Decentralized Throughput Scheduling^{*}

Jasper de Jong, Marc Uetz, and Andreas Wombacher

University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
{j.dejong-3, m.uetz, a.wombacher}@utwente.nl

Abstract. Motivated by the organization of online service systems, we study models for throughput scheduling in a decentralized setting. In throughput scheduling, we have a set of jobs j with values w_j , processing times p_j , and release dates r_j and deadlines and d_j , to be processed non-preemptively on a set of unrelated machines. The goal is to maximize the total value of jobs scheduled within their time window $[r_j, d_j]$. While several approximation algorithms with different performance guarantees exist for this and related models, we are interested in the situation where subsets of servers are governed by selfish players. We give a universal result that bounds the price of decentralization, in the sense that any local α -approximation algorithms, $\alpha \geq 1$, yield equilibria that are at most a factor $(\alpha + 1)$ away from the global optimum, and this bound is tight. For models with identical machines, we improve this bound to $\sqrt[3]{e}/(\sqrt[3]{e} - 1) \approx (\alpha + 1/2)$, which is shown to be tight, too. We also address some variations of the problem.

1 Model and Notation

We consider a non-preemptive scheduling problem with unrelated machines, to which we refer as *decentralized throughput scheduling problem* throughout the paper. The input of an instance $I \in \mathcal{I}$ consists of a set of jobs $\mathcal{J}$, a set of machines $\mathcal{M}$, and a set of players $\mathcal{N}$. Each job $j \in \mathcal{J}$ comes with a release time r_j , deadline d_j , nonnegative value w_j and processing time p_{ij} if scheduled on machine $i \in \mathcal{M}$. Machines can process only one job at a time. Job j is feasibly scheduled (on any of the machines) if its processing starts no earlier than r_j and finishes no later than d_j . For any set of jobs $S \subseteq \mathcal{J}$, we let $w(S) = \sum_{j \in S} w_j$ be the total value. Each player $n \in \mathcal{N}$ controls a subset of machines $M_n \subseteq \mathcal{M}$ and aims to maximize the total value of jobs that can be feasibly scheduled on its set of machines M_n . Here M_n , $n \in \mathcal{N}$, is a partition of the set of machines $\mathcal{M}$.

In this paper we are primarily interested in *equilibrium allocations*, which we define as a situation in which none of the players n can improve the total value of jobs that can be feasibly scheduled on its set of machines M_n by removing some of its jobs and adding some of the yet unscheduled jobs. Note that a player cannot make a claim on jobs that are scheduled on machines of other players. Let us denote by $w(S)$ the total weight of jobs in S , for $S \subseteq \mathcal{J}$. An equilibrium

^{*} Research supported by CTIT, Centre for Telematics and Information Technology, University of Twente, project “Mechanisms for Decentralized Service Systems”.

allocation is a (pure) Nash equilibrium in a strategic form game where player n 's strategies are the subsets of jobs $S_n \subseteq \mathcal{J}$ that can be feasibly scheduled on machines M_n , and valuations are $w(S_n)$, but with the condition that a strategy profile $(S_n)_{n \in \mathcal{N}}$ is feasible if and only if the sets $S_n, n \in \mathcal{N}$, are pairwise disjoint¹. We will refer to these allocations as Nash equilibrium (*NE*) allocations.

The question if a given player can improve, generally describes an NP-hard optimization problem. Therefore, we will also consider a relaxed equilibrium condition: We say an allocation is an α -approximate Nash equilibrium (α -*NE*) if none of the players n can improve the total value of jobs that can be feasibly scheduled on its set of machines M_n by a factor larger than α by removing some of its jobs and adding some of the yet unscheduled jobs. By the existence of constant factor approximation algorithms for (centralized) throughput scheduling, e.g. [2, 4], the players are thus equipped with polynomial time algorithms to indeed reach an α -*NE* in polynomial time, for certain, constant values of α .

Our main focus will be the analysis of the price of anarchy (PoA) [14], lower bounding the quality of any Nash equilibrium relative to the quality of a globally optimal solution. More specifically, we are interested in the ratio

$$\text{PoA} = \max_{I \in \mathcal{I}} \max_{NE \in NE(I)} \frac{w(OPT)}{w(NE)}, \quad (1)$$

where OPT denotes a globally optimal allocation for I , and $NE(I)$ denotes the set of all Nash equilibria of instance I . Note that OPT is a Nash equilibrium, too, hence the price of stability, as proposed in [1], trivially equals 1.

