J Comb Optim (2017) 33:567–579
DOI 10.1007/s10878-015-9980-9

A coordination mechanism for a scheduling game with parallel-batching machines

Q. Q. Nong[1] · G. Q. Fan[1] · Q. Z. Fang[1]

Published online: 4 January 2016

Abstract In this paper we consider the scheduling problem with parallel-batching machines from a game theoretic perspective. There are m parallel-batching machines each of which can handle up to b jobs simultaneously as a batch. The processing time of a batch is the time required for processing the longest job in the batch, and all the jobs in a batch start and complete at the same time. There are n jobs. Each job is owned by a rational and selfish agent and its individual cost is the completion time of its job. The social cost is the largest completion time over all jobs, the makespan. We design a coordination mechanism for the scheduling game problem. We discuss the existence of pure Nash Equilibria and offer upper and lower bounds on the price of anarchy of the coordination mechanism. We show that the mechanism has a price of anarchy no more than $2-\frac{2}{3b}-\frac{1}{3\max\{m,b\}}$.

Keywords Game · Scheduling · Coordination mechanism · Nash Equilibrium · Price of anarchy

1 Introduction

With the development of the Internet, large-scale autonomous systems have become more and more common. The systems consist of many independent and selfish agents that compete for the usage of shared resources and act rationally on behalf of their own interest. Each agent has an individual cost and every system has some social cost. The selfish behavior of the agents results in a situation that can be characterized by

✉ Q. Q. Nong
qqnong@ouc.edu.cn

[1] School of Mathematical Science, Ocean University of China, Qingdao 266071, Shandong, People's Republic of China

some kind of equilibrium. Since the agents try to minimize their own cost, rather than the performance of the systems as a whole, such an equilibrium may lead to high cost compared to the global social optimum. A natural approach is designing protocols a priori in such a way that the selfish agents are induced to behavior that results in equilibria that nevertheless exhibit a good overall system performance. In this paper we are interested in a scheduling game on parallel-batching machines. Every agent owns a job and its individual cost is the completion time of its job. The social cost is the largest completion time over all jobs, the *makespan*.

Parallel-batching scheduling There are n jobs $\mathcal{J} = \{1, \ldots, n\}$, where each job $j \in \mathcal{J}$ has a processing time p_j, which is a rational number. The jobs are indexed in non-increasing order of their processing times, i.e., $p_1 \geq p_2 \geq \cdots \geq p_n$. There are m parallel-batching machines $\mathcal{M} = \{M_1, \ldots, M_m\}$ each of which is a system that can handle up to b ($b < n$) jobs simultaneously as a batch. The processing time of a batch is the time required for processing the longest job in the batch, and all the jobs in a batch start and complete at the same time. A schedule σ is m series of batches

$$\begin{array}{c} B_{11}, B_{12}, \ldots, B_{1b_1}; \\ \vdots \\ B_{i1}, B_{i2}, \ldots, B_{ib_i}; \\ \vdots \\ B_{m1}, B_{m2}, \ldots, B_{mb_m}, \end{array}$$

where: (i) $B_{i1}, B_{i2}, \ldots, B_{ib_i}$ are the batches processed on machine M_i; (ii) the batches form a partition of $\mathcal{J}$, i.e., $\bigcup_{i=1}^{m} \bigcup_{k=1}^{b_i} B_{ik} = \mathcal{J}$; (iii) the starting time of each batch is definite. The *makespan* $C_{\max}(\sigma)$ of a schedule σ is the maximal completion time over all jobs, and it is the *social cost* of the schedule. According to the scheduling notation introduced by Graham et al. (1979), the scheduling problem is expressed as $P|b < n|C_{\max}$.

Nash Equilibria and coordination mechanism What we described so far is a traditional scheduling problem, in which a central decision maker is equipped with all data of the problem and is asked to derive a feasible solution to optimize the social cost. In this paper we consider the scheduling problem from a game theoretic perspective as follows. Each of the n jobs is owned by an independent agent whose strategy is to choose a machine on which the job is to be processed (the term job and agent will be used interchangeably henceforth). The (pure) strategy set of a job is $\mathcal{M} = \{M_1, \ldots, M_m\}$. Each job is self-interested and does not care about the social optimum, its goal is to minimize its own completion time C_j. Each machine has a scheduling policy that determines how to assign all the jobs that select it to batches and specify the starting times of the resulting batches. A machine knows the IDs of those jobs that select it and their processing times. A machine does not know the processing times and strategies of the jobs that do not use it. The set of the policies of the machines is named as a *coordination mechanism*. All the data, including the policies of the machines and

