2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology

The Price of Stability in Selfish Scheduling Games

Lucas AGUSSURJA
The Logistics Institute Asia Pacific
National University of Singapore
tlila@nus.edu.sg

Hoong Chuin LAU
School of Information Systems
Singapore Management University
hclau@smu.edu.sg

Abstract

Game theory has gained popularity as an approach to analysing and understanding distributed systems with self-interested agents. Central to game theory is the concept of Nash equilibrium as a stable state (solution) of the system, which comes with a price − the loss in efficiency. The quantification of the efficiency loss is one of the main research concerns. In this paper, we study the quality and computational characteristic of the best Nash equilibrium in two selfish scheduling models: the congestion model and the sequencing model. In particular, we present the following results: (1) In the congestion model: first, the best Nash equilibrium is socially optimum and consequently, computing the best Nash is NP-hard. And second, any ϵ-approximation algorithm for finding the optimum can be transformed into an ϵ-approximation algorithm for the best Nash. (2) In sequencing model for identical machines, we show that the best Nash is no better than the worst Nash and it is easy to compute. For related machines, we show that there is a gap between the worst and the best Nash equilibrium. We left the bounding of this gap for future work.

1 Introduction

Increasingly, business decision-making has evolved from consideration of optimal performance within an organization, to the ability to coordinate/contend with external agencies while maintaining a self-interested optimal position within bounded rationality. We observe scenarios of such emerging behavior in supply chain systems for example. As firms start exploring innovative collaboration strategies in effort to improve their supply chain efficiency, getting multiple firms to agree on joint decisions has been identified as one of the major research problems.

Game theory has become a key area in AI research. It has gained popularity as an approach to analysing and understanding distributed systems with self-interested agents. Central to game theory is the concept of Nash equilibrium as a stable state (solution) of the system, which comes with a price − a possible loss in efficiency. The problem of finding Nash equilibrium is the most fundamental computational problem whose complexity is wide open?[15].

In this paper, we consider decentralized scheduling systems with independent, rational, and self-interested job agents, where the overall system behavior and performance is a result of the interactions and actions of these agents. One scenario is autonomous job agents competing for resources, where resources can be bandwith (e.g. in a network) or processing power (e.g. in a Grid computing environment). The role of the central planner is to design and propose a mechanism/protocol mediating the interactions among the agents. An agent can choose to follow or to defect from the proposed protocol. Hence, it is an interesting problem (especially from the planner's standpoint) to analyze what system performance one can expect when selfish agents interact according to a proposed protocol, and how a stable solution that is mutually agreeable among agents may be constructed. Put in game-theoretic terms, the game designer is interested to know how to solve the scheduling game and what the efficiency of the game would be, and such information allows him to assess/estimate the performance of the proposed mechanism.

This paper focuses on models with complete information where all information are assumed to be available to all the agents and the central authority. This allows both the central authority and the agents to compute a *Nash equilibrium*. A Nash equilibrium is a state of the system where no agent has the incentive to singly defect away from it. Among all Nash equilibria, we are interested to find the $best$ one, where best means one that maximizes the social welfare. Our study of the quality of $best$ Nash equilibrium (termed the *price of stability* or POS) [2], which is measured by the ratio of the best Nash equilibrium to the optimum, stands in contrast with the broader line of work on the *price of anarchy* which measures the ratio of the Nash equilibrium to the optimum in the worst-case.

The price of stability is an important notion in cases where players may be guided to play at the best Nash equi-

0-7695-3027-3/07 $25.00 © 2007 IEEE
DOI 10.1109/IAT.2007.37

librium. This happens for instance in most networking applications where agents are neither centrally controlled nor completely unregulated; rather, they interact with an underlying protocol that essentially proposes a collective solution to all participants, who can each either accept or defect from it. As a result, it is in the interest of the protocol designer to seek the best Nash; this can naturally be viewed as the optimum subject to the constraint that the solution be stable, with no agent having an incentive to unilaterally defect once it is offered. Indeed, one can view the activity of the protocol designer seeking the best Nash as being aligned with the general goal of mechanism design to produce a game that yields good outcomes when players act in their own self-interest.

