whatever is the outcome of the first round play SPT with its remaining tasks. In this case, A could win the first round by submitting either a_1 or a_2. However, since $M \gg 1$, a malevolent strategy would lose the first round with task a_3 and then play SPT. In fact, if A wins the first round, B's objective is $\approx M$, else, if A loses the first round B's objective becomes $\approx 3M$.

Example 13 *Consider the following instance of problem $(f^A, \sum w_j C_j^B)$:*

$$a_i = \begin{cases} p-\varepsilon & \text{if } 1 \le i \le n-1 \\ p+\varepsilon & \text{if } i = n; \end{cases} \qquad b_i = p \quad \text{for } 1 \le i \le n; \qquad w_i = \begin{cases} 1 & \text{if } 1 \le i \le n-1 \\ M & \text{if } i = n \end{cases}$$

Assume now that B's strategy for the first ℓ rounds is the following (where ℓ is a fixed value between 2 and n): (i) for the first $\ell-1$ rounds, play any task among $\{b_1, \ldots, b_{n-1}\}$; (ii) starting in round ℓ, whatever has happened before, play b_n until it is scheduled; (iii) play the remaining tasks from $\{b_1, \ldots, b_{n-1}\}$ in arbitrary order.

Against this specific strategy of B, a malevolent strategy of A works as follows:
(i) Let B win the first $\ell - 1$ rounds (by playing a_n).
(ii) Win against b_n in the $n-1$ rounds starting with round ℓ.
This strategy of A is malevolent because it postpones as much as possible the execution of the crucial task b_n which attains a completion time $C_n^B = (n+\ell-1)p$ (neglecting ε). However, if A plays SPT against the above strategy of B, we would have $C_n^B = np$.

In the following, we will show that although SPT is not always a malevolent strategy for A, the minimax strategy of B still consists in the best reaction against an SPT submission of A.

In the remainder of this section let $f^B = \sum w_j C_j^B$. The results for the total completion time are an obvious consequence. As already mentioned, when both agents can only use linear strategies, SPT is a malevolent strategy of agent A.

$$SPT = \arg \max_{s^A \in \Lambda^A} \{f^B(\sigma(s^A, s^B))\} \quad \forall\ s^B \in \Lambda^B \tag{23}$$

In the following we first argue that these facts remain valid as long as one of the agents is restricted to use a submission sequence, i.e. a linear strategy, while the other agent can pursue any strategy, linear or not.

It can be shown by a simple exchange argument that if B is forced to use linear strategies, SPT is still a malevolent strategy for A, even if A is allowed to use any strategy in Ω^A.

$$SPT = \arg \max_{s^A \in \Omega^A} \{f^B(\sigma(s^A, s^B))\} \quad \forall\ s^B \in \Lambda^B \tag{24}$$

In fact, A could use its flexibility only to lose voluntarily which does not make sense against a submission sequence of B.

Symmetrically, if agent A adopts a linear strategy, there is an optimal response for agent B which is also a linear strategy (i.e. the flexibility of choosing at each round any available task does not yield a better outcome for B). This is proved by the following lemma.

Lemma 14 *Let A follow a linear strategy $\bar{s}^A$.*

$$\min_{s^B \in \Omega^B} \{f^B(\sigma(\bar{s}^A, s^B))\} = \min_{s^B \in \Lambda^B} \{f^B(\sigma(\bar{s}^A, s^B))\} \tag{25}$$

Proof. Let $\tilde{s}^B \in \Omega^B$ be an optimal response strategy for B against $\bar{s}^A \in \Lambda^A$ an let $\tilde{\sigma} = \sigma(\bar{s}^A, \tilde{s}^B)$ be the corresponding solution schedule. The optimal solution schedule $\tilde{\sigma}$ can be viewed as a sequence of alternating A-*blocks* and B-*blocks*:

$$\tilde{\sigma} = \langle A_1, B_1, A_2, B_2, \dots A_q, B_q \rangle \tag{26}$$

where an A-block (resp., B-block) is a maximal subsequence of consecutive A-tasks (resp., B-tasks). Note that the first and the last blocks A_1 and B_q may be empty. Starting from $\tilde{s}^B$, it is possible to build a new linear strategy $\bar{s}^B$ corresponding to the submission subsequence of B-tasks as they appear in the optimal schedule $\tilde{\sigma}$, i.e., in the order $\langle B_1, B_2, \dots B_q \rangle$.

