Selfish Scheduling with Setup Times

Laurent Gourvès[1], Jérôme Monnot[1], and Orestis A. Telelis[2]

1. LAMSADE, CNRS FRE 3234, Université de Paris-Dauphine, 75775 Paris, France
{laurent.gourves, monnot}@lamsade.dauphine.fr
2. PNA1, Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
telelis@cwi.nl

Abstract. We study multiprocessor scheduling games with setup times on identical machines. Given a set of scheduling policies (coordination mechanism) on the machines, each out of n players chooses a machine to assign his owned job to, so as to minimize his individual completion time. Each job has a processing length and is of a certain type. Same-type jobs incur a setup overhead to the machine they are assigned to. We study the Price of Anarchy with respect to the makespan of stable assignments, that are pure Nash or strong equilibria for the underlying strategic game. We study in detail the performance of a well established preemptive scheduling mechanism. In an effort to improve over its performance, we introduce a class of mechanisms with certain properties, for which we examine existence of pure Nash and strong equilibria. We identify their performance limitations, and analyze an optimum mechanism out of this class. Finally, we point out several interesting open problems.

1 Introduction

We study multiprocessor job scheduling games with setup times, where each out of n players assigns his owned job for execution to one out of m machines. Jobs are of certain *types*. On any machine, jobs of a given type may be executed *only* after a type-dependent preprocessing (performed once for all same-type jobs) called *setup*. Each machine schedules its assigned jobs according to a *scheduling policy* (algorithm). Given the deployed scheduling policies, players assign their jobs selfishly, to minimize their individual completion times. We examine the impact of selfish behavior on the *overall* (social) cost of *stable* assignments and how this can be alleviated, by deployment of appropriate scheduling policies on the machines. Stable assignments are pure Nash equilibria (PNE) of an underlying strategic game, or strong equilibria (SE); the latter extend PNE by being resilient to coalitional deviations [1]. The overall cost of stable assignments is measured by the latest completion time among all players, known as *makespan*. System performance degradation due to selfish behavior is measured by the *Price of Anarchy*, the worst-case ratio of the makespan of the most expensive equilibrium, relative to the optimum achievable makespan [2].

Our work is motivated by concerns in performance optimization of large-scale distributed systems (computational grids, P2P file sharing systems etc.).

In these systems autonomous entities (end-users, communities, enterprises) compete strategically for resources (distributed storage, processing power, bandwidth) to increase their individual profit. Setup overheads in these systems may well dominate the net processing load of tasks assigned by users; consider e.g. loading application environments, booting operating systems, establishing QoS for network connections. Scheduling with setup times also provides for modeling another very natural situation; the case where many autonomous users may benefit from the output of an identical process. In this case users may only care for the output of the setup (which takes non-negligible time) but their jobs may have virtually zero load for the machine. As an example, consider the setup corresponding to computation and writing of output to a remote file which users simply read. Then the machine may also have to deal *obliviously* of users' presence; only one of them may actually declare his presence by requesting execution of the setup, whereas the rest simply benefit from the request.

The vast number of resources and users in modern distributed environments renders centralized deployment of global resource management policies expensive and inefficient; a central authority may only specify local operational rules per resource, to coordinate users towards globally efficient system utilization. The deployment of such local rules was formalized in [3], under the notion of *coordination mechanisms* and demonstrated for scheduling and network congestion games. In scheduling, local policies are scheduling algorithms deployed on the machines; their set is referred to as a coordination mechanism. The policies may be preemptive or non-preemptive, deterministic or randomized; a policy decides the order of execution of assigned jobs on each machine and may also introduce delays in a systematic manner. A coordination mechanism induces a strategic game, by affecting the players' completion times. The purpose of designing coordination mechanisms is to induce a strategic game that has stable assignments (in our case, PNE or SE outcomes) with low PoA, that can be found efficiently.

A series of works concerned the study of strategic games induced by well established and novel scheduling policies applied on basic scheduling settings [2, 4, 3, 5–10]. The PoA of strong equilibria (SPoA) was first studied for selfish scheduling games in [11] under the preemptive mechanism introduced in [2]. We give a brief account of these works below, in section 2. Our focus is on *strongly local scheduling policies* under which, the completion time of a job j on a machine i is solely dependent on the parameters (with respect to i) of jobs assigned to i. It is called simply *local* if the completion time of j depends on parameters of jobs assigned to i across all machines. We investigate in detail the performance of strongly local mechanisms that can handle the challenges outlined above.

Contribution We analyze the performance of deterministic strongly local coordination mechanisms for selfish scheduling with setup times, on identical machines. First we give a detailed account of the performance of a well established preemptive scheduling mechanism, referred to as `Makespan` (section 4), which was introduced in [2] and studied subsequently for several scheduling game models [7, 11, 12]. We show that the game induced by `Makespan` always has SE, by generalizing a proof of [11]. If k denotes the number of different types of jobs, the PoA

of `Makespan` is shown to be m when $m \leq k$ and $k+1-\epsilon$ (for some $1 > \epsilon \geq 1/m$) when $m > k$. We show that $SPoA = 3/2$ for two machines and 2 for $m \geq 3$. In section 5 we study a class of deterministic strongly local mechanisms, referred to as *type ordering* mechanisms, that can schedule jobs obliviously of the number of players with zero processing lengths. We prove that any deterministic type ordering mechanism induces a strategic game that has PNE, for any number of machines, and SE for 2 machines. We prove a lower bound of $\frac{m+1}{2}$ for the PoA of type ordering mechanisms, and argue that other intuitive solutions are no more powerful than type ordering mechanisms in our setting. In section 6 we analyze the performance of an optimal type ordering mechanism. It achieves a PoA of $\frac{m+1}{2}$, when $m \leq k$, and $\frac{k+3}{2} - \epsilon$ ($\epsilon = \frac{k}{m}$, when m is even and $\epsilon = \frac{k-1}{m-1}$ otherwise) when $m > k$. We conclude with challenging open problems in section 7.

2 Related Work

Performance of coordination mechanisms with respect to Nash equilibria in multi-processor scheduling *games* has been the subject of several recent works [2, 4, 3, 7, 6, 5, 11, 12, 8, 9]. The preemptive mechanism known as `Makespan` was introduced and studied in [2] in the scheduling setting of *uniformly related* machines; each machine i has speed v_i and each job j has processing length ℓ_j, so that the time needed by i to execute j is ℓ_j/v_i. The `Makespan` mechanism schedules jobs in parallel on each machine, so that they all have the same completion time. `Makespan` was shown to have $PoA = \Theta(\frac{\log m}{\log\log m})$ on uniformly related machines in [4, 5] (see also [13]). In case of *identical* machines, all speeds are equal and the PoA of PNE is known to be $\frac{2m}{m+1}$ by the works of [14, 15]. This holds also for the $(S)PoA$ of strong equilibria, which were shown to exist in any machine model [11]. [12] studied the $SPoA$ as a function of the number of different speeds.