The objective of our work is to deepen understanding of game theoretic scheduling models. While much of the literature (see below) has focussed on analyzing its worst case behavior, in this paper we are concerned with the computational complexity of computing the best equilibrium, as well as the quality guarantee of such a solution. We consider only $pure$ Nash equilibria and seek to quantify the gap between the best and the worst Nash equilibrium. We obtain the following results on two different models:

1. In the congestion model: the price of stability is 1 (i.e. no loss of efficiency), and thus the gap is exactly the price of anarchy.

2. In the sequencing model: (a) For identical machines, the price of stability is equal to the price of anarchy. Hence there is no gap between the worst and the best Nash equilibrium. (b) For related machines, we identify a gap between the worst and the best Nash equilibrium although the analytical bound on the gap is left as future work.

2 Notation and Definitions

We consider scheduling games that consist of: (1) a set of machines indexed by $M = \{1, 2, ..., m\}$, and (2) a set of jobs indexed by $J = \{1, 2, ..., n\}$, where each job $j \in J$ has a length (processing time) of l_j. Each job is viewed as an agent whose decision is to choose the machine on which the job is to be processed (the term job and agent will be used interchangeably henceforth). Let $x_j \in M$ be the strategy (decision) of job j, and let $x = (x_1, x_2, ..., x_n)$ denotes a strategy profile (schedule) of the jobs. We assume that jobs are processed non-preemptively by the machines. We consider 2 types of machines as follows:

- identical machines: all machines have the same speed, and the processing time of a job j on machine i is just l_j.
- related machines: the machines can have different speeds. The processing time of a job j on machine i is l_j/s_i, where s_i is the speed of the machine.

The social objective of the central authority is to derive a schedule that minimizes the overall makespan, for example in related machines, it is to minimize the cost function:

$$F(x) = \max_{i \in M} \frac{1}{s_i} \sum_{j:x_j=i} l_j.$$

Given a schedule x, a *critical machine* in x is a machine with the maximum total processing time. Note that there can be more than one critical machine in a given schedule.

Each agent is self-interested and wants to minimize its own completion time (the time when the job is completed and released by the machine). We consider the following two popular models in the literature with different agent utility functions:

1. **Congestion model**. This model assumes that a job is released by a machine after all the jobs in that machine have been processed. In this case, the utility function of a job j in related machines is:

$$u_j(i, x_{-j}) = -\frac{1}{s_i} \sum_{k:x_k=i} l_k,$$

which is also the negative of machine i's total processing time. Note that all the agents in the same machine have the same utility.

2. **Sequencing model**. This model assumes that a job is released immediately after it is processed. For this model, a sequencing policy is needed for each of the machine. Two examples of such policy are: SPT (shortest processing time first) and LPT (longest processing time first). In this case, the utility function of a job j in related machines is:

$$u_j(i, x_{-j}) = -\frac{1}{s_i} \sum_{k:x_k=i \wedge k \preceq_i j} l_k,$$

where $\preceq_i$ defines an ordering on the set J and its definition depends on the sequencing policy used by machine i. In this work, we assume that in a sequencing model, all the machines adopt the same sequencing policy, and $\preceq$ is used to denote the ordering.

A Nash equilibrium (or simply Nash) solution is a strategy profile $x' = (x'_1, x'_2, ..., x'_n)$ such that for every job j we have:

$$u_j(x'_j, x'_{-j}) \geq u_j(x_j, x'_{-j}) \qquad \forall x_j \in M.$$

Given a Nash solution, no agent has the incentive to defect from it assuming all the other agents follow the solution. The best Nash solution in this case is one that minimizes the objective function $F(x)$. One can thus view the problem of finding the best Nash solution as an optimization problem, i.e. find x that minimizes $F(x)$ with a constraint that x must be a Nash equilibrium. Given a game, let $\mathcal{N}$ be the set of all the Nash solutions and x^* be the optimum solution, the price of stability (POS) of the game is given by:

$$\min_{x' \in \mathcal{N}} \frac{F(x')}{F(x^*)}$$

while the price of anarchy (POA) is one that maximizes the expression.

A