the processing times of the jobs, are publicly known to the jobs. A job will select a machine according to the policies of the machines and the processing time information of other jobs. Let $x_j \in \mathcal{M}$ be the strategy of job j, and let $x = (x_1, x_2, \ldots, x_n)$ denote a (pure) strategy profile of the jobs. If the coordination mechanism of the machines is specified, there is a one-to-one correspondence between the set of strategy profiles and the set of schedules of the jobs. That is, given a coordination mechanism and a strategy profile, one can specify how is $\mathcal{J}$ partitioned into batches and the completion time of each job in $\mathcal{J}$. Henceforth, we will use the term schedule and strategy profile interchangeably if we have specified a coordination mechanism. A *pure Nash Equilibrium solution* is a strategy profile $x = (x_1, x_2, \ldots, x_n)$ such that for every job j we have

$$C_j(x) \leq C_j(x'_j, x_{-j}), \forall x'_j \in \mathcal{M} \setminus \{x_j\},$$

where $C_j(x)$ is the completion time of j under strategy profile $x = (x_1, x_2, \ldots, x_n)$. The quality of a coordination mechanism is measured by the *price of anarchy* (PoA) (Koutsoupias and Papadimitriou 2009). It is defined as the ratio between the social cost (makespan) of the worst Nash Equilibrium and the social cost of an optimal schedule. The price of anarchy of a coordination mechanism of a scheduling game is defined by

$$PoA = \max_{I \in \mathcal{G}} \max_{x \in Nash(I)} \frac{C_{\max}(x)}{C^*_{\max}(I)},$$

where $\mathcal{G}$ is the set of all instances of the scheduling game problem, $Nash(I)$ is the set of all Nash Equilibria for I, and $C^*_{\max}(I)$ is the social cost of the optimal schedule of I.

Related work Coordination mechanism design is introduced by Christodoulou et al. (2009). From then on many researchers are interested in the design of coordination mechanisms for scheduling game problems. They mainly study coordination mechanisms for four machine environments. To describe the machine environments, we denote the processing time of job j on machine M_i by p_{ij}. The four machine environments are: (i) *identical* machine environment in which $p_{ij} = p_{kj}$ for each job j and machines M_i and M_k; (ii) *uniform* or *related* machine environment in which every job j has a length l_j and every machine M_i has a speed s_i such that $p_{ij} = l_j/s_i$; (iii) *unrelated* machine environment in which the processing times p_{ij} are arbitrary positive numbers; (iv) *restricted identical* machine environment in which every job j comes with a length l_j and a set of machines $\mathcal{S}_j$ such that $p_{ij} = l_j$ for $M_i \in \mathcal{S}_j$ and $p_{ij} = \infty$ otherwise. They analyzed the existence of pure Nash Equilibria under SPT, LPT, MAKESPAN, RANDOM and EQUI for certain machine environment and estimated the price of anarchy of the coordination mechanisms. SPT and LPT are policies that schedule the jobs without preemption respectively in order of increasing or decreasing processing times for each machine. MAKESPAN is a policy that each machine schedules its jobs in parallel by time-multiplexing and each job shares a fraction of CPU of the machine that is proportional to its processing time. RANDOM policy schedules the jobs in a random order without preemption. EQUI is a policy that

Table 1 The price of anarchy for five different policies and scheduling problems

Problem	Scheduling Policy				
	SPT	*LPT*	*Makespan*	*Randomized*	*EQUI*
$P\|\|C_{\max}$	$2-\frac{1}{m}$ (Graham 1969; Immorlica et al. 2009)	$\frac{4}{3}-\frac{1}{3m}$ (Christodoulou et al. 2009; Graham 1969)	$2-\frac{2}{m+1}$ (Finn and Horowitz 1979; Schuurman and Vredeveld 2007)	$2-\frac{2}{m+1}$ (Finn and Horowitz 1979; Schuurman and Vredeveld 2007)	$2-\frac{1}{m}$ (Durr and Thang 2011)
$Q\|\|C_{\max}$	$\Theta(\log m)$ (Aspnes et al. 1997; Immorlica et al. 2009)	$1.52 \leq POA \leq 1.59$ (Dobson 1984; Friesen 1987; Immorlica et al. 2009)	$\Theta(\frac{\log m}{\log\log m})$ (Czumaj and Vocking 2007)	$\Theta(\frac{\log m}{\log\log m})$ (Czumaj and Vocking 2007)	$\Theta(\log m)$ (Durr and Thang 2011)
$B\|\|C_{\max}$	$\Theta(\log m)$ (Aspnes et al. 1997; Immorlica et al. 2009)	$\Theta(\log m)$ (Azar et al. 1995; Immorlica et al. 2009)	$\Theta(\frac{\log m}{\log\log m})$ Awerbuch et al. (2006), Gairing et al. (2004)	$\Theta(\frac{\log m}{\log\log m})$ (Azar et al. 1995; Friesen 1987)	$\Theta(\log m)$ (Durr and Thang 2011)
$R\|\|C_{\max}$	$\Theta(m)$ (Azar et al. 2008; Cho and Sahni 1980; Ibarra and Kim 1997)	Unbounded	Unbounded (Schuurman and Vredeveld 2007)	$\Theta(m)$ (Graham 1969)	$\Theta(m)$ (Durr and Thang 2011)