Scheduling games in *unrelated* machines were additionally studied in [7–9]. In the *unrelated* machines model a job's processing time depends solely on the machine it is assigned to. In [7], bounds on the PoA of well known deterministic and randomized scheduling policies were studied for unrelated machines. A wide class of non-preemptive strongly local scheduling policies was shown in [8] to have $PoA \geq \frac{m}{2}$. This class contains well known policies such as "longest" and "shortest job first". The authors designed a simply *local* mechanism that induces PNE with $PoA = O(\log^2 m)$. A local mechanism with PNE having $PoA = O(\log m)$ was given recently in [9]. Deterministic and randomized *non-clairvoyant* mechanisms for scheduling on unrelated machines were studied in [10].

Preemptive multi-processor scheduling *with setup times* [16] (see also [17] [SS6]) requires a minimum makespan preemptive schedule, such that the setup is executed by a machine between execution of two job portions of different type. The best known approximation algorithm has a performance guarantee of $\frac{4}{3}$ [18] (see [19] for a previous $\frac{3}{2}$ factor). For equal setup times a PTAS is given in [18] and an FPTAS for 2 machines in [20]. See [21] for a slightly different version.

3 Definitions

We consider m identical machines, indexed by $i \in \mathcal{M} = \{1, \ldots, m\}$ and n jobs $j \in \mathcal{J} = \{1, \ldots n\}$, each owned by a self-interested player. We use interchangeably the terms *job* and *player*. Every job j has a *type* $t_j \in \mathcal{U}$, where $\mathcal{U}$ is the universe of all possible types [1]. The subset of $\mathcal{U}$ corresponding to the set of jobs $\mathcal{J}$ is denoted by $\mathcal{T} = \{t_j | j \in \mathcal{J}\}$ and define $k = |\mathcal{T}|$. We refer to any specific type by θ. Each job $j \in \mathcal{J}$ and each type $\theta \in \mathcal{U}$ are respectively associated to processing length $\ell_j \geq 0$ and setup time $w(\theta) \geq 0$. If $w(\theta) = 0$, then $\ell_j > 0$ for all j with $t_j = \theta$. Otherwise, we allow $\ell_j = 0$. Agent $j \in \mathcal{J}$ chooses a strategy $s_j \in \mathcal{M}$ (a machine to assign his job to). A strategy profile (assignment) is denoted by $s = (s_1, \ldots, s_n)$; s_{-j} refers to s, without the strategy of j.

The cost of a player j under assignment s is $c_j(s)$, the completion time of his job. $c_j(s)$ depends on the scheduling policy deployed on machine s_j. The completion time of machine $i \in \mathcal{M}$ under s is $C_i(s) = \max_{j:s_j=i} c_j(s)$. The social cost function is the makespan $C(s) = \max_i C_i(s) = \max_j c_j(s)$. We use $L_i(s) = \sum_{j:s_j=i} \ell_j$ for the total job processing load on machine i under assignment s, excluding setup times. $L_{i,\theta}(s) = \sum_{t_j=\theta, s_j=i} \ell_j$ is the total processing length of type θ assigned on machine i. $T_i(s)$ denotes the subset of types that have jobs on machine i under s. A scheduling policy is a scheduling algorithm. The set of scheduling policies deployed on all machines is a coordination mechanism.

Definition 1. **[1, 11]** *A strategy profile s is a strong equilibrium if for every subset of players $J \subseteq \mathcal{J}$ and every assignment $s' = (s_{-J}, s'_J)$, where $s'_j \neq s_j$ for all $j \in J$, there is at least one player $j_0 \in J$ with $c_{j_0}(s) \leq c_{j_0}(s')$.*

The makespan of a socially optimum assignment s^* can be lower bounded as:

$$\begin{array}{llll}
\textbf{(a)} & mC(s^*) & \geq \sum_{\theta \in \mathcal{T}} w(\theta) + \sum_{j \in \mathcal{J}} \ell_j & \\
\textbf{(b)} & C(s^*) & \geq \quad w(t_j) + \ell_j & \text{for any } j \in \mathcal{J} \\
\textbf{(c)} & (k-1)C(s^*) & \geq \sum_{\xi \in \mathcal{T} \setminus \{\theta\}} w(\xi) & \text{for any } \theta \in \mathcal{T}
\end{array} \qquad (1)$$

The only restriction that we impose on the scheduling policies is that the setup of any type θ on any machine i is executed before execution of type θ jobs on i.

4 On the `Makespan` Mechanism

We study an adaptation of the preemptive mechanism introduced in [2] and referred to as `Makespan` [7, 10]. In any assignment s under `Makespan` it is $c_j(s) = C_{s_j}(s)$ for every $j \in \mathcal{J}$; completion time of j equals completion time of the machine that j is assigned to. `Makespan` schedules jobs on a machine in parallel by usage of time multiplexing. They are broken into small pieces that are executed in a round-robin fashion; each job is assigned a fraction of the machine's time proportionally to its processing length.

[1] E.g. the set of application environments installed on each machine.

Theorem 1. *Strong Equilibria exist in the scheduling game with setup times, under the* `Makespan` *mechanism.*

The proof of this result can be found in **Appendix A**. It is a generalization of the proof of [11]. Their proof depends crucially on all jobs having non-zero processing length (see Lemma $A.1$ in the full version [11]). To overcome this, we used a more detailed vector potential function.

Theorem 2. *The PoA of the* `Makespan` *mechanism for the scheduling game with setup times is m when $m \leq k$, at most $k + 1 - k/m$ when $m > k$ and at least $\frac{k+1}{1+\epsilon}$ for $m \geq 3k - 2$ and $\epsilon = \frac{2k-1}{m-k+1}$.*

Proof. **Case** $m \leq k$: The most expensive PNE s has makespan $C(s) \leq \sum_\theta w(\theta) + \sum_j \ell_j$ (all jobs on one machine). By (1a), it is $PoA \leq m$. For the lower bound take $k = m$ types of jobs, each type θ having $w(\theta) = 1$ and containing m jobs of zero processing length. A job of each type is assigned to each machine in s . Then $C(s) = m$ and s is clearly a PNE. In the social optimum s^* each type is assigned to a dedicated machine, thus $C(s^*) = 1$.

Case $m > k$: Assume that for the most expensive PNE s it is $C(s) = C_1(s)$. Let x be a job of type $t_x = \theta$ executed on machine 1. x cannot decrease its cost $c_x(s) = C_1(s)$ by switching to any machine $i \neq 1$. Then $C_1(s) \leq C_i(s) + \ell_x$ if $\theta \in T_i(s)$ or $C_1(s) \leq C_i(s) + w(\theta) + \ell_x$ otherwise. We sum up the inequalities over all machines, assuming that θ does not appear on α machines, and add $C_1(s)$ to both sides to obtain $mC_1(s) \leq \sum_{i=1}^m C_i(s) + \alpha w(\theta) + (m-1)\ell_x \leq m \sum_{\xi \in \mathcal{T}} w(\xi) + (m-1)\ell_x + \sum_{j \in \mathcal{J}} \ell_j$. Divide by m and rewrite it as:

$$C_1(s) \leq \frac{m-1}{m}\Big(\sum_{\xi \in \mathcal{T} \setminus \{\theta\}} w(\xi) + w(\theta) + \ell_x \Big) + \frac{1}{m}\Big(\sum_{\xi \in \mathcal{T}} w(\xi) + \sum_{j \in \mathcal{J}} \ell_j \Big)$$

Using (1a,b,c), $C(s) \leq \frac{m-1}{m}\left((k-1)C(s^*) + C(s^*)\right) + C(s^*) = (k+1-\frac{k}{m})C(s^*)$.