2 Motivation, Related Work and Contribution

Our motivation to study this problem is to analyze the performance of decentralized service systems, where jobs are posted, e.g. on an online portal, and service providers can select these on an take-it-or-leave-it basis. The problem we address can be seen as a stylized version of coordination problems that appear in several application domains. We give three examples: (1) When operating micro grids for decentralized energy production and consumption, the goal is to consume locally produced energy as much as possible. Here, jobs can be defined as the operation of appliances, e.g. operating a washing machine), typically bounded by a time window and attached with a certain \$-value. Machines, on the other side, are local energy producers like PV-panels, micro CHPs or local micro windmills [3, 12]. (2) In cloud computing, service providers such as Amazon and Google provide an infrastructure service, that is, provide a virtual machine with a specific service level for a certain period of time. The aim of a federated cloud computing environment, e.g. [18], is to “coordinate load distribution among different cloud-based data centers in order to determine optimal location for hosting application services”. (3) In private car sharing portals like Tamycar [17] or Autonetzer [16],

¹ This is achieved by introducing externalities, letting the valuation of player n be $-\infty$ whenever S_n is not disjoint with the sets S_{-n} chosen by all other players $-n$.

clients post their requests to rent a car for a certain time period, and the price they are willing to pay. Car owners (private or not) in the neighborhood can select these requests and rent their car(s).

The underlying non-strategic optimization problem is known in the optimization literature and sometimes referred to as throughput scheduling. See for example [2], and follow-up papers, e.g. [4]. In the 3-field notation of [8], the problem reads $R|r_j|\sum w_j U_j$, where R denotes the unrelated machine model, r_j specifies that there are nontrivial release dates, and the objective is to minimize the total weight of late jobs (which is equivalent to the maximization objective considered here). Approximation algorithms for several versions of the problem have been discussed in the literature (e.g., with or without weights, identical or unrelated machines), most notably [2, 4]. Special cases that are of particular interest are the single machine case with unit weights and zero release dates, solved in polynomial time by the Moore-Hodgson algorithm [13], and the case with identical machines and unit processing times, which can be cast and solved as an assignment problem [5]. To the best of our knowledge, the decentralized version that we propose here has not been addressed before.

Our main contribution lies in the the informal claim that, in general, the price of decentralization is very moderate: If local decisions of all players are approximately optimal with relative performance guarantee α , then any equilibrium allocation is not worse than an $(\alpha + 1)$ -fraction of the global optimum. We improve this to $\approx (\alpha + 1/2)$ when all machines are identical, and the equilibria are obtained in a special, sequential way which will be explained later. In fact, except for the identical machine case, the proofs for the upper bounds are fairly simple, yet surprisingly universal. Most work lies in the corresponding tight lower bound constructions.

3 A First Encounter

Example 1. There are two players $\mathcal{N} = \{1, 2\}$, each controlling exactly one of two related machines $\mathcal{M} = \{1, 2\}$, with machine speeds $s_1 = 1, s_2 = \frac{2}{3}$, respectively². There are two jobs $\mathcal{J} = \{1, 2\}$ with processing times $p_1 = p_2 = 1$, deadlines $d_1 = 1, d_2 = \frac{3}{2}$ and values $w_1 = w_2 = 1$. Release times are 0. When job 1 is allocated to machine 1 and job 2 to machine 2, both jobs can meet their respective deadlines. This is obviously an optimal allocation. However when job 2 is allocated to machine 1, only one job can be scheduled before its deadline. See also Figure 1. Note that both allocations are a Nash equilibrium. Now $w(OPT)/w(NE) = 2/1 = 2$ for the second allocation, and we see from this simple example that

$$\text{PoA} \geq 2$$

in (1), even for the case of related machines, unit weights, unit processing times and zero release dates. $\square$

² The related machine model is a special case of the unrelated machine model by letting $p_{ij} = \frac{p_j}{s_i}$.

Fig. 1. An optimal solution and a Nash equilibrium in case of related machines

In the next section(s) we argue that $\text{PoA} = 2$ for this problem, and even for the general decentralized throughput scheduling problem.

4 Upper Bound for Equilibrium Allocations

Players $n \in \mathcal{N}$ face the computational complexity of computing an optimal subset of jobs that can be feasibly scheduled on their set of machines M_n , given the jobs that are available to them. We come back to this issue in the next section, where we only require that players' behave approximately optimal. For the time being, we ignore this issue, and first show that the trivial Example 1 is already the worst case, even for the general model with unrelated machines, nontrivial release times, arbitrary processing times, and arbitrary job values w_j .