each machine schedules its jobs in parallel by time-multiplexing and each job shares the same fraction until it is completed. The results are summarized in Table 1.

A number of researchers have studied the traditional parallel-batching scheduling problems that are under centralized situation (Deng et al. 2003; Lee et al. 1992; Lee and Uzsoy 1999; Ng et al. 2003; Potts and Kovalyov 2000; Uzsoy 1994; Zhang et al. 2001). Scheduling problem $P|b < n|C_{\max}$ is $NP-$hard in the strong sense even for $b = 1$ (Lageweg et al. 1981). Researchers therefore turn to studying approximation algorithms for the problem. The quality of an approximation algorithm is often measured by its *worst-case ratio*: the smaller the ratio is, the better the algorithm will be. We say that an algorithm has a worst-case ratio ρ (or is a ρ-approximation algorithm) if for any input instance, it always returns in polynomial time of the input size a feasible solution with an objective value not greater than ρ times of the optimal value. Lee et al. (1992) considered the problem $P|b < n|C_{\max}$ and developed a $\frac{4}{3}-\frac{1}{3m}$-approximation algorithm, where m is the number of batching machines. When $b = 1$ the problem $P|b < n|C_{\max}$ is $P||C_{\max}$, which is a scheduling problem considering how to assign jobs to parallel machines and process them without preemption such that the makespan is minimized. A well known approximation algorithms for $P||C_{\max}$ is the *LPT-Greedy algorithm* (Graham 1969). The algorithm assigns at time 0 the m longest jobs on the m machines and after that, whenever a machine is free, put the longest job among those not yet processed on the machine. The algorithm has a worst-case ratio of $\frac{4}{3}-\frac{1}{3m}$.

Our contribution In this paper we design a coordination mechanism for the scheduling game problem $P|b < n|C_{\max}$. We discuss the existence of pure Nash Equilibria and show that the mechanism has a price of anarchy no more than $2 - 2/(3b) - 1/(3\max\{m, b\})$.

2 The FBLPT coordination mechanism

In this section we present a coordination mechanism, named FBLPT (Full Batch Longest Processing Time), for the scheduling game problem $P|b < n|C_{\max}$. Before explaining the coordination mechanism, we first describe the FBLPT rule, which is a rule to assign jobs into batches.

FBLPT Rule

(1) Sort the jobs in $\mathcal{J}$ in non-increasing order of their processing times (if $p_j = p'_j$ and $j < j'$, job $j \prec$ job j') and obtain a job list.

(2) If there are more than b jobs in the job list, place the first b jobs in a batch and iterate. Otherwise, place the remaining jobs in a batch.

Lee et al. (1992) prove that the FBLPT rule is a rule to assign jobs into batches such that the total processing time of the resulting batches is minimized. Based on this insight, we design a coordination mechanism as follows.

FBLPT Coordination Mechanism

Scheduling Policy of M_i: Group all the jobs that have selected M_i into batches by FBLPT rule. At time 0 start processing the resulting batches greedily in non-increasing order of their processing times.

2.1 The existence of pure Nash Equilibrium

Consider the LPT-Greedy algorithm for the scheduling problem $P|b < n|C_{\max}$. The idea of the algorithm is sorting the jobs in non-increasing order of their processing times and then assigning the jobs one by one into some batch and processing the resulting batches greedily. Define a subschedule to be in *state* j if jobs $1, 2, \ldots, j$ have been scheduled. Job $j + 1$ will be allocated in the last batch (if the batch is not full) or in a new batch (if the last batch has been full) of a machine and the machine will be the one that completes the processing of the job as early as possible. Denote the completion time on machine M_i in state j by $L_i(j)$. It is equal to the total processing time of the batches on machine M_i in state j. Let $N_i(j)$ be the number of jobs scheduled on machine M_i in state j. Let

$$T_i(j+1) = \begin{cases} L_i(j) + p_{j+1}, & \text{if } N_i(j) \pmod b = 0; \\ L_i(j), & \text{otherwise.} \end{cases}$$

The algorithm runs as below.