For the lower bound take k types, $m \geq 3k-2$, and let $w(1) = 0$ and $w(\theta) = 1$ for $\theta \in \{2, \ldots, k\}$. There are $k+1$ jobs of type 1 and length 1 and $\frac{m-1}{\epsilon}$ jobs of type 1 and length $\epsilon = \frac{2k-1}{m-k+1}$. Types $\theta \in \{2, \ldots, k\}$ have $m-1$ jobs each, of processing length 0. A PNE s is as follows. $k+1$ jobs of type 1 and length 1 are assigned to machine 1. One job from each type $\theta \geq 2$ is assigned to each machine $i = 2, \ldots, m$. $\frac{1}{\epsilon}$ jobs of type 1 and length ϵ are also assigned to each machine $i \geq 2$. Thus $C_i(S) = k$ for $i \geq 2$ and $C_1(s) = k+1$. No job may decrease its completion time (equal to the makespan of the machine it is assigned to) by switching machine. In the optimum assignment s^* assign two jobs of type 1 - with lengths 1 and ϵ - to each machine $i = 1 \ldots k+1$. Every machine $i = k+2 \ldots 2k$, has $m-1$ jobs of type $i-k$, each of length 0. Every machine $i = 2k+1 \ldots m$, has $1/\epsilon + 1$ jobs of type 1, of length ϵ. The makespan of s^* is $1 + \epsilon$. $\square$

Theorem 3. *The Price of Anarchy of strong equilibria under* `Makespan` *for the scheduling game with setup times is 2 for $m \geq 3$, and $\frac{3}{2}$ for $m = 2$ machines.*

Proof. We give the proof for the case $m \geq 3$. The reader is referred to **Appendix B** for the case $m = 2$. Let s be a SE, s^* the socially optimum assignment, and $C(s) = C_1(s)$. If $C_1(s) \leq C(s^*)$ we get $SPoA = 1$. If $C_1(s) > C(s^*)$, there is machine $i \neq 1$ with $C_i(s) \leq C(s^*)$, because otherwise s would not be a SE; all jobs would reduce their completion time by switching from s to s^*. For any job x with $s_x = 1$, it is $c_x(s) \leq c_x(s_{-x}, i)$. Thus $C_1(s) = c_x(s) \leq C_i(s) + w(t_x) + \ell_x$. Thus $C(s) = c_x(s) \leq 2C(s^*)$, because $C_i(s) \leq C(s^*)$ and (1b). For the lower bound, take 3 machines and 4 jobs, with $t_1 = t_2 = \theta_1$ and $t_3 = t_4 = \theta_2$. Set $w(\theta_1) = \epsilon$, $\ell_1 = \ell_2 = 1$ and $w(\theta_2) = 1$, $\ell_3 = \ell_4 = \epsilon$. An assignment where jobs $1, 2$ play machine 1 and jobs $3, 4$ play machines $2, 3$ respectively is a strong equilibrium of makespan $2 + \epsilon$. In the social optimum jobs $3, 4$ are assigned to the same machine and 1 and 2 on dedicated machines; the makespan becomes then $1 + 2\epsilon$. Thus $SPoA \geq \frac{2+\epsilon}{1+2\epsilon} \to 2$, as $\epsilon \to 0$. $\square$

5 Type Ordering Mechanisms

We describe a class of (deterministic) *type ordering* mechanisms, for *batch scheduling* of same-type jobs. Each machine i groups together jobs of the same type θ, into a *batch* of type θ. A type batch is executed by the machine as a whole; the setup is executed first, followed by *preemptive* execution of all jobs in the batch, in a `Makespan` fashion. Jobs within the same batch have equal completion times and are scheduled preemptively in parallel. Type batches are executed serially by each machine.

Policies in type ordering mechanisms satisfy a version of the property of *Independence of Irrelevant Alternatives* (IIA) [8]. Under the IIA property, for any set of jobs $J_i \subseteq \mathcal{J}$ assigned to machine $i \in \mathcal{M}$ and for any pair of types $\theta, \theta' \in \mathcal{U}$ with jobs in J_i if the θ-type batch has smaller completion time than the θ'-type batch, then the θ batch has a smaller completion time than the θ' batch in any set $J_i \cup \{j\}$, $j \in \mathcal{J} \setminus J_i$. Presence of j does not affect the relative order of execution of θ and θ' batches. The IIA property was used in [8] for proving a lower bound on the PoA of a class of job ordering mechanisms in the context of unrelated machines scheduling. Type ordering policies do not introduce delays in the execution of batches, but only decide their relative order of their execution, based on a batch's type index and setup time. They do not use the number of jobs within each batch; otherwise the IIA property may not be satisfied. Job lengths are used only for `Makespan`-wise scheduling within batches. Hence type ordering mechanisms function obliviously of "hidden" players with zero job lengths.

We prove next existence of PNE for any number of machines, and SE for $m = 2$ under type ordering mechanisms. An algorithm for finding PNE follows. Let $o(i)$ be the ordering of types on machine i, and $O = \{o(i) | i \in \mathcal{M}\}$ be the set of all orderings of the mechanism. By $\prec_o$ denote the precedence relation of types, prescribed by $o \in O$. Let M_o be the set of machines that schedule according to $o \in O$. Initialize $o \in O$ arbitrarily, and **repeat** until all jobs are assigned:

1. Find the earliest type θ according to $\prec_o$, with at least one unassigned job.
2. Let j be the largest length unassigned job with $t_j = \theta$.

3. Pick $i \in \mathcal{M}$ minimizing completion time of j [2] (break ties in favor of $i \in M_o$).
4. **If** $i \in M_o$ set $s_j = i$ **else** switch ordering o to $o(i)$.

Theorem 4. *The scheduling game with setup times has pure Nash equilibria, under type ordering mechanisms.*

Proof. The algorithm terminates in polynomial time; once a job is assigned, it is never considered again and within every $O(m+n)$ iterations some job is always assigned. For any type θ, denote by $\hat{s}_\theta$ the partial assignment up to the time after the last job of type θ has been assigned. We show by contradiction that no job j has incentive to deviate under an assignment s returned by the algorithm.