Theorem 1. *The decentralized throughput scheduling problem has $\text{PoA} = 2$. The lower bound $\text{PoA} \geq 2$ even holds for the special case of unit values w_j , unit processing times p_j , related machines and in the absence of release dates.*

Proof. $\text{PoA} \geq 2$ follows from Example 1. The proof of the upper bound follows the same line as that of [2, Thm 3.2]. Take any instance with optimal solution OPT and Nash equilibrium NE and let NE_n and OPT_n , $n \in \mathcal{N}$, be the allocations in NE and OPT , respectively. Now assume for contradiction that $w(OPT) > 2w(NE)$ ³. Let Y be the set of feasibly scheduled jobs in both OPT and NE , and let N be the set of jobs feasibly scheduled in OPT but not in NE . Clearly $w(Y) \leq w(NE) < \frac{1}{2}w(OPT) = \frac{1}{2}(w(Y) + w(N))$. Hence $w(Y) < w(N)$, and therefore,

$$w(NE) < w(N).$$

But now observe that the jobs N are available to the players in the allocation of NE . By additivity of the objective across players, i.e., $w(N) = \sum_n w(N \cap OPT_n)$,

³ In a slight abuse of notation, we use OPT and NE to also denote the set of feasibly scheduled jobs in the respective solutions.

we conclude that there exists at least one player n for which $w(NE_n) < w(N \cap OPT_n)$, and as $N \cap OPT_n$ can be feasibly scheduled on machines M_n , this contradicts the definition of a Nash equilibrium. $\square$

5 Upper Bound for Approximate Equilibrium Allocations

The result in Theorem 1 implicitly assumes that players are able to verify that their strategies are at equilibrium. The players' problem, however, is polynomially solvable only for special cases, for instance when jobs have unit values and zero release dates, and when each player controls exactly a single machine. In that case, the Moore-Hodgson algorithm [13] maximizes the total number of early jobs. But already here, when players control more than one machine, the players' problem is NP-complete as generalization of the makespan minimization problem on parallel machines, which is NP-complete [7]. When the machines M_n of a player n are identical, and jobs have unit processing times, the players' problem can be cast and solved as an assignment problem [5]. In most other cases, the players' problem is NP-complete. For example, for a player that controls a single machine, when jobs have zero release dates, but arbitrary processing times and weights, the problem is (weakly) NP-hard [10, 9]. Adding nontrivial release dates makes the problem strongly NP-hard [11].

Therefore, we also consider a relaxed equilibrium concept, assuming that players moves are only approximately optimal. This leads to the concept of α -approximate equilibria, which has lately been discussed in the literature on computing Nash equilibria, for instance in the context of congestion games [15]. Approximate Nash equilibria can also be defined by allowing additive deviations instead of relative deviations, e.g. [6], but given that there exist constant-factor approximation algorithms for throughput scheduling, e.g. [2, 4], it appears more reasonable to work with relative bounds here. We say the allocation is an α -approximate Nash equilibrium, or α -NE, if no player n can improve the total value of its jobs by a factor larger than α , by removing some of its jobs and adding some of the yet unscheduled jobs. That said, we obtain the following generalization of Theorem 1.

Theorem 2. *The decentralized throughput scheduling problem has $PoA = \alpha + 1$, assuming that equilibrium allocations are α -approximate Nash equilibria. The lower bound $PoA \geq \alpha + 1$ even holds for the special case of unit values w_j , unit processing times p_j , related machines and in the absence of release dates.*

Proof. The upper bound follows in the same way as in the proof of Theorem 1, and is therefore moved to the appendix. For the proof that $PoA \geq \alpha + 1$, consider the following instance with unit processing times and values, related machines, and zero release dates. Consider any rational $\alpha = p/q$, $p \geq q$, and assume players deploy an α -approximation each. There are $q + 1$ players $\mathcal{N}$, each controlling one of $q + 1$ machines $\mathcal{M} = \{1, \dots, q + 1\}$ with machine speeds $s_1 = 1$ and $s_2 = s_3 = \dots = s_{q+1} = 1/(p + \varepsilon)$ for some $0 < \varepsilon < 1$. There are $p + q$ jobs $\mathcal{J} = \{1, \dots, p + q\}$. All jobs have unit processing times and unit weights.

Fig. 2. Optimal solution and α -NE in case of related machines.

Jobs $J_1 = \{1 \dots p\}$ have deadline p . Jobs $J_2 = \{p+1 \dots p+q\}$ have deadline $p+\epsilon$. Therefore machine 1 can schedule at most p jobs. Machines $2 \dots q+1$ can schedule no jobs from J_1 and only one job from J_2 each. In *OPT* all $p+q$ jobs are feasibly scheduled: jobs J_1 on Machine 1 and each of machines $2, \dots, q+1$ has one job from J_2 . Now consider the α -NE where only q jobs are scheduled: Machine 1 schedules all q jobs from J_2 , and machines $2 \dots q+1$ schedule no job. This is indeed an α -approximate Nash equilibrium, as machine 1 can schedule at most $p = \alpha q$ jobs, and since all jobs from J_2 are scheduled on machine 1, machines $2 \dots q+1$ cannot improve from their 0 jobs either. See Figure 2 for an illustration. We conclude that $\text{PoA} \geq (p+q)/q = \alpha + 1$. $\square$

6 Sequential Moves of Players

Up to this point our result on the quality of equilibria is universal. Clearly, determining an order in which players make a move, may improve the resulting equilibria. We say an equilibrium allocation (α -approximate Nash equilibrium) is *sequential* when it can be attained by starting with an empty schedule, and then sequentially according to some fixed order, allowing each player n to make a move, that is, selecting the set of jobs S_n . The following example shows that indeed, not all (α -approximate) Nash equilibria can be obtained sequentially.