Algorithm LPT-Greedy
Initialization: Set $j := 1$. For $i = 1, 2, \ldots, m$ set $L_i(0) := 0$ and $N_i(0) := 0$.
Begin
While $j \leq n$ do
Suppose i^* is the machine such that $T_{i^*}(j) = \min_{1 \leq i \leq m} \{T_i(j)\}$. Then job j is assigned in the $\lceil \frac{N_{i^*}(j-1)+1}{b} \rceil$-th batch of M_{i^*}.
Set $L_{i^*}(j) := T_{i^*}(j)$, $N_{i^*}(j) := N_{i^*}(j-1) + 1$; for each $i \neq i^*$, set $L_i(j) := L_i(j-1)$, $N_i(j) := N_i(j-1)$; set $j := j + 1$.
End

Consider the schedule produced by the LPT-Greedy algorithm. It is not difficult to see that the subschedule of the jobs processed on machine M_i $(1 \leq i \leq m)$ is consistent with the one produced by the Scheduling Policy of M_i. We prove the following theorem.

Theorem 1 *The schedule generated by the LPT-Greedy algorithm is a pure Nash Equilibrium of the scheduling game problem $P|b < n|C_{\max}$ under the FBLPT coordination mechanism.*

Proof Let σ be the schedule produced by the Job-Greedy algorithm. We show that σ is a Nash Equilibrium by induction. Job 1 starts at time 0. It has no incentive to switch to other machines. Suppose that all jobs $1, 2, \ldots, k$ have no incentive to switch to other machines. We consider job $k + 1$. Since a machine assigns all the jobs that have selected it into batches by FBLPT rule and processes the batches in the LPT order, job $k + 1$ cannot start earlier than jobs $1, 2, \ldots, k$. For the same reason, the choices of job $k + 2, \ldots, n$ have no influence on job j. The machine for job $k + 1$ in state $k + 1$ is the best selection and it has no incentive to switch to other machines. Thus σ is a Nash Equilibrium. □

2.2 An upper bound on the price of anarchy

To analyze the price of anarchy of the FBLPT coordination mechanism, we need the following lemma.

Lemma 2 [See Lee et al. (1992)] *There is an optimal schedule for the problem $P|b < n|C_{\max}$ such that the jobs in $\mathcal{J}$ are partitioned into batches by the FBLPT rule.*

Theorem 3 *The price of anarchy of the FBLPT coordination mechanism is at most*

$$\frac{C_{\max}}{C^*_{\max}} \leq 2 - \frac{2}{3b} - \frac{1}{3\max\{m, b\}}.$$

Proof We prove the result by contradiction. Assume that there are counterexamples each of which has a Nash Equilibrium with a makespan strictly larger than $2 - \frac{2}{3b} - \frac{1}{3\max\{m,b\}}$ times of the optimum. Consider the one with the smallest number of jobs. Denote the counterexample by I_1 and assume that it has n_1 jobs. Let σ_1 be the worst

Nash Equilibrium of I_1, whose makespan is strictly larger than $2 - \frac{2}{3b} - \frac{1}{3\max\{m,b\}}$ times of the optimum. Let us have a look at the structure of σ_1. Assume that there are b_i $(1 \leq i \leq m)$ batches processed on M_i. Sort them in non-increasing order of their starting times and denote them by $B_{i1}, B_{i2}, \ldots, B_{ib_i}$, respectively. □

Claim 1 There is only one batch completing at $C_{\max}(\sigma_1)$ and it consists of only one job.

If it is not the case, consider the batches that complete at $C_{\max}(\sigma_1)$. Let B_{lb_l} be the batch with the earliest starting time and suppose that it starts at t^*. Except the longest job in batch B_{lb_l}, discard all jobs that start at or after t^* in σ_1. Consider the resulting schedule. We obtain a new instance I_2 with a smaller number of jobs. All the jobs that start before t^* is larger than the job remaining in B_{lb_l}, and all the batches that start before t^* is full (otherwise the job in B_{lb_l} can reduce its completion time by unilaterally changing its strategy in σ_1). Therefore, the resulting schedule is also a Nash Equilibrium for instance I_2. The makespan of it is equal to $C_{\max}(\sigma_1)$ and the optimal makespan of I_2 is not greater than that of I_1. Thus, I_2 is a counterexample with fewer jobs than I_1, a contradiction.