We now show that the completion time of any B-task in the new schedule $\bar{\sigma} = \sigma(\bar{s}^A, \bar{s}^B)$ obtained when B adopts $\bar{s}^B$, is not greater than that in $\tilde{\sigma}$. Consider three consecutive blocks of $\tilde{\sigma}$: A_i, B_i, and A_{i+1}, for some $i = 1, \dots, q-1$, and let b' and a' be the first tasks of blocks B_i, and A_{i+1}, respectively. Since $\bar{s}^A \in \Lambda^A$, then $a' > b_j$ for all $b_j \in B_i$, i.e. agent A loses with task a' against all tasks of block B_i. On the other hand, agent B adopts a flexible strategy $\tilde{s}^B \in \Omega^B$, and B may voluntarily lose against tasks of block A_i with some tasks larger than b'. So, in general, we cannot say that $b' > a_j$ for all $a_j \in A_i$, i.e. for some $a_j \in A_i$ it is possible that $a_j > b'$. In the derived linear strategy $\bar{s}^B$, B would keep submitting b' against all $a_j \in A_i$ until it wins. Recalling that $a' > b'$, in $\bar{\sigma}$ b' would win against a' (at the latest) or against a task of block A_i which has been submitted by A before a'. Similarly, since $a' > b_j$ for all $b_j \in B_i$, each task of block B_i in $\bar{\sigma}$ would win not later than in the original schedule $\tilde{\sigma}$. Applying the above arguments for $i = 1$ to $q-1$, we have that the completion times of all B-tasks in $\bar{\sigma}$ have not increased with respect to the corresponding values in the original schedule $\tilde{\sigma}$. Therefore $f^B(\bar{\sigma}) \le f^B(\tilde{\sigma})$ and the linear strategy $\bar{s}^B$ is also an optimal response strategy.

In conclusion, starting from any flexible strategy $\tilde{s}^B \in \Omega^A$ of agent B, there is always a linear strategy $\bar{s}^B \in \Lambda^A$ which does not worsen B's performance and this proves the thesis. □

Putting together the facts established above, we can show that the situation remains unchanged even if *both* agents are not restricted to linear strategies.

Theorem 15

$$z^\Omega = \min_{s^B \in \Omega^B} \left(\max_{s^A \in \Omega^A} f^B(\sigma(s^A, s^B)) \right) = \min_{s^B \in \Lambda^B} \left(\max_{s^A \in \Lambda^A} f^B(\sigma(s^A, s^B)) \right) = z^\Lambda \tag{27}$$

Furthermore, the minimax strategy is given by the best response submission sequence of B against the SPT submission of A.

Proof. $z^\Omega \le z^\Lambda$:

$$\begin{aligned} z^\Omega &\le \min_{s^B \in \Lambda^B} \left(\max_{s^A \in \Omega^A} f^B(\sigma(s^A, s^B)) \right) \qquad (\text{since } \Lambda^B \subseteq \Omega^B) \\ &= \min_{s^B \in \Lambda^B} f^B(\sigma(SPT, s^B)) \qquad (\text{from (24)}) \\ &\le \min_{s^B \in \Lambda^B} \left(\max_{s^A \in \Lambda^A} f^B(\sigma(s^A, s^B)) \right) = z^\Lambda \end{aligned}$$

$z^\Omega \ge z^\Lambda$:

$$\begin{aligned} z^\Omega &\ge \min_{s^B \in \Omega^B} f^B(\sigma(SPT, s^B)) \\ &= \min_{s^B \in \Lambda^B} f^B(\sigma(SPT, s^B)) \qquad (\text{Lemma 14 with } \bar{s}^A = SPT) \\ &= \min_{s^B \in \Lambda^B} \left(\max_{s^A \in \Lambda^A} f^B(\sigma(s^A, s^B)) \right) = z^\Lambda \qquad (\text{from (23)}) \end{aligned}$$

The second line of this part of the proof also shows that the minimax objective of B is attained by a best response submission sequence against SPT. □

The above results show that a minimax strategy for agent B consists in a linear strategy representing the best response submission sequence when A plays its tasks in SPT order.

Corollary 16 *A minimax strategy of agent B for problem $(f^A, \sum w_j C_j^B)$ can be computed in time $O(n^2)$ and for problem $(f^A, \sum C_j^B)$ in $O(n \log n)$ time.*

Proof. In [3, Sec. 4.2] a nontrivial $O(n^2)$ time algorithm WSPTInsert, which computes the best response of B against any given submission sequence of A, is presented. Thus, we can compute the minimax strategy of agent B for problem $(f^A, \sum w_j C_j^B)$ by applying algorithm WSPTInsert against a SPT submission of A.