Assume that j does have incentive to deviate from s_j, and let s' be the resulting assignment after deviation of j. At the time corresponding to the partial assignment $\hat{s}_{t_j}$, there is no type $\theta \neq t_j$ and machine i such that $\theta \in T_i(\hat{s}_{t_j})$ and $t_j \prec_{o(i)} \theta$. If it was the case, the first job of type $\theta \neq t_j$ assigned to i would have been chosen before jobs of type t_j were exhausted, which contradicts step 1. of the algorithm. Thus, batches of type t_j are scheduled - under $\hat{s}_{t_j}$ - last on all machines with $t_j \in T_i(\hat{s}_{t_j})$. Furthermore, if j wishes to deviate to a machine $i \neq s_j$, then $c_j(s) = c_j(\hat{s}_{t_j}) > C_i(\hat{s}_{t_j}) + \ell_j = c_j(s')$, if $t_j \in T_i(\hat{s}_{t_j})$, and $c_j(s) = c_j(\hat{s}_{t_j}) > C_i(\hat{s}_{t_j}) + w(t_j) + \ell_j = c_j(s')$, if $t_j \notin T_i(\hat{s}_{t_j})$. Let j' be the last job of type t_j assigned to machine s_j (it may be $j' = j$). Because $\ell_{j'} \leq \ell_j$, it is also $c_{j'}(\hat{s}_{t_j}) = c_j(\hat{s}_{t_j}) > C_i(\hat{s}_{t_j}) + \ell_{j'}$ or $c_{j'}(\hat{s}_{t_j}) = c_j(\hat{s}_{t_j}) > C_i(\hat{s}_{t_j}) + w(t_{j'}) + \ell_{j'}$ accordingly. By the time j' was assigned, the completion time of i was at most $C_i(\hat{s}_{t_j})$. This contradicts step 3. of the algorithm with respect to j'. $\square$

Theorem 5. *For the scheduling game with setup times under type ordering mechanisms, any pure Nash equilibrium is strong, when $m = 2$.*

Proof. Assume that s is PNE, but not SE, and let $J \subseteq \mathcal{J}$ be a coalition of jobs that have incentive to deviate jointly. Define $J_1 = \{j \in J | s_j = 1\}$, $J_2 = \{j \in J | s_j = 2\}$; since s is PNE, $J_1, J_2 \neq \emptyset$. Let θ_i be the earliest type according to $\prec_{o(i)}$ with jobs in J_i and denote by J'_i type θ_i jobs in J_i. Take two jobs $j_1 \in J'_1$, $j_2 \in J'_2$, and let s' be the resulting assignment after deviation.

CASE 1: $\theta_1 \neq \theta_2$. Since s is a PNE, it must be $\theta_1 \prec_{o(1)} \theta_2$ and $\theta_2 \prec_{o(2)} \theta_1$ because, if e.g. $\theta_2 \prec_{o(1)} \theta_1$, j_2 would have incentive to deviate unilaterally to machine 1, since it wishes to deviate jointly with coalition J. Hence $c_{j_2}(s') \geq c_{j_1}(s) - \sum_{j \in J'_1} \ell_j + \sum_{j \in J'_2} \ell_j + w(\theta_2)$ if J'_1 does not contain the entire batch of type θ_1 and $c_{j_2}(s') \geq c_{j_1}(s) - \sum_{j \in J'_1} \ell_j - w(\theta_1) + \sum_{j \in J'_2} \ell_j + w(\theta_2)$ otherwise. So, in the worst case, we get $c_{j_2}(s') \geq c_{j_1}(s) - \sum_{j \in J'_1} \ell_j - w(\theta_1) + \sum_{j \in J'_2} \ell_j + w(\theta_2)$. Similarly, $c_{j_1}(s') \geq c_{j_2}(s) - \sum_{j \in J'_2} \ell_j - w(\theta_2) + \sum_{j \in J'_1} \ell_j + w(\theta_1)$. Summing up these two inequalities, we obtain $c_{j_2}(s') + c_{j_1}(s') \geq c_{j_2}(s) + c_{j_1}(s)$ which is impossible since it must be $c_{j_2}(s') < c_{j_2}(s)$ and $c_{j_1}(s') < c_{j_1}(s)$.

CASE 2: $\theta_1 = \theta_2$. Then, in the worst case we obtain $c_{j_2}(s') = c_{j_1}(s) - \sum_{j \in J'_1} \ell_j + \sum_{j \in J'_2} \ell_j$ and $c_{j_1}(s') = c_{j_2}(s) - \sum_{j \in J'_2} \ell_j + \sum_{j \in J'_1} \ell_j$. The rest of the proof is similar to the previous case. $\square$

[2] j incurs processing load $w(t_j) + \ell_j$ if a t_j-type job is not already assigned to i.

We give in **Appendix B** an example of PNE that is not SE for a type ordering mechanism, when $m \geq 3$. The following result identifies performance limitations of type ordering mechanisms, due to lack of a priori knowledge of $\mathcal{T} \subseteq \mathcal{U}$.

Theorem 6. *The Price of Anarchy of the scheduling game with setup times is* $\frac{m+1}{2}$ *for every deterministic type ordering mechanism.*

Proof. For any deterministic type ordering mechanism, assume there is a subset $\mathcal{T} \subseteq \mathcal{U}$ of $k = 2m - 1$ types, say $\mathcal{T} = \{1, \cdots, 2m-1\}$, such that: all types of $\mathcal{T}$ are scheduled in order of ascending index in a machines and in order of descending index in $d = m - a$ machines. Then, there is a family of instances with $PoA \geq \frac{m+1}{2}$. Next we prove existence of $\mathcal{T}$. Set $w(\theta) = 1$ for all $\theta \in \mathcal{U}$. When $a = m$ or $d = m$, take an instance of m zero length jobs for each type $\theta \in \{1, \cdots, m\}$. Placing one job of each type on every machine yields a PNE with makespan m. An assignment of makespan 1 has all same-type jobs assigned to a dedicated machine, thus $PoA \geq m$. When $a \geq 1$ and $d \geq 1$, the instance has:

- a jobs of zero length for each type $\theta \in \{1, \cdots, m-1\}$
- d jobs of zero length for each type $\theta \in \{m+1, \cdots, 2m-1\}$
- $m-1$ jobs of of zero length and type m
- one job of length 1 and type m
- no jobs for $\theta \in \mathcal{U} \setminus \mathcal{T}$

Assign one job of type $\theta \in \{1, \cdots, m-1\}$ on each of the a *ascending* type index machines, and one job of type $\theta \in \{m+1, \cdots, 2m-1\}$ on each of the d *descending* type index machines. Put one job of type m on every machine. This is a PNE of makespan $m+1$. Placing all jobs of type $\theta \in \{i, 2m-i\}$ on machine i yields makespan 2. Thus it is $PoA \geq \frac{m+1}{2}$.

We show existence of $\mathcal{T}$ for sufficiently large universe $\mathcal{U}$. We use the fact that any sequence of n different real numbers has a monotone (not necessarily contiguous) subsequence of $\sqrt{n}$ terms (a corollary of Theorem 4.4, page 39 in [22]). By renaming types in $\mathcal{U}$ we can assume w.l.o.g. that $\mathcal{U}$ is ordered monotonically (index-wise) on machine 1, and set $T_1 = \mathcal{U}$. Then, there is $T_2 \subseteq T_1$ such that $|T_2| \geq \sqrt{|T_1|}$ and all the types of T_2 are ordered monotonically according to index, on machines 1 and 2. After $m-1$ applications of the corollary, we obtain a set $T_m \subseteq T_{m-1} \subseteq \cdots \subseteq T_1 = \mathcal{U}$ with $|T_m| \geq |\mathcal{U}|^{2^{1-m}}$ and all its types are scheduled monotonically to their index on every machine. We set $\mathcal{T} = T_m$, and take a universe $\mathcal{U}$ of types with $|\mathcal{U}| = (2m-1)^{2^{m-1}}$, to ensure existence of $\mathcal{T}$ with $k = |\mathcal{T}| = 2m-1$ types. □

Let us note that “longest” or “shortest batch first” policies are no more powerful than type ordering mechanisms; they reduce to them for zero length jobs.