The claim together with the fact that all the batches that start before t^* is full means that $n_1 \pmod b = 1$ and thus $B_{lb_l} = \{n_1\}$.

Denote the processing time of batch B_{ik} by P_{ik}. We construct a new instance $\mathcal{F}$ by adding $b-1$ jobs, each with a processing time of p_{n_1}, to the smallest counterexample I_1. Assume that the set of jobs in $\mathcal{F}$ is $\{1, 2, \ldots, n\}$. Clearly, $n \pmod b = 0$. Consider a schedule of $\mathcal{F}$ by adding the $b-1$ jobs to the batch B_{lb_l} in σ_1. Denote it by σ. Since the processing time of the added $b-1$ jobs is p_{n_1}, which is the smallest processing time. No job in σ can reduce its completion time by unilaterally changing its strategy. Therefore, σ is a Nash Equilibrium for $\mathcal{F}$. It is not difficult to see that σ is equal to σ_1 in the sense that in σ the number of the batches processed on each M_i $(1 \leq i \leq m)$ is the same as in σ_1 and the processing times of the batches are equal correspondingly. Thus, the makespan of σ is equal to the makespan of σ_1. Without causing confusion, denote the makespan of σ by $C_{\max}$ and denote the batches processed on M_i in σ by $B_{i1}, B_{i2}, \ldots, B_{ib_i}$, respectively.

On the other hand, consider the optimal schedule of $\mathcal{F}$. Let σ^* be the optimal schedule of I_1 that is described in Lemma 2. Sort the batches of σ^* in non-increasing order of their processing times and obtain a list $L^* = (B_1^*, B_2^*, \ldots, B_d^*)$, where $d = \lceil \frac{n_1}{b} \rceil$. Partition the jobs in $\mathcal{F}$ into batches by FBLPT rule. Sort them in non-increasing order of their processing times and obtain a list $L = (B_1, B_2, \ldots, B_e)$, where $e = \lceil \frac{n}{b} \rceil$. Noting that $n_1 \pmod b = 1$ and $n = n_1 + b - 1$, we have $\lceil \frac{n_1}{b} \rceil = \lceil \frac{n}{b} \rceil$, i.e., $d = e$. For each $1 \leq k \leq d$ the processing time of B_k^* is equal to that of B_k. By Lemma 2, we can deduce that the optimum value of $\mathcal{F}$, denoted by $C_{\max}^*$, is equal to the optimum value of I_1. Therefore, the following claim holds.

Claim 2 $\mathcal{F}$ is a counterexample and σ is a Nash Equilibrium of $\mathcal{F}$ with a makespan strictly larger than $2 - \frac{2}{3b} - \frac{1}{3\max\{m,b\}}$ times of the optimum.

In the following we analyze σ. We first give an upper bound on $C_{\max}$. In σ the completion time of M_l is $C_{\max}$ and the completion time of each other machine M_i $(i \neq l)$ is at least $C_{\max} - P_{lb_l}$. We have

$$\sum_{i=1}^{m}\sum_{k=1}^{b_i} P_{ik} \geq (m-1)(C_{\max} - P_{lb_l}) + C_{\max}$$
$$= mC_{\max} - (m-1)P_{lb_l}. \quad (1)$$

We then provide lower bounds on $C^*_{\max}$. Clearly,

$$C^*_{\max} \geq \frac{\sum_{j=1}^{n} p_j}{mb}. \quad (2)$$

Let us give a lower bound on $\sum_{j=1}^{n} p_j$. From each batch in σ remove the longest job. Then the total processing time of the remaining jobs is

$$\sum_{j=1}^{n} p_j - \sum_{i=1}^{m}\sum_{k=1}^{b_i} P_{ik}.$$

For each $1 \leq i \leq m$, consider the resulting batches on machine M_i. Note that each batch in σ is full. There are $b-1$ jobs remaining in B_{ik} ($1 \leq k \leq b_i$) after removing the longest job from it. Further, since $B_{i1}, B_{i2}, \ldots, B_{ib_i}$ satisfy the $FBLPT$ rule, one can see that for each $k \in \{1, 2, \ldots, b_i - 1\}$, the processing time of each job in batch B_{ik} is at least $P_{i,k+1}$ and the processing time of each job in batch B_{ib_i} is at least P_{lb_l}. Thus the total processing time of the remaining jobs on machine M_i is at least

$$(b-1)\left(\sum_{k=2}^{b_i} P_{ik} + P_{lb_l}\right).$$

Thus

$$\sum_{j=1}^{n} p_j - \sum_{i=1}^{m}\sum_{k=1}^{b_i} P_{ik} \geq (b-1)\left(\sum_{i=1}^{m}\sum_{k=2}^{b_i} P_{ik} + mP_{lb_l}\right)$$
$$= (b-1)\left(\sum_{i=1}^{m}\sum_{k=1}^{b_i} P_{ik} - \sum_{i=1}^{m} P_{i1} + mP_{lb_l}\right),$$

implying that,

$$\sum_{j=1}^{n} p_j \geq b\sum_{i=1}^{m}\sum_{k=1}^{b_i} P_{ik} - (b-1)\left(\sum_{i=1}^{m} P_{i1} - mP_{lb_l}\right). \quad (3)$$