Note that the above algorithm returns a SPT submission sequence of B-tasks, when all w_j are identical. Hence, SPT is a minimax strategy of B for problem $(f^A, \sum C_j^B)$. Clearly, it can be computed in $O(n \log n)$ time. □

6 Conclusions

In this paper we study a two-agent scheduling problem where a single machine iteratively selects the next task to be processed from two tasks submitted by each of the two agents. We address the scenario in which, at each round, the shortest submitted task is selected for processing, while the agents pursue the most common scheduling performance indices (makespan, total weighted or unweighted completion time). A first detailed analysis of this type of scheduling structure is given in [3] where the case of a shop with two linear conveyors—one for each agent—is presented. Here we study a *flexible* environment, introduced in [2], in which each agent is allowed to withdraw a losing task and submit any unprocessed task in the next round. While the results addressing Pareto optima are quite similar in the two situations, the problem faced by a single agent turns out to be a different kettle of fish, since any strategy adopted by one agent may suffer from a completely erratic behavior of the opponent.

The scheduling scenarios introduced in this and the above cited papers leave room for several variants and put a number of promising directions forward. Some of them are listed hereafter.

- The situation described in this paper can be viewed as a classical game, in which for each pair of strategies adopted by the two agents, one can determine the corresponding agents' payoffs. Thus, we can apply the concepts of extensive games with a decision tree. In our case, the special variant with *simultaneous moves* applies (see e.g. [13, Sec. 7.1]) since both players (agents) submit their tasks in parallel.

  In this framework it is interesting to study the optimal strategy of one agent, taking into account the fact that also the other agent follows a selfish optimal strategy. Such an optimal strategy for both agents can be determined by backward induction, but it requires exponential time. This situation was studied for other combinatorial optimization problems, e.g. the subset sum problem [6]. An interesting question would be to identify polynomially solvable scenarios consisting either of restrictions to the scheduling environment or fixing the strategy of one agent.

- When addressing the problem from a single agent perspective, similarly to the minimax setting, one agent might have little information on the opponent data. So, an interesting information setting to be studied is the following. The number and length of the opponent's tasks are known in advance but they are only disclosed over time: in this case, an agent may only define a strategy in a dynamic way.

- In [2], a different shop configuration, called *circular conveyor* model, is briefly discussed. Here the losing task of each round is moved to the end of the submission sequence of the corresponding agent. This scenario delivers several mathematically intriguing questions which still remain open.

Acknowledgements

We would like to thank Alessandro Agnetis (Università degli Studi di Siena) for contributions and comments in the early phase of this research.

Ulrich Pferschy was supported by the Austrian Science Fund (FWF): [P 23829-N13].

References

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

[2] Agnetis A., G. Nicosia, A. Pacifici, U. Pferschy (2013). Two agents competing for a shared machine, Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT 2013), *Lecture Notes in Computer Science*, 8176, 1–14, Springer.

[3] Agnetis A., G. Nicosia, A. Pacifici, U. Pferschy (2013). Scheduling of two agents task chains with a central selection mechanism, submitted (a preliminary version is available as Optimization Online 2013-08-3980).

[4] Angel E., E. Bampis, F. Pascual (2006). Truthful algorithms for scheduling selfish tasks on parallel machines, *Theoretical Computer Science*, 369(13), 157–168.

[5] Baker, K., J.C. Smith (2003). A multiple criterion model for machine scheduling, *Journal of Scheduling*, 6(1), 7–16.

[6] Darmann A., G. Nicosia, U. Pferschy, J. Schauer (2014). The subset sum game, *European Journal of Operational Research*, 233(3), 539–549.

[7] Huynh Tuong, N., A. Soukhal, J.-C. Billaut (2012). Single-machine multi-agent scheduling problems with a global objective function, *Journal of Scheduling*, 15(1), 311–321.

[8] Immorlica N., L. Li, V. Mirrokni, A. Schulz (2009). Coordination mechanisms for selfish scheduling. *Theoretical Computer Science*, 410(17), 1589–1598.

[9] Leung Y.T., M. Pinedo, G. Wan (2010). Competitive two-agent scheduling and its applications. *Operations Research*, 58(2), 458–469.

[10] Marini C., G. Nicosia, A. Pacifici, U. Pferschy (2013). Strategies in competing subset selection, *Annals of Operations Research*, 207 (1), 181–200.