6 An Optimal Type Ordering Mechanism

We analyze the PoA of a type ordering mechanism termed `AD` (short for *Ascending-Descending*), that schedules type batches by ascending type index on half of the machines, and by descending type index on the rest. If m is odd one of the policies in applied to one machine more. First we prove the following lemma.

Lemma 1. *Let $\mathcal{T}' \subseteq \mathcal{T}$ include types with non-zero setup times. If two jobs of the same type in $\mathcal{T}'$ play an ascending and a descending index machine respectively under the* `AD` *mechanism, their type batches are scheduled last on the respective machines.*

Proof. We show the result by contradiction. Let jobs x_1, x_2 with $t_{x_1} = t_{x_2} = \theta$ be assigned on the ascending and descending machines 1, 2 respectively. Assume that a job y, $t_y = \theta' \neq \theta$, is scheduled on 1 after type θ. Because s is a PNE, job x_2 does not decrease its completion time if it moves to machine 1; because y is scheduled after x_1 on 1:

$$c_{x_1}(s) \geq c_{x_2}(s) - \ell_{x_2}, \text{ and } c_y(s) \geq c_{x_1}(s) + w(\theta') + \ell_y \tag{2}$$

If y switches to processor M_2 then it will be scheduled before type θ, thus its completion time will be at most $c_{x_2}(s) - w(\theta) - \ell_{x_2} + w(\theta') + \ell_y$ if $\theta' \notin T_2(s)$ (and at most $c_{x_2}(s) - w(\theta) - \ell_{x_2} + \ell_y$ otherwise). In the worst case, we obtain:

$$c_y(s) \leq c_{x_2}(s) - w(\theta) - \ell_{x_2} + w(\theta') + \ell_y \tag{3}$$

By (2) and (3), $c_y(s) \leq c_y(s) - w(\theta) < c_y(s)$, a contradiction, because $\theta \in \mathcal{T}'$. □

The next result identifies upper bounds on the *PoA* of `AD`. A proposition that follows proves tightness, through lower bounds on the Price of Stability, the ratio of the *least* expensive PNE makespan over the optimum makespan. We take $k \geq 2$; `AD` is identical to `Makespan` for $k = 1$.

Theorem 7. *The Price of Anarchy of the* `AD` *mechanism for the scheduling game with setup times is at most $\frac{m+1}{2}$ when $m \leq k$ and at most $\frac{k+3}{2} - \epsilon$ ($\epsilon = \frac{k}{m}$ when m is even and $\epsilon = \frac{k-1}{m-1}$ otherwise), when $m > k$.*

Proof. Let s be the most expensive PNE assignment and $C(s) = C_1(s) = \max_i C_i(s)$. Let θ_0 be the type scheduled last on machine 1 and x a job with $t_x = \theta_0$. Define $\mathcal{T}'_C \subseteq \mathcal{T}'$ to be types with jobs assigned to both ascending and descending machines under s. Let $T'_A \subseteq T' \setminus T'_C$ and $T'_D \subseteq T' \setminus T'_C$ contain types exclusively assigned to ascending and descending machines respectively. Notice that at most one type $\theta_1 \in \mathcal{T}'_C$ may appear in at least $\frac{m}{2} + 1$ machines (when m even) and $\frac{m+1}{2}$ machines (when m odd); thus any type in $\mathcal{T}'_C \setminus \{\theta_1\}$ appears on at most $\frac{m}{2}$ machines (actually, $\frac{m-1}{2}$ machines when m is odd). We study two cases depending on whether θ_1 exists and whether it coincides with θ_0 or not.

CASE 1: $\theta_0 = \theta_1$ or θ_1 does not exist. Job x will not decrease its completion time by moving to machine p for $p = 2, \ldots, m$. If $M_{\theta_0}(s)$ are the indices of machines which contain type θ_0, then:

$$\forall p \in M_{\theta_0}(s),\ c_x(s) \leq C_p(s) + \ell_x \text{ and } \forall p \notin M_{\theta_0}(s),\ c_x(s) \leq C_p(s) + w(\theta_0) + \ell_x \tag{4}$$

To obtain the upper bound we sum up (4) for $p \in \{2, \ldots, m\}$, add $C_1(s)$ in the left and right hand part, and take $\sum_{\theta \notin \mathcal{T}'} w(\theta) = 0$. We will do this analysis below, collectively for cases 1 and 2.

CASE 2: $\theta_0 \neq \theta_1$ and θ_1 exists. Assume $\theta_0 < \theta_1$ and let R contain the indices of ascending machines which have at least one job of type θ_1 assigned (if $\theta_0 > \theta_1$, we consider the indices of descending machines). Let R be the indices of these machines and $R' \subseteq R$ be the indices of machines that are also assigned type θ_0 jobs (note that $\theta_0 \notin \mathcal{T}'_C$ if $R' \neq \emptyset$). If job x moves to a machine with index in $p \in R'$, the completion time of x becomes at most $C_p(s) - w(\theta_1) + \ell_x$ and $C_p(s) - w(\theta_1) + w(\theta_0) + \ell_x$ if $p \in R'' = R \setminus R'$. Since s is a PNE:

$$\forall p \in R',\ c_x(s) \leq C_p(s) - w(\theta_1) + \ell_x,\ \forall p \in R'',\ c_x(s) \leq C_p(s) - w(\theta_1) + w(\theta_0) + \ell_x \quad (5)$$

We will sum up inequalities (4) or (5) for $p \in \{2, \ldots, m\}$ depending on whether $p \in R$ or not. As in case 1 we add $C_1(s)$ to left and right hand parts and consider $\sum_{\theta \notin \mathcal{T}'} w(\theta) = 0$. Before summing note that when m is even, each type in $\mathcal{T}'_A \cup \mathcal{T}'_D$ has jobs assigned to at most $\frac{r}{2}$ machines, for $r = m$. When m is odd assume w.l.o.g. that there are $\frac{m+1}{2}$ descending machines. *We ignore one of them* - different than M_1 - in the summation (we assume $m \geq 3$; otherwise $m = 1$ and $C(s) = C(s^*)$). Then, in case 2, type θ_1 appears at most $\frac{r}{2}$ times, $r = m - 1$, in the remaining $m - 1$ machines.

$$\begin{aligned}
rC_1(s) &\leq \frac{r}{2}\left(\sum_{\theta \in \mathcal{T}'_A \cup \mathcal{T}'_D \setminus \{\theta_0\}} w(\theta) + \sum_{\theta \in \mathcal{T}'_C \setminus \{\theta_0\}} w(\theta)\right) + rw(\theta_0) + \sum_{j \in \mathcal{J}} \ell_j + (r-1)\ell_x \\
&= \frac{r}{2}\left(\sum_{\theta \in \mathcal{T}} w(\theta) + \sum_{j \in \mathcal{J}} \ell_j\right) + \frac{r}{2} w(\theta_0) + (r-1)\ell_x - \frac{r-2}{2} \sum_{j \in \mathcal{J}} \ell_j \qquad (6) \\
&\leq \frac{r}{2}\left(\sum_{\theta \in \mathcal{T}} w(\theta) + \sum_{j \in \mathcal{J}} \ell_j\right) + \frac{r}{2}\left(w(\theta_0) + \ell_x\right) \text{ since } \textstyle\sum_{j \in \mathcal{J}} \ell_j \geq \ell_x \qquad (7)
\end{aligned}$$