Since $C^*_{\max} \geq P_{i1}$ for $i = 1, 2, \ldots, m$, we have

$$mC^*_{\max} \geq \sum_{i=1}^{m} P_{i1}. \tag{4}$$

By inequalities (2), (3), (1) and (4), we have the following result:

$$\begin{aligned} mbC^*_{\max} &\geq \sum_{j=1}^{n} p_j \\ &\geq b \sum_{i=1}^{m} \sum_{k=1}^{b_i} P_{ik} - (b-1)\left(\sum_{i=1}^{m} P_{i1} - mP_{lb_l} \right) \\ &\geq b\left(mC_{\max} - (m-1)P_{lb_l}\right) - (b-1)m\left(C^*_{\max} - P_{lb_l}\right), \end{aligned}$$

which implies that

$$mbC_{\max} \leq m(2b-1)C^*_{\max} + (m-b)P_{lb_l},$$

and thus

$$\frac{C_{\max}}{C^*_{\max}} \leq 2 - \frac{1}{b} + (\frac{1}{b} - \frac{1}{m})\frac{P_{lb_l}}{C^*_{\max}}. \tag{5}$$

In the sequel we show that the ratio $\frac{C_{\max}}{C^*_{\max}}$ is not greater than $2 - \frac{2}{3b} - \frac{1}{3\max\{m,b\}}$. We distinguish three cases to discuss.

Case 1 $C^*_{\max} \geq 3P_{lb_l}$ and $b \leq m$.

From inequality (5), we have

$$\begin{aligned} \frac{C_{\max}}{C^*_{\max}} &\leq 2 - \frac{1}{b} + \frac{1}{3}(\frac{1}{b} - \frac{1}{m}) \\ &= 2 - \frac{2}{3b} - \frac{1}{3m} \\ &= 2 - \frac{2}{3b} - \frac{1}{3\max\{m,b\}}. \end{aligned}$$

Case 2 $C^*_{\max} \geq 3P_{lb_l}$ and $b > m$.

Then $\frac{1}{b} - \frac{1}{m} < 0$ and we have

$$\begin{aligned} \frac{C_{\max}}{C^*_{\max}} &\leq 2 - \frac{1}{b} + (\frac{1}{b} - \frac{1}{m})\frac{P_{lb_l}}{C^*_{\max}} \\ &< 2 - \frac{1}{b} \end{aligned}$$

$$= 2 - \frac{2}{3b} - \frac{1}{3b}$$
$$= 2 - \frac{2}{3b} - \frac{1}{3\max\{m, b\}}.$$

Case 3 $C^*_{\max} < 3P_{lb_l}$.

Note that P_{lb_l} is the processing time of the shortest job in $\mathcal{J}$. The inequality $C^*_{\max} < 3P_{lb_l}$ implies that in the optimal schedule of $\mathcal{F}$ there are at most two batches on each machine and thus $n \leq 2mb$.

If $n \leq mb$, recalling that each batch is full in σ, we can deduce that there is at most one batch on each machine. Hence

$$C_{\max} \leq \max_{1 \leq i \leq m} \{P_{i1}\} \leq C^*_{\max},$$

which means that $\frac{C_{\max}}{C^*_{\max}} \leq 1$.

If $n > mb$, there is at least one machine which processes two batches in the optimal schedule. Thus $C^*_{\max} \geq 2P_{lb_l}$. We claim that

$$C^*_{\max} \geq C_{\max} - P_{lb_l}.$$

Since $n \leq 2mb$ and each batch in σ is full, there are at most $2m$ batches in σ. If $b_l = 2$ then $C_{\max} = P_{l1} + P_{lb_l}$ and thus

$$C^*_{\max} \geq P_{l1} \geq C_{\max} - P_{lb_l}.$$

If $b_l \geq 3$, there exists one machine M_a which processes only one batch and the batch is completed at P_{a1}. Recalling that in σ the completion time of a machine is at least $C_{\max} - P_{lb_l}$, we have

$$P_{a1} \geq C_{\max} - P_{lb_l}.$$

Together with the fact that $C^*_{\max} \geq P_{a1}$, we have

$$C^*_{\max} \geq P_{a1} \geq C_{\max} - P_{lb_l}.$$

In both cases the claim follows. Therefore,

$$\frac{C_{\max}}{C^*_{\max}} \leq 1 + \frac{P_{lb_l}}{C^*_{\max}} \leq \frac{3}{2}.$$