Example 2. There are n players controlling one of n machines $\mathcal{M} = \{1, \dots, n\}$ each, with unit speeds, and $2n-1$ jobs $\mathcal{J} = \{1, \dots, 2n-1\}$ with unit weights. Jobs $J_1 = \{1 \dots n\}$ have processing time $1/n$ and deadline 1. Jobs $J_2 = \{n+1 \dots 2n-1\}$ have processing time 1 and deadline 1. In *OPT* machine 1 schedules jobs J_1 and machines $2 \dots n$ schedule jobs J_2 . In *NE* each machine schedules one job from J_1 . Note that *NE* is indeed an equilibrium: No machine can schedule 2 jobs without exchanging jobs with another machine. See Figure 3 for an illustration. For this instance $w(\text{OPT})/w(\text{NE}) = \frac{2n-1}{n} \rightarrow 2$ for $n \rightarrow \infty$. This Nash equilibrium cannot be attained sequentially. The first player to move would necessarily schedule all jobs from J_1 on his machine. $\square$

Fig. 3. Nash equilibrium and an optimal solution in case of identical machines

This example also shows that the identical machine model does not allow an improvement of the result of Theorem 1.

6.1 A Negative Result for Slowest Machines First

For the lower bound in Example 1, we use related machines (i.e., machines with speeds). In that example, *OPT* is obtained as a Nash equilibrium once we assume that players with slow machines are allowed to select their jobs first. However, as we show next, this does not improve the price of anarchy in general.

Theorem 3. *Consider the special case with related machines, unit weights, arbitrary processing times and zero release dates. Sequential Nash equilibria, where players move in order of increasing machine speed, still have $PoA = 2$.*

Proof. There are n players controlling exactly one of n machines $\mathcal{M} = \{1, \dots, n\}$ with machine speeds $s_j = 3n+j-1, \forall j \in \mathcal{M}$. There are $2n$ jobs $\mathcal{J} = \{1, 2, \dots, 2n\}$. All jobs have unit weights. Jobs $J_1 = \{1 \dots n\}$ have processing times $p_i = n+i$ and deadlines $d_i = \frac{n+i}{3n+i-1}$. Jobs $J_2 = \{n+1 \dots 2n\}$ have processing times $p_i = 1$ and deadlines $d_i = 1$. We may assume that players use optimal algorithms, that is, the Moore-Hodgson algorithm. In *OPT*, $2n-1$ jobs are scheduled: Machine 1 schedules only job $n+1$. Machines $2 \dots n$ schedule two jobs each. One from J_1 , and one from J_2 , where machine i schedules job $i-1$ followed by job $n+i$, for all $i = 2, \dots, n$. In *NE* only n jobs are scheduled, as follows. Machine i schedules job i for all $i \in \mathcal{M}$. See Figure 5 in the appendix for an example. Note that this is a Nash equilibrium corresponding to a sequential schedule where players move in order of increasing speed. To see this, we show that, when scheduled in order of increasing speed, no machine could have scheduled 2 jobs. From the jobs in J_1 , machine i could have only scheduled job i , since jobs $j < i$ have already been scheduled by slower machines, and jobs $j > i$ do not meet their deadline on machine i . Any job from J_2 scheduled after i does not meet its deadline, and any job from J_2 scheduled before i causes i to not meet its deadline. Also,

no machine could have scheduled 2 jobs from J_2 since the second job will have completion time $4n$ and deadline 1. We get $w(OPT)/w(NE) = (2n - 1)/n \rightarrow 2$ for $n \rightarrow \infty$. $\square$

7 Identical machines

In this section we improve our previous result for the case of identical machines. Recall that in the identical machine model, the processing time of each job is independent on the machine on which it is scheduled.

7.1 Identical machines: Lower Bound

We start with a lower bound on the price of anarchy for sequential equilibria.

Theorem 4. *PoA $\geq \sqrt[n]{e}/(\sqrt[n]{e} - 1)$ for identical machines, even in the restricted model where the α -approximate NE is obtained sequentially, and for unit processing times, unit weights, and zero release dates.*