When $k \geq m$ we use (1a,b) with (7) to obtain $C_1(s) \leq \frac{m+1}{2} OPT$. When $k < m$ we rewrite (6) as:

$$C(s) \leq \frac{1}{r}\left(\sum_{\theta \in \mathcal{T}} w(\theta) + \sum_{j \in \mathcal{J}} \ell_j\right) + \left(\frac{1}{2} - \frac{1}{r}\right)\left(\sum_{\theta \in \mathcal{T}} w(\theta) + \ell_x\right) + \frac{1}{2}\left(w(\theta_0) + \ell_x\right) \quad (8)$$

Using (1b,c), we get $kC(s^*) \geq \sum_{\theta \in \mathcal{T}} w(\theta) + \ell_x$ and replacing $r = m$ and $r = m-1$ for even and odd m respectively, yields the stated bounds with respect to k. □

Proposition 1. *The Price of Stability of the scheduling game with setup times under the* AD *mechanism is $\frac{m+1}{2}$ when $k > m$ and $\frac{k+3}{2} - \epsilon$ ($\epsilon = \frac{k}{m}$ when m is even and $\epsilon = \frac{k-1}{m-1}$ otherwise) when $k \leq m$.*

Proof. For $k > m$ we use the same example as in the proof of theorem 6, but replace the zero length jobs with very small $\epsilon > 0$ length. For AD the described assignment for $a, d \geq 1$ applies, and it is a PNE with makespan $m + 1 + (m-1)\epsilon$; the socially optimum makespan has length $2 + m\epsilon$. In any PNE, all jobs of types 1 and $2m - 1$ will play exactly the strategies specified in the described PNE assignment, because a lower completion time is not achievable for them in any assignment. Inductively, jobs of types i and $2m-i$, $i = 2 \dots m-1$, follow the same practice, given the strategies of jobs of types $i-1$, $2m-i+1$. For the jobs of type m, the strategies described in the aforementioned assignment are best possible, given the strategies of all other jobs. Therefore the described PNE is unique, hence $PoS \to \frac{m+1}{2}$ for $\epsilon \to 0$. The same uniqueness argument holds when $k \leq m$, for the instances given below.

$k \geq 2$ **even.** There are $m = 2k$ machines, k ascending A_i and k descending D_i for $i = 1, \dots, k$. For each type $\theta \neq \frac{k}{2}+1$, $w(\theta) = 1$ and there are k jobs of this type with length ε. Finally, there are $k+1$ jobs of type $\frac{k}{2}+1$ with length 1 and $w(1+k/2) = 0$. Consider the state s where A_1 has a job of each type $1, \dots, \frac{k}{2}+1$, machine A_i, $i = 2, \dots, k$, has one job of each type $1, \dots, \frac{k}{2}$, and finally descending machine D_i, $i = 1, \dots, k$, has a job of each type $\frac{k}{2}+1, \dots, k$. s is a PNE and $C(s) = \frac{k}{2}+1+\frac{k}{2}\varepsilon$. A socially optimum assignment s^* is defined as follows. For each type $\theta \neq \frac{k}{2}+1$, a dedicated machine schedules all jobs of type θ. Thus, $k-1$ machines are busy and $k+1$ are free. Each job of type $\frac{k}{2}+1$ is scheduled on a dedicated machine out of the $k+1$ free ones. Then $C(s^*) = 1+k\varepsilon$. Since $m = 2k$, when ε tends to 0, we get: $PoA = \frac{k}{2}+1 = \frac{k+3}{2} - \frac{k}{m}$.

$k \geq 3$ **odd.** Take $m = k$ machines: $\frac{k+1}{2}$ of ascending index and $\frac{k-1}{2}$ of descending index. Each type has setup time 1 and the length of each job is ϵ. There are $\frac{k+1}{2}$ jobs for each of the first $\frac{k-1}{2}$ types, assigned to a distinct ascending index machine each. There are $\frac{k-1}{2}$ jobs for each of the last $\frac{k-1}{2}$ types, assigned to a distinct descending index machine each. The middle type (with index $\frac{k+1}{2}$) has k jobs, each assigned to a distinct machine. This assignment is a PNE and has makespan $\frac{k+1}{2}(1+\epsilon)$. In the socially optimum assignment we place all jobs of every type on a dedicated machine and achieve makespan $1+k\epsilon$ (type $\frac{k+1}{2}$). The ratio tends to $\frac{k+1}{2} = \frac{k+3}{2} - \frac{k-1}{m-1}$ as $\epsilon \to 0$. □

7 Open Problems

Notice that the universe of types $\mathcal{U}$ is required to be huge in the proof of Theorem 6 (double exponential). This size is non-realistic for most interesting practical settings. Is there a lower size of $\mathcal{U}$ that also yields $PoA \geq \frac{m+1}{2}$ for type ordering mechanisms? For example, the proof of Theorem 6 requires that $|\mathcal{U}| \geq 9$ when $m = 2$, although $|\mathcal{U}| \geq 3$ is enough. The performance of type ordering mechanisms is not fully characterized by theorem 6; there may be certain sizes of $|\mathcal{U}|$ below which these mechanisms may perform better. Another interesting issue to be examined, is when type ordering mechanisms are a priori aware of the subset of types $\mathcal{T}$ that corresponds to players $\mathcal{J}$. What is the impact of such an a priori knowledge to the achievable PoA by type ordering mechanisms? Finally, we have not considered in this paper any simply local mechanisms, or more challenging machine environments (uniformly related or unrelated machines). All these constitute very interesting aspects for future developments on the subject.