The above discussion implies that

$$\frac{C_{\max}}{C^*_{\max}} \leq 2 - \frac{2}{3b} - \frac{1}{3\max\{m, b\}}.$$

Thus σ is not a Nash Equilibrium of $\mathcal{F}$ with a makespan strictly larger than $2-\frac{2}{3b}-\frac{1}{3\max\{m,b\}}$ times of the optimum, leading to a contradiction. This completes the proof of the theorem. □

Interestingly, if $b=1$ we have

$$2-\frac{2}{3b}-\frac{1}{3\max\{m,b\}}=\frac{4}{3}-\frac{1}{3m},$$

which is exactly the worst-case ratio of the LPT-Greedy algorithm.

2.3 Lower bounds on the price of anarchy

To give a lower bound on the price of anarchy of the mechanism, we only need to offer an instance and estimate the ratio between the makespan of a Nash Equilibrium and the optimum. We distinguish two cases to discuss.

Case 1 $m \le b$.

Consider the following instance. There are $b(m-1)X+b$ jobs, where $X=mk+1$ for some positive integer k. b jobs are long and each of them has a processing time of X. $b(m-1)X$ jobs are short and each of them has a processing time of 1.

We can see that the optimal schedule of the instance is as follows: a machine processes a batch consisting of all the long jobs, and each of the other machines processes X batches each consisting of b short jobs. Thus the optimal makespan is $C^*_{\max}=X=mk+1$. Consider a schedule π as follows. For each $1\le i\le m-1$, the first batch processed on M_i is a batch that contains one long job and $b-1$ short jobs. The first batch on M_m consists of $b-m+1$ long jobs and $m-1$ short jobs. On each machine, following the long batch there are $(m-1)k$ batches each of which consists of b short jobs. It is easy to see that π is a Nash Equilibrium and its makespan is

$$C_{\max}=X+(m-1)k=(2m-1)k+1.$$

Thus

$$\frac{C_{\max}}{C^*_{\max}}=\frac{(2m-1)k+1}{mk+1}\to 2-\frac{1}{m},\ \text{as } k\to\infty.$$

Specially, if $m=b$ we have

$$\frac{C_{\max}}{C^*_{\max}}=\frac{(2b-1)k+1}{bk+1}\to 2-\frac{1}{b},\ \text{as } k\to\infty.$$

Case 2 $m > b$.

Define r to be a positive integer such that

$$r=\begin{cases} b, & \text{if } m \pmod b = 0;\\ m \pmod b, & \text{otherwise.}\end{cases}$$

Set $z = \lceil m/b \rceil$, and we have $z = \frac{m+b-r}{b}$. Consider the following instance. There are m long jobs each with a processing time of X, as before, $X = mk + 1$ for some positive integer k. Besides the long jobs, there are

$$(m - z)bX + b - r$$

short jobs and each of them has a processing time of 1.

We can see that the optimal schedule of the instance is as follows: for each $1 \leq i \leq z - 1$, machine M_i processes a batch consisting of b long jobs; M_z processes a batch containing r long jobs and $b - r$ short jobs; for each $z + 1 \leq i \leq m$, machine M_i processes X batches each consisting of b short jobs. Thus the optimal makespan is $C^*_{\max} = X = mk + 1$. Consider a schedule π' as follows. For each $1 \leq i \leq m$, machine M_i processes a batch containing one long job and then $b - 1$ shorts jobs and $(m - z)k$ batches each with b short jobs. Clearly, π' is a Nash Equilibrium and its makespan is

$$C_{\max} = X + (m - z)k = (2m - z)k + 1.$$

Thus

$$\frac{C_{\max}}{C^*_{\max}} = \frac{(2m - z)k + 1}{mk + 1} \to 2 - \frac{z}{m} = 2 - \frac{\frac{m+b-r}{b}}{m} = 2 - \frac{1}{b} - \frac{1}{m} + \frac{r}{mb}, \text{ as } k \to \infty.$$

In summary, we have proved the following theorem.