Proof. We give a corresponding example. Again there are n players controlling one of n identical machines $\mathcal{M} = \{1, \dots, n\}$. There are n^2 jobs $\mathcal{J} = \{1, 2, \dots, n^2\}$ with unit processing times and unit weights. Jobs have deadlines $\delta \in \{1, 2, \dots, n\}$ and for each deadline, there are n jobs with this deadline, that is, for all δ , $d_j = \delta$ for $j = 1 + (\delta - 1)n, \dots, \delta n$. We refer to jobs as δ -jobs, $\delta = 1, \dots, n$. In Figure 4 we see an instance for $n = 5$ and $\alpha = 2$ (that is, machines use a 2-approximation). For each of the jobs, the number displayed on it corresponds to its deadline. In OPT , every machine schedules n jobs with different deadlines, ordered by increasing deadline. Therefore $w(OPT) = n^2$. We construct an α -approximate Nash Equilibrium, say S , by sequential moves of players as follows. For every machine $i = 1, \dots, n$ in this order, we find the maximum number of jobs that can be scheduled, say O_i , and let S_i be the $\lceil |O_i|/\alpha \rceil$ jobs with the largest deadlines (which are the most flexible jobs). For example, for $n = 5$ and $\alpha = 2$, $w(NE^\alpha) = 3 + 3 + 2 + 2 + 2 = 12$ as can be seen in Figure 4. We bound $w(S)$ in the following way. In S , denote by $r_\delta(i)$ the fraction of δ -jobs on machine i , relative to the total number of jobs on machine i . Let $r_\delta = \sum_i r_\delta(i)$. In our example, $r_4 = 0 + \frac{1}{3} + 1 + 1 + 0$. Observe that $\sum_\delta r_\delta = n$ for any allocation. In S , any machine scheduling a δ -job, does not schedule any $(\delta + 2)$ -jobs, or jobs with larger deadlines, hence it schedules at most $\lceil (\delta + 1)/\alpha \rceil \leq (\delta + 1 + \alpha)/\alpha$ jobs. Therefore, for any δ for which all n δ -jobs are allocated in S , we get $r_\delta \geq n\alpha/(\delta + 1 + \alpha)$.

Now, for some $\delta' \geq 0$, by construction of the allocation we have that all n δ -jobs with $\delta = n - \delta', \dots, n$ are fully scheduled, as well as a fraction of the $(n - \delta' - 1)$ -jobs. We get

$$n \geq \sum_{\delta=n-\delta'}^n r_\delta \geq \sum_{\delta=n-\delta'}^n \frac{n\alpha}{\delta + 1 + \alpha} \geq \int_{\delta=n-\delta'}^n \frac{n\alpha}{(\delta + 1 + \alpha) + 1} d\delta. \quad (2)$$

<i>OPT</i>					<i>α-NE</i>		
1	2	3	4	5	5	5	5
1	2	3	4	5	4	5	5
1	2	3	4	5	4	4	
1	2	3	4	5	4	4	
1	2	3	4	5	3	3	

Fig. 4. Optimal solution and sequential 2-NE in case of identical machines. Numbers denote job deadlines.

Because the last term is upper bounded by n , we can derive an upper bound on δ' . In fact, basic calculus shows that

$$\delta' > \frac{(n+2+\alpha)(\sqrt[n]{e}-1)}{\sqrt[n]{e}} \Rightarrow \int_{\delta=n-\delta'}^n \frac{n\alpha}{(\delta+1+\alpha)+1} d\delta > n,$$

which together with (2) yields that $\delta' \leq \frac{(n+2+\alpha)(\sqrt[n]{e}-1)}{\sqrt[n]{e}}$. Because all δ -jobs with $\delta \geq n - \delta'$ are scheduled, we conclude that

$$w(S) \leq (\delta' + 1)n \leq \frac{(n+4+\alpha)(\sqrt[n]{e}-1)}{\sqrt[n]{e}} \cdot n.$$

We see that

$$\frac{w(OPT)}{w(S)} \geq \frac{n \sqrt[n]{e}}{(n+4+\alpha)(\sqrt[n]{e}-1)} \rightarrow \frac{\sqrt[n]{e}}{\sqrt[n]{e}-1} \quad \text{for } n \rightarrow \infty,$$

and the claim follows $\square$

Note that the lower bound construction assumes that players choose the most flexible jobs first, which seems reasonable. The bound also holds for the case with unit processing times, where we can even assume that the players use optimal strategies [5]. Assuming that players indeed act locally optimal, the result shows that the price of decentralization can be as high as $e/(e-1) \approx 1.58$.

7.2 Identical Machines: Upper Bound

To derive a matching upper bound for identical machines, when moves are sequential, we use a proof idea from Bar-Noy et al. [2] in their analysis of k -GREEDY.

Theorem 5. $PoA \leq \sqrt[m]{e}/(\sqrt[m]{e} - 1)$ for identical machines and sequential, α -approximate Nash equilibria.

Proof. Let us assume there are n players and m machines. Each player i controls m_i machines. We use a proof idea of [2, Thm 3.3], but need a nontrivial generalization to make it work also for the case where players control several machines. The crucial definition is: For a given set of jobs J , denote by γ the average approximation ratio per machine that is achieved by a single player i in comparison to $OPT(J)$, where $OPT(J)$ denotes the optimum solution on n machines and jobs J . Therefore, when $S_i \subseteq J$ is the set of jobs scheduled by player i , $w(S_i) \geq m_i w(OPT(J))/\gamma$.