References

1. Aumann, R.J.: Acceptable points in games of perfect information. Pacific Journal of Mathematics **10** (1960) 381–417
2. Koutsoupias, E., Papadimitriou, C.H.: Worst-case Equilibria. In: Proc. STACS, Springer LNCS 1543. (1999) 404–413
3. Christodoulou, G., Koutsoupias, E., Nanavati, A.: Coordination Mechanisms. In: Proc. ICALP, Springer LNCS 3142. (2004) 345–357
4. Koutsoupias, E., Mavronicolas, M., Spirakis, P.G.: Approximate Equilibria and Ball Fusion. Theory of Computing Systems **36**(6) (2003) 683–693
5. Czumaj, A., Vöcking, B.: Tight bounds for worst-case equilibria. ACM Transactions on Algorithms **3**(1) (2007)
6. Christodoulou, G., Gourvès, L., Pascual, F.: Scheduling Selfish Tasks: About the Performance of Truthful Algorithms. In: Proc. COCOON, Springer LNCS 4598. (2007) 187–197
7. Immorlica, N., Li, L., Mirrokni, V.S., Schulz, A.: Coordination Mechanisms for Selfish Scheduling. In: Proc. WINE, Springer LNCS 3828. (2005) 55–69
8. Azar, Y., Jain, K., Mirrokni, V.S.: (almost) optimal coordination mechanisms for unrelated machine scheduling. In: Proc. ACM-SIAM SODA. (2008) 323–332
9. Caragiannis, I.: Efficient coordination mechanisms for unrelated machine scheduling. In: Proc. AMC-SIAM SODA. (2009) 815–824
10. Durr, C., Nguyen, T.K.: Non-clairvoyant Scheduling Games. In: Proc. SAGT (to appear), Springer LNCS. (2009)
11. Andelman, N., Feldman, M., Mansour, Y.: Strong price of anarchy. Games and Economic Behavior (to appear) (2009) 189–198
12. Epstein, L., van Stee, R.: The Price of Anarchy on Uniformly Related Machines Revisited. In: Proc. SAGT, Springer LNCS 4997. (2008) 46–57
13. Vöcking, B.: Selfish Load Balancing (Chapter 20). In: Algorithmic Game Theory. Cambridge University Press (2007) 517–542
14. Finn, G., Horowitz, E.: A linear time approximation algorithm for multiprocessor scheduling. BIT **19** (1979) 312–320
15. Schuurman, P., Vredeveld, T.: Performance guarantees of local search for multiprocessor scheduling. In: Proc. IPCO, Springer LNCS 2081. (2001) 370–382
16. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer (1999)
17. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W.H.Freeman & Co Ltd (1979)
18. Schuurman, P., Wöginger, G.J.: Preemptive scheduling with job-dependent setup times. In: Proc. ACM-SIAM SODA. (1999) 759–767
19. Chen, B.: A better heuristic for preemptive parallel machine scheduling with batch setup times. SIAM Journal on Computing **22** (1993) 1303–1318
20. Wöginger, G.J., Yu, Z.: A heuristic for preemptive scheduling with set-up times. Computing **49**(2) (1992) 151–158
21. Chen, B., Ye, Y., Zhang, J.: Lot-sizing scheduling with batch setup times. Journal of Scheduling **9**(3) (2006) 299–310
22. Jukna, S.: Extremal Combinatorics with Applications in Computer Science. Springer-Verlag (2001)

Appendix A: Proof of Theorem 1

We suppose that $\mathcal{J} = \{1, \ldots, n\}$. To every state s we associate a permutation π_s of the set of players such that $\pi_s(j) \leq \pi_s(j')$ if and only if $c_j(s) \geq c_{j'}(s)$. Obviously, such a permutation always exists (for instance we sort the jobs by non increasing completion time). In particular, $\pi_s^{-1}(1)$ is a job whose completion time is the makespan under s, i.e., $C(s) = c_{\pi_s^{-1}(1)}(s)$. Associate the vector $v_s = \langle c_{\pi_s^{-1}(1)}(s), c_{\pi_s^{-1}(2)}(s), \cdots, c_{\pi_s^{-1}(n)}(s) \rangle$ to every state s. In the following, v_s^i denotes the i-th coordinate of v_s. Given two assignments s and r, we say that v_r is *lexicographically smaller* than v_s iff $v_r^1 < v_s^1$ or there is an index $i^* > 1$ such that $v_r^i = v_s^i$ for $i \in \{1, \cdots, i^* - 1\}$ and $v_r^{i^*} < v_s^{i^*}$. We show by contradiction that, if s is the lexicographically smallest assignment, then it is a strong equilibrium. Let r be an assignment and $J \subseteq \mathcal{J}$ be a nonempty coalition such that $\forall j \in J \; s_j \neq r_j$ and $\forall j \notin J \; s_j = r_j$. Moreover, by construction of a coalition, we must get: $\forall j \in J \; c_j(s) > c_j(r)$. Let $c_{max} = \max_{j \in J} c_j(s)$, that is c_{max} is the maximum completion time of the jobs of the coalition J under state s. We prove the following properties:

(i) For any $j \in \mathcal{J}$ such that $c_j(s) > c_{max}$, $c_j(r) = c_j(s)$.
(ii) For any $j \in J$, $c_j(r) < c_{max}$.
(iii) If $j \in \mathcal{J} \setminus J$ and $c_j(s) = c_{max}$, then $c_j(r) \leq c_{max}$.
(iv) For $j \in \mathcal{J} \setminus J$ with $c_j(s) < c_{max}$, $c_j(r) < c_{max}$.

For (i). Let j be a player such that $c_j(s) > c_{max}$. Since $c_{max} \geq c_p(s)$ for all $p \in J$, we deduce that $j \in \mathcal{J} \setminus J$ and $s_j = r_j$. If at least one job of J, say p, moves to machine s_j then $c_p(s) \leq c_{max} < c_j(s) \leq c_p(r)$, contradiction with $c_p(r) < c_p(s)$. Then no job quit or moves to s_j, the completion time on that machine remains unchanged.

For (ii). If $j \in J$ then $c_j(r) < c_j(s)$ and $c_j(s) \leq c_{max}$.
For (iii). Since $j \in \mathcal{J} \setminus J$, it is $s_j = r_j$. If no job of J moves to machine s_j then $c_j(r) \leq c_j(s) = c_{max}$ (the completion of s_j cannot increase). If at least one job of J, say p, moves to s_j then $c_j(r) = c_p(r)$ and $c_p(r) < c_{max}$ by item (ii).

For (iv). The proof is quite similar to item (iii). By definition we have $s_j = r_j$, i.e. j stays on machine s_j. If no job of J moves to s_j, then $c_j(r) \leq c_j(s)$ (the completion of machine s_j cannot increase). If at least one job of J, say p, moves to s_j then $c_j(r) = c_p(r)$ and $c_p(r) < c_{max}$ by item (ii).

Items (i) to (iv) lead to $|\{i : c_i(r) = c_{max}\}| \leq |\{i : c_i(s) = c_{max}\}|$. Moreover, using item (ii), we deduce that $|\{i : c_i(r) = c_{max}\}| \neq |\{i : c_i(s) = c_{max}\}|$ since by construction there exists at least one job $j \in J$ with $c_j(s) = c_{max}$ (and $c_j(r) < c_{max}$ by item (ii)). Thus globally, items (i) to (iv) imply that v_r is lexicographically smaller than v_s (contradiction with the minimality of v_s) because $v_r^i = v_s^i$ when $v_r^i > c_{max}$ and $|\{i : v_r^i = c_{max}\}| < |\{i : v_s^i = c_{max}\}|$. □

Appendix B

Strong Price of Anarchy for 2 machines (Theorem 3)

We partition $\mathcal{T}$ in 3 groups $\mathcal{T}_A$, $\mathcal{T}_B$ and $\mathcal{T}_C$ each containing respectively types only present on M_1, types present on both M_1 and M_2, and types only present on M_2. Let $T_A = \sum_{\theta \in \mathcal{T}_A} w(\theta)$, $T_B = \sum_{\theta \in \mathcal{T}_B} w(\theta)$ and $T_C = \sum_{\theta \in \mathcal{T}_C} w(\theta)$. We partition $\mathcal{J}$ in 4 groups $\mathcal{J}_1 = \{j : t_j \in \mathcal{T}_A\}$, $\mathcal{J}_2 = \{j : t_j \in \mathcal{T}_B \text{ and } s_j = 1\}$, $\mathcal{J}_3 = \{j : t_j \in \mathcal{T}_B \text{ and } s_j = 2\}$ and $\mathcal{J}_4 = \{j : t_j \in \mathcal{T}_C\}$. Let J_1, J_2, J_3, J_4 denote the total loads of jobs on these sets respectively. W.l.o.g. assume $C_1(s) \geq C_2(s)$. Then $C_2(s) \leq OPT$ since otherwise all jobs would prefer the optimal assignment. We study the cases $\mathcal{J}_1 = \emptyset$ and $\mathcal{J}_2 \neq \emptyset$, $\mathcal{J}_2 = \emptyset$ and $\mathcal{J}_1 \neq \emptyset$ or $\mathcal{J}_1 \neq \emptyset$ and $\mathcal{J}_2 \neq \emptyset$ (we know that $\mathcal{J}_1 \cup \mathcal{J}_2 \neq \emptyset$).