[11] Nicosia G., A. Pacifici, U. Pferschy (2011). Competitive subset selection with two agents, *Discrete Applied Mathematics*, 159(16), 1865–1877.

[12] Nisan N., A. Ronen (1999). Algorithmic mechanism design, Proceedings of the 31st Annual ACM Symposium on Theory of Computing, STOC 1999, 129–140.

[13] Osborne M.J. (2004). *An Introduction to Game Theory*, Oxford University Press.

[14] T'Kindt V., J-C. Billaut (2006). *Multicriteria scheduling. Theory, models and algorithms*, Springer.

[15] Wellman, M.P., W.E. Walsh, P.R. Wurman, J.K. MacKie-Mason (2001). Auction protocols for decentralized scheduling, *Games and Economic Behavior*, 35 (1-2), 271–303.

[16] Zhao, K., Lu, X. (2013). Approximation schemes for two-agent scheduling on parallel machines, *Theoretical Computer Science*, 468 , 114–121.

A Proof of Proposition 3

Consider an instance of $(C^A_{\max}, \sum w_j C^B_j)$ with the following processing times for $\varepsilon \ll 1$ and $M \gg 2^n$:

$$a_i = \begin{cases} \varepsilon & i = 1, \dots, n-1 \\ M & i = n \end{cases} \qquad b_i = w_i = \begin{cases} 2^i & i = 1, \dots, n-1 \\ M + \varepsilon & i = n \end{cases}$$

Let $S \subseteq \{1, \dots, n-1\}$, $\bar{S} = \{1, \dots, n-1\} \setminus S$. We build a feasible solution $\sigma(S)$ assuming that agent A submits its tasks in SPT order and agent B submits tasks b_i for all $i \in S$ (in any order) first, then b_n, so that a_n can win that round. At this point, B is left with b_n and all tasks b_i with $i \in \bar{S}$ which are scheduled again in arbitrary order.

Note that, since for each B-task processing time equals weight, once the set S is chosen, the relative order of B-tasks in the blocks preceding and succeeding a_n is irrelevant for B by the usual WSPT argument. So, consider the following resulting schedule:

$$\langle \underbrace{a_1, \dots, a_{n-1}}_{(n-1)\varepsilon} \mid \underbrace{B\text{-tasks in } S}_{\sum_{i \in S} 2^i} \mid \underbrace{\quad a_n \quad}_{M} \mid \underbrace{\quad b_n \quad}_{M+\varepsilon} \mid \underbrace{B\text{-tasks in } \bar{S}}_{\sum_{i \in \bar{S}} 2^i} \rangle.$$

We claim that, for all $S \subseteq \{1, \dots, n-1\}$, $\sigma(S)$ is Pareto optimal. In fact, let $w(S) = \sum_{i \in S} 2^i$ be the total weight (equal to the total duration) of the B-tasks corresponding to set S and, similarly, $w(\bar{S})$ that of tasks corresponding to $\bar{S}$, while W denotes the total weight of *all* B-tasks.

For simplicity, but w.l.o.g. we will set $\varepsilon = 0$ in the following and assume that B submits its tasks in S and $\bar{S}$ in SPT order. Then the objective function values for the two agents are:

$$f^A(\sigma(S)) = w(S) + M \tag{28}$$

$$f^B(\sigma(S)) = \sum_{j \in S} \left(w_j \sum_{\substack{i \in S \\ i \leq j}} b_i \right) + M(w(S) + 2M) + \sum_{j \in \bar{S}} \left(w_j \left(w(S) + 2M + \sum_{\substack{i \in \bar{S} \\ i \leq j}} b_i \right) \right)$$

$$= 2M^2 + M \underbrace{(w(S) + 2w(\bar{S}))}_{=W + w(\bar{S})} + \sum_{j \in S} \left(w_j \sum_{\substack{i \in S \\ i \leq j}} b_i \right) + \sum_{j \in \bar{S}} \left(w_j \left(w(S) + \sum_{\substack{i \in \bar{S} \\ i \leq j}} b_i \right) \right) \tag{29}$$

If we consider a different subset $S' \neq S$ with $w(S) < w(S')$, then $f^A(\sigma(S)) < f^A(\sigma(S'))$ and clearly $w(\bar{S}) > w(\bar{S}')$. Comparing $f^B(\sigma(S))$ to $f^B(\sigma(S'))$ it suffices to consider terms depending on M, since M can be chosen arbitrarily large and thus obliterates the influence of the sum terms in (29). Hence we get that $f^B(\sigma(S)) - f^B(\sigma(S'))$ is of order $M(w(\bar{S}) - w(\bar{S}')) > 0$ and thus M can be chosen such that $f^B(\sigma(S)) > f^B(\sigma(S'))$. Since there are exponentially many sets S all with different weights Proposition 3 holds.