Suppose we have already determined $S_1, \dots, S_{i-1}$. The jobs in OPT that have not been scheduled yet have a value $w(OPT) - \sum_{j=1}^{i-1} w(S_j)$. Therefore, by definition, $w(S_i) \geq m_i(w(OPT) - \sum_{j=1}^{i-1} w(S_j))/\gamma$. Add $\sum_{j=1}^{i-1} w(S_j)$ to both sides to get

$$\sum_{j=1}^i w(S_j) \geq \frac{m_i w(OPT)}{\gamma} + \frac{\gamma - m_i}{\gamma} \sum_{j=1}^{i-1} w(S_j) \quad (3)$$

We prove by induction on i that

$$\sum_{j=1}^i w(S_j) \geq \frac{\gamma^{m'_i} - (\gamma - 1)^{m'_i}}{\gamma^{m'_i}} w(OPT),$$

where $m'_i = \sum_{j=1}^i m_j$. When $i = 1$, we show $w(S_1) \geq \frac{\gamma^{m_1} - (\gamma - 1)^{m_1}}{\gamma^{m_1}} w(OPT)$ in the appendix (Lemma 2). Assume the claim holds for $i - 1$. Applying the induction hypothesis to (3) we get

$$\sum_{j=1}^i w(S_j) \geq \frac{m_i w(OPT)}{\gamma} + \frac{\gamma - m_i}{\gamma} \cdot \frac{\gamma^{m'_{i-1}} - (\gamma - 1)^{m'_{i-1}}}{\gamma^{m'_{i-1}}} w(OPT)$$

This yields the inductive claim, as shown in the appendix (Lemma 3). Hence we get for $i = n$ (see also [2, Thm 3.3])

$$w(\alpha\text{-NE}) = \sum_{j=1}^n w(S_j) \geq \frac{\gamma^m - (\gamma - 1)^m}{\gamma^m} w(OPT).$$

Next observe $\gamma \leq m\alpha$. This follows because $w(S_1) \geq OPT_1/\alpha \geq m_1 OPT/(m\alpha)$, for any instance. Here OPT_1 denotes the optimal schedule for player 1. We get

$$PoA \leq \frac{\gamma^m}{\gamma^m - (\gamma - 1)^m} \leq \frac{(m\alpha)^m}{(m\alpha)^m - (m\alpha - 1)^m} \leq \frac{\sqrt[m]{e}}{\sqrt[m]{e} - 1}, \quad (4)$$

where the second inequality in (4) follows from basic calculus, and the third inequality follows because the right hand side is exactly the limit for $m \rightarrow \infty$, and the series $b_m = (m\alpha)^m / ((m\alpha)^m - (m\alpha - 1)^m)$ is monotone in m , with $b_1 = \alpha \leq \sqrt[m]{e}/(\sqrt[m]{e} - 1)$. $\square$

Theorems 4 and 5 yield $\text{PoA} = \sqrt[3]{e}/(\sqrt[3]{e} - 1)$ for identical machines with sequential moves of the players. Basic calculus shows that $\alpha + \frac{1}{2} \leq \sqrt[3]{e}/(\sqrt[3]{e} - 1) \leq \alpha + \frac{1}{e-1}$ for $\alpha \geq 1$. Also, for $\alpha \rightarrow \infty$ this value approaches $\alpha + \frac{1}{2}$. Note that for $\alpha = 1$, $\text{PoA} = e/(e - 1) \approx 1.58$.

7.3 Universal Bound for Identical Machines

In this section we show that the bound $\text{PoA} = \sqrt{e}/(\sqrt{e} - 1)$ for identical machines with unit processing times, unit weights and zero release dates is universal, i.e., it also holds without requiring that the Nash equilibria are obtained sequentially. Note that, due to unit processing times, it is reasonable to assume that players' strategies are optimal. Also note that, once processing times are arbitrary, Example 2 gives a lower bound of 2, matching the universal upper bound of 2 in Theorem 1.

Lemma 1. *Consider any instance with identical machines, unit processing times, unit weights and zero release dates. There exists a worst case Nash Equilibrium that is obtained sequentially.*

Proof. We construct a specific Nash equilibrium (sequentially) and show that this must be worst case. Consider the greedy algorithm $\mathfrak{G}$ that considers jobs in order of non-increasing deadlines, and greedily assigns the jobs, first to the machines of player 1, then the remaining jobs to the machines of player 2, etc. Clearly, this gives a Nash equilibrium. Observe that, once a job j can't be scheduled on machines of player i , the remaining jobs from $j + 1, \dots, n$ can't be scheduled either. However when j can be scheduled on some machine of player i , instead of j any of the remaining jobs from $j + 1, \dots, n$ could be scheduled as well. This follows because all of the jobs sitting in front of j , if any, have deadlines at least that of j , and thus can be shifted one time slot if j is removed. So for any other job $h > j$, a free time slot at time 0 can be made available.