Theorem 4 *If $m \leq b$ there is an instance with a Nash Equilibrium such that $\frac{C_{\max}}{C^*_{\max}} \geq 2 - \frac{1}{m}$; If $m > b$ there is an instance with a Nash Equilibrium such that $\frac{C_{\max}}{C^*_{\max}} \geq 2 - \frac{1}{b} - \frac{1}{m} + \frac{r}{mb}$.* □

Corollary 5 *When $m = b$ the analysis of the price of anarchy of the FBLPT coordination mechanism in* Theorem 3 *is tight.*

Proof Note that $2 - \frac{2}{3b} - \frac{1}{3\max\{m,b\}} = 2 - \frac{1}{b} = 2 - \frac{1}{m}$ when $m = b$. The result follows. □

Acknowledgements This research was supported in part by the *National Natural Science Foundation of China* under Grant Numbers 11201439 and 11271341. This work was also supported in part by the *Shandong Provincial Natural Science Foundation, China*, under Grant Number ZR2012AQ12 and by the *Doctoral Fund of Ministry of Education of China* (20120132120001).

References

Aspnes J, Azar Y, Fiat A, Plotkin SA, Waarts O (1997) On-line routing of virtual circuits with applications to load balancing and machine scheduling. J ACM 44(3):486–504

Awerbuch B, Azar Y, Richter Y, Tsur D (2006) Tradeoffs in worst-case equilibria. Theor Comput Sci 361(2–3):200–209

Azar Y, Jain K, Mirrokni VS (2008) (Almost) optimal coordination mechanisms for unrelated machine scheduling. In: Proceedings of the 19th annual ACM-SIAM symposium on discrete algorithms, SODA, pp 323-332

Azar Y, Naor J, Rom R (1995) The competitiveness of on-line assignments. J Algorithms 18(2):221–237

Cho Y, Sahni S (1980) Bounds for list schedules on uniform processors. SIAM J Comput 9:91–103

Christodoulou G, Koutsoupias E, Nanavati A (2009) Coordination mechanisms. Theor Comput Sci 410(36):3327–3336

Czumaj A, Vocking B (2007) Tight bounds for worst-case equilibria. ACM Trans Algorithms (TALG) 3(1):1–17

Deng X, Poon CK, Zhang Y (2003) Approximation algorithms in batching processing. J Comb Optim 7:247–257

Dobson G (1984) Scheduling independent tasks on uniform processors. SIAM J Comput 13:716–721

Durr C, Thang NK (2011) Non-clairvoyant scheduling games. Theory Comput Syst 49(1):3–23

Finn G, Horowitz E (1979) A linear time approximation algorithm for multiprocessor scheduling. BIT 19:312–320

Friesen DK (1987) Tighter bounds for LPT scheduling on uniform processors. SIAM J Comput 16:554–560

Gairing M, Lucking T, Mavronicolas M, Monien B (2004) Computing nash equilibria for scheduling on restricted parallel links. In: STOC, pp 613-622

Graham RL (1969) Bounds on multiprocessing timing anomalies. SIAM J Appl Math 45:416–429

Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing andscheduling: a survey. Ann Discrete Math 5(2):287–326

Ibarra OH, Kim CE (1997) Huristic algorithms for scheduling independent tasks on nonidentical processors. J ACM 24:280–289

Immorlica N, Li L, Mirrokni VS, Schulz A (2009) Coordination mechanisms for selfish scheduling. Theor Comput Sci 410:1589–1598

Koutsoupias E, Papadimitriou C (2009) Worst-case equilibria. Comput Sci Rev 3(2):65–69

Lageweg BJ, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1981) Computer aided complexity classification of deterministic scheduling problems. Research report BW138/81, Mathematisch Centrum, Amsterdam

Lee CY, Uzsoy R, Martin Vega LA (1992) Efficient algorithms for scheduling semiconductor burn-in operations. Oper Res 40:764–775

Lee CY, Uzsoy R (1999) Minimizing makespan on a single batch processing machine with dynamic job arrivals. Int J Prod Res 37:219–236

Ng CT, Cheng TCE, Yuan JJ (2003) The single machine batching problem with family setup times to minimize maximum lateness is strongly NP-hard. J Sched 6:483–490

Potts CN, Kovalyov MY (2000) Scheduling with batching: a review. Eur J Oper Res 120:228–249

Schuurman P, Vredeveld T (2007) Performance guarantees of local search for multiprocessor scheduling. INFORMS J Comput 19:52–63

Uzsoy R (1994) A single batch processing machine with non-identical job sizes. Int J Prod Res 32:1615–1635

Zhang G, Cai X, Wong CK (2001) On-line algorithms for minimizing makespan on batch processing machines. Nav Res Logist 48:241–258