CASE 1: $\mathcal{J}_1 = \emptyset$ and $\mathcal{J}_2 \neq \emptyset$. We have $C_1(s) = T_B + J_2$ and $C_2(s) = T_B + T_C + J_3 + J_4$. We know that $|\mathcal{J}_2| \geq 2$ (since otherwise $C_1(s) = OPT$ by inequality (1b)). If $|\mathcal{J}_2| \geq 3$ and $x \in \mathcal{J}_2$ is the job with the smallest processing length, then $3\ell_x \leq J_2 \leq C_1(s)$. Because s is also a PNE, job x does not benefit by switching to M_2, thus $C_2(s) + \ell_x \geq C_1(s)$. Then we deduce that $\frac{2}{3}C_1(s) \leq C_2(s) \leq OPT$.

Now, assume $|\mathcal{J}_2| < 3$, that is $\mathcal{J}_2 = \{x, y\}$. As previously, we deduce that $C_2(s) + \ell_x \geq C_1(s) = T_B + \ell_x + \ell_y$ (note that T_B may contain one or two types), which is equivalent to $\ell_y \leq T_C + J_3 + J_4$. Because of this we have $OPT \geq C_2(s) = T_B + T_C + J_3 + J_4 \geq T_B + \ell_y$. On the other hand, using inequality (1a), we get $2OPT \geq T_B + T_C + J_2 + J_3 + J_4$. Combining these two last inequalities, we deduce that $3OPT \geq 2T_B + T_C + J_2 + J_3 + J_4 + \ell_y$. Because job y does not benefit by switching to machine M_2, i.e., $C_2(s) + \ell_y \geq C_1(s)$, we deduce that $2T_B + J_2 + T_C + J_3 + J_4 + \ell_y = C_1(s) + C_2(s) + \ell_y \geq 2C_1(s)$, which is at most $3OPT$.

CASE 2: $\mathcal{J}_1 \neq \emptyset$ and $\mathcal{J}_2 = \emptyset$. We have $C_1(s) = T_A + J_1$ and $C_2(s) = T_C + J_4$. Let $x \in \mathcal{J}_1$ with a type t_x. Since s is a NE, we get that $C_1(s) \leq C_2(s) + w(t_x) + \ell_x$. Then $2C_1(s) \leq C_1(s) + C_2(s) + w(t_x) + \ell_x = T_A + J_1 + T_C + J_4 + w(t_x) + \ell_x$, which is at most $3OPT$, because $2OPT \geq T_A + T_C + J_1 + J_4$ by (1a) and $OPT \geq w(t_x) + \ell_x$ by (1b).

CASE 3: $\mathcal{J}_1 \neq \emptyset$ and $\mathcal{J}_2 \neq \emptyset$. The completion time on M_1 and M_2 are respectively $T_A + T_B + J_1 + J_2$ and $T_B + T_C + J_3 + J_4$. Using inequality (1a), we get $2OPT \geq T_A + T_B + T_C + J_1 + J_2 + J_3 + J_4$. We consider the following coalitional deviations:

– If all jobs of $\mathcal{J}_1$ that play M_1 switch simultaneously to M_2 then no job of $\mathcal{J}_1$ will benefit, because s is a strong equilibrium. This yields $T_A + T_B + J_1 + J_2 \leq T_B + T_C + J_3 + J_4 + T_A + J_1$, thus $J_2 \leq T_C + J_3 + J_4$.
– If all jobs of $\mathcal{J}_2$ that play M_1 switch simultaneously to M_2 then no job of $\mathcal{J}_2$ will benefit, because s is a strong equilibrium. This yields $T_A + T_B + J_1 + J_2 \leq T_B + T_C + J_3 + J_4 + J_2$, thus $T_A + J_1 \leq T_C + J_3 + J_4$.

We know that $C_2(s) \leq OPT$. This yields $OPT \geq T_B + T_C + J_3 + J_4$. We are ready to prove the upper bound:

$$\begin{aligned} 2C(s) &= 2(T_A + T_B + J_1 + J_2) \\ &\leq 2T_A + 2T_B + 2J_1 + J_2 + T_C + J_3 + J_4 \\ &\leq 2OPT + T_A + T_B + J_1 \\ &\leq 2OPT + T_B + T_C + J_3 + J_4 \leq 3OPT \end{aligned}$$

The inequalities follow by the two cases of coalitional deviation, the lower bound of $2OPT$, and by the latter lower bound for OPT respectively. For the lower bound of $SPoA$ take 4 jobs with $t_1 = t_2 = 1$, $t_3 = t_4 = 2$ and $\ell_1 = \ell_3 = 2$, $\ell_2 = \ell_4 = 1$. Let $w(1) = w(2) = 1$. If tasks 1 and 3 are on machine 1, tasks 2 and 4 on machine 2, then the state is a strong equilibrium with makespan $w_1+w_2+\ell_1+\ell_3 = 6$. If tasks 1 and 2 is on machine 1, tasks 3 and 4 is on machine 2, then the state is a social optimum with makespan $w_1+\ell_1+\ell_2 = w_2+\ell_3+\ell_4 = 4$. □

A PNE that is not SE under a Type Ordering Mechanism

Take $m = 3$ machines and $\mathcal{U} = \{\theta_1, \theta_2, \theta_3\}$. We give below the parameters associated with types in $\mathcal{U}$ and jobs of the instance:

- $w(\theta_1) = 0$, 2 jobs of type θ_1, j_1, j_2, of length 3 each.
- $w(\theta_2) = 2$, 2 jobs of type θ_2, j_3, j_4, of length 0 each.
- $w(\theta_3) = 3$, 2 jobs of type θ_3, j_5, j_6, of length 1 each.

We consider a mechanism that defines the following type orderings $o(i)$ on the machines $i = 1, 2, 3$:

- **Machine 1**: $\theta_1 \prec_{o(1)} \theta_2 \prec_{o(1)} \theta_3$
- **Machine 2**: $\theta_2 \prec_{o(2)} \theta_3 \prec_{o(2)} \theta_1$
- **Machine 3**: $\theta_2 \prec_{o(3)} \theta_3 \prec_{o(3)} \theta_1$

So $o(2) = o(3)$. A PNE s is given by the following assignment:

- **Machine 1**: j_1, j_2 (completion times 6)
- **Machine 2**: j_3, j_5 (completion times 2, 6)
- **Machine 3**: j_4, j_6 (completion times 2, 6)

But such an assignment is not a SE, because all jobs except for j_3, j_4 have incentive to jointly switch to the following assignment:

- **Machine 1**: j_5, j_6 (completion times 5)
- **Machine 2**: j_3, j_1 (completion times 2, 5)
- **Machine 3**: j_4, j_2 (completion times 2, 5)