Assume the equilibrium allocation S resulting from $\mathfrak{G}$ is not worst case. Then, specifically there exists a job j , scheduled on some machine m of some player i , which is the first job for which the following holds: when jobs $1 \dots j$ are allocated according to $\mathfrak{G}$, it is not possible to augment this partial allocation to a Nash equilibrium worse than S (using any allocation of the remaining jobs). However when jobs $1 \dots j - 1$ are allocated according to $\mathfrak{G}$, this can be augmented to a worst case Nash equilibrium W . Consider the partial allocation of $\{1, \dots, j - 1\}$ according to $\mathfrak{G}$. Since j can be feasibly allocated to a machine of player i , so can jobs $j + 1, \dots, n$. Therefore, since W is a Nash equilibrium, some job $h > j$ is allocated to a machine of player i . But now we can remove h from i , and replace it by j . If h (or any other unscheduled job) can be feasibly allocated to the player where j used to be, we do so. In any case, the result is a worst-case Nash equilibrium as well. However, this worst-case Nash equilibrium has jobs $1, \dots, j$ allocated according to $\mathfrak{G}$, a contradiction. We conclude that the allocation produced by $\mathfrak{G}$ is a worst-case Nash equilibrium. $\square$

We conclude with the following theorem.

Theorem 6. *The decentralized throughput scheduling problem with identical machines, unit processing times, unit weights and zero release dates has $PoA = \sqrt{e}/(\sqrt{e} - 1)$.*

Acknowledgements. Thanks to Johann Hurink for some helpful discussions on the context of the problem.

References

1. E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. In: Proc. 45th Symp. Foundations of Computer Science (FOCS), 295–304, IEEE, 2004.
2. A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Approximating the throughput of multiple machines in real-time scheduling. *SIAM J. Comp.*, 31(2): 331–52, 2001.
3. V. Bakker, M.G.C. Bosman, A. Molderink, J.L. Hurink and G.J.M. Smit. Demand side load management using a three step optimization methodology. In: 1st IEEE Int. Conf. on Smart Grid Communications, 431–436. IEEE, 2010.
4. P. Berman and B. Dasgupta. Multi-phase Algorithms for Throughput Maximization for Real-Time Scheduling. *J. Combinatorial Optimization*, 4(3): 307–323, 2000.
5. P. Brucker. *Scheduling Algorithms* (4th ed.), Springer Verlag, Berlin, 2004
6. Progress in Approximate Nash Equilibria. In: Proceedings 8th ACM Conference on Electronic Commerce (EC), 355–358. ACM, 2007.
7. M.R. Garey and D.S. Johnson. “Strong” NP-completeness results: motivation, examples, and implications. *Journal of the ACM*, 25(3):499–508, 1978.
8. R. Graham, E. Lawler, J. Lenstra, and A. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5(2):287–326, 1979.
9. R.M. Karp. Reducibility among combinatorial problems. In: *Complexity of computer computations*, R.E. Miller and J.W. Thatcher (eds.), 85–103. Plenum Press, New York, 1972.
10. E.L. Lawler and J.M. Moore. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 16:77–84, 1969.
11. J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of Machine Scheduling Problems. *Annals Disc. Math.*, 1:343–362, 1977.
12. A. Molderink, V. Bakker, M.G.C. Bosman, J.L. Hurink and G.J.M. Smit. Management and Control of Domestic Smart Grid Technology. *IEEE transactions on Smart Grid*, 1(2): 109–119, 2010.
13. J.M. Moore. An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science*, 15:102–109, 1968.
14. E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In: *Proceedings 16th STACS, Lecture Notes in Computer Science 1563*, 404–413. Springer, 1999.
15. E. Skopalik and B. Vöcking. Inapproximability of pure Nash equilibria. In: Proc. 40th Symp. on Theory of Computing (STOC), 355–364, ACM, 2008.
16. www.autonetzer.com
17. www.tamyca.com
18. R. Buyya, R. Ranjan, and R. Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In C.-H. Hsu, L. Yang, J. Park, and S.-S. Yeo (eds.) *Algorithms and Architectures for Parallel Processing*, Lecture Notes in Computer Science 6081, 13–31. Springer, 2010.

8 Appendix

Theorem 2. The decentralized throughput scheduling problem has $\text{PoA} = \alpha + 1$, given that equilibrium allocations are α -approximate Nash equilibria.