B Proof of Proposition 4

In this proof, we use the following well-known (binary) $\mathcal{NP}$-complete problem:

> Knapsack:
> **Instance**: a set of *items* $I = \{1, \ldots, n\}$, having nonnegative integer *utilities* $\{u_1, \ldots, u_n\}$ and *volumes* $\{v_1, \ldots, v_n\}$, two integer target values U^T and V^T.
> **Question**: Is there a subset $S \subseteq I$ of items such that
>
> $$\sum_{i \in S} v_i \leq V^T \tag{30}$$
>
> $$\text{and} \qquad \sum_{i \in S} u_i \geq U^T \ ? \tag{31}$$

We reduce Knapsack to our recognition problem. Denote by $U = \sum_{i=1}^n u_i$ the total utility and by $V = \sum_{i=1}^n v_i$ the total volume of all items in Knapsack.

Given an instance of Knapsack, define the following instance of our problem with $n+1$ tasks for each agent, where $M \gg UV$ is a large enough integer and $\varepsilon \ll 1$.

$$a_i = \begin{cases} \varepsilon & i = 1, \ldots, n \\ M & i = n+1 \end{cases} \qquad b_i = \begin{cases} v_i & i = 1, \ldots, n \\ M + \varepsilon & i = n+1 \end{cases}$$

$$w_i = \begin{cases} u_i & i = 1, \ldots, n \\ \varepsilon & i = n+1 \end{cases} \qquad \begin{array}{rcl} Q^A & = & V^T + M \\ Q^B & = & M(U - U^T) + UV \end{array}$$

For simplicity, we will set $\varepsilon = 0$ and assume w.l.o.g. that A submits its tasks in SPT order. We first observe that any feasible solution has the following structure:

$$\langle \underbrace{a_1, \ldots, a_n}_{n\varepsilon} \mid \text{first block of } B\text{-tasks} \mid \underbrace{\quad a_{n+1} \quad}_{M} \mid \text{second block of } B\text{-tasks including } b_{n+1} \ \rangle$$

Hence, the structure of a feasible solution is defined by the round in which agent B decides to submit b_{n+1} to let a_{n+1} win. We call *first block* and *second block* the two sets of B-tasks scheduled before and, respectively, after the long task a_{n+1}.

We want to show that our problem has a solution if and only if Knapsack has a solution. Given any yes-solution $\hat{\sigma}$ to our recognition problem, we can associate to it a schedule σ obtained by moving b_{n+1} to the last position. Obviously, since $f^A(\hat{\sigma}) = f^A(\sigma)$ and $f^B(\sigma) \leq f^B(\hat{\sigma})$ for ε sufficiently small, also σ is a yes-solution. In the corresponding Knapsack instance, we take as set S the set of items associated to the tasks scheduled in the first block in σ. Hence, the total length of the B-tasks in the first block in σ equals the total volume of these items, and since $C^A_{\max}(\sigma) \leq Q^A = V^T + M$, one has that (30) is satisfied. If we let U_S denote the total utility of the items corresponding to the tasks in the first block in σ, the total weight of the tasks in the second block is $U - U_S$. Hence, the contributions to f^B of the first, resp. second block, do not exceed VU_S, resp. $(M+V)(U-U_S)$, which yields

$$f^B(\sigma) \leq M(U - U_S) + UV. \tag{32}$$

In order to prove (31), namely that $U_S \geq U^T$, we assume in contrary that $U_S \leq U^T - 1$: Then the contribution of the second block would be at least $M(U - (U^T - 1)) = M(U - U^T) + M$, and this would exceed Q^B (recalling $M > UV$), in contradiction to the feasibility of σ. Hence, (31) is verified.

Viceversa, given the set S in a yes-instance of KNAPSACK, we build a schedule σ with the structure indicated in the picture above with b_{n+1} submitted to let a_{n+1} win and thus separating the first block of B-tasks containing the tasks in S from the tasks in $\bar{S}$, scheduled in the second block. Clearly, (30) implies that $C^A_{\max} \leq Q^A$. Similarly, from (31), the total weight U_S of the tasks in the first block (i.e., the total utility of the items in S) is at least U^T and hence, from (32), one has $f^B(\sigma) \leq Q^B$.