Proof. We still have to argue that $\text{PoA} \leq \alpha + 1$. Take any instance with optimal solution OPT and α -approximate Nash equilibrium NE and let NE_n and OPT_n , $n \in \mathcal{N}$, and be the allocations in NE and OPT , respectively. Now assume for contradiction that $w(OPT) > (\alpha + 1)w(NE)$. Let Y be the set of feasibly scheduled jobs in both OPT and NE , and let N be the set of jobs feasibly scheduled in OPT but not in NE . Clearly $w(Y) \leq w(NE) < \frac{1}{\alpha+1}w(OPT) = \frac{1}{\alpha+1}(w(Y) + w(N))$. Hence $w(Y) < \frac{1}{\alpha}w(N)$, and therefore,

$$w(NE) < \frac{1}{\alpha}w(N).$$

But again, the jobs N are available to the players in the allocation of NE . By additivity of the objective across players, i.e., $w(N) = \sum_n w(N \cap OPT_n)$, we conclude that there exists at least one player n for which $w(NE_n) < \frac{1}{\alpha}w(N \cap OPT_n)$, and as $N \cap OPT_n$ can be feasibly scheduled on machines M_n , this contradicts the definition of an α -approximate Nash equilibrium. $\square$

Illustration. We here give an example for the instance described in the proof of Theorem 3.

Fig. 5. Sequential NE in order of increasing speed and optimal solution.

Lemma 2.

$$w(S_1) \geq \frac{\gamma^{m_1} - (\gamma - 1)^{m_1}}{\gamma^{m_1}} w(OPT).$$

Proof. We know by definition of γ

$$\frac{w(S_1)}{w(OPT)} \geq \frac{m_1}{\gamma}.$$

We prove by induction on m_i that, for any i ,

$$\frac{m_i}{\gamma} \geq \frac{\gamma^{m_i} - (\gamma - 1)^{m_i}}{\gamma^{m_i}}.$$

When $m_i = 1$, we get

$$\frac{1}{\gamma} \geq \frac{\gamma - (\gamma - 1)}{\gamma} = \frac{1}{\gamma},$$

which clearly holds. Assume the claim holds for $m_i - 1$. We get

$$\begin{aligned} \frac{m_i}{\gamma} &= \frac{m_i - 1}{\gamma} + \frac{1}{\gamma} \geq \frac{\gamma^{m_i-1} - (\gamma - 1)^{m_i-1}}{\gamma^{m_i-1}} + \frac{1}{\gamma} = \\ &= \frac{\gamma^{m_i} - \gamma(\gamma - 1)^{m_i-1}}{\gamma^{m_i}} + \frac{\gamma^{m_i-1}}{\gamma^{m_i}} = \frac{\gamma^{m_i} - (\gamma - 1)^{m_i}}{\gamma^{m_i}}. \end{aligned}$$

□

Lemma 3.

$$\frac{m_i}{\gamma} + \frac{\gamma - m_i}{\gamma} \cdot \frac{\gamma^{m'_{i-1}} - (\gamma - 1)^{m'_{i-1}}}{\gamma^{m'_{i-1}}} \geq \frac{\gamma^{m'_i} - (\gamma - 1)^{m'_i}}{\gamma^{m'_i}}.$$

Proof. We have

$$\frac{m_i}{\gamma} + \frac{\gamma - m_i}{\gamma} \cdot \frac{\gamma^{m'_{i-1}} - (\gamma - 1)^{m'_{i-1}}}{\gamma^{m'_{i-1}}} = \frac{m_i}{\gamma} \cdot \frac{(\gamma - 1)^{m'_{i-1}}}{\gamma^{m'_{i-1}}} + \frac{\gamma^{m'_{i-1}} - (\gamma - 1)^{m'_{i-1}}}{\gamma^{m'_{i-1}}}. \quad (5)$$

From the proof of Lemma 2 we get

$$\frac{m_i}{\gamma} \cdot \frac{(\gamma - 1)^{m'_{i-1}}}{\gamma^{m'_{i-1}}} \geq \frac{\gamma^{m_i} - (\gamma - 1)^{m_i}}{\gamma^{m_i}} \cdot \frac{(\gamma - 1)^{m'_{i-1}}}{\gamma^{m'_{i-1}}}.$$

Hence we get the following lower bound for (5)

$$\frac{\gamma^{m_i} - (\gamma - 1)^{m_i}}{\gamma^{m_i}} \cdot \frac{(\gamma - 1)^{m'_{i-1}}}{\gamma^{m'_{i-1}}} + \frac{\gamma^{m'_{i-1}} - (\gamma - 1)^{m'_{i-1}}}{\gamma^{m'_{i-1}}} = 1 - \frac{(\gamma - 1)^{m_i}}{\gamma^{m_i}} \cdot \frac{(\gamma - 1)^{m'_{i-1}}}{\gamma^{m'_{i-1}}}.$$

Observe that $m'_i = m'_{i-1} + m_i$, which yields

$$1 - \frac{(\gamma - 1)^{m_i}}{\gamma^{m_i}} \cdot \frac{(\gamma - 1)^{m'_{i-1}}}{\gamma^{m'_{i-1}}} = \frac{\gamma^{m'_i} - (\gamma - 1)^{m'_i}}{\gamma^{m'_i}}.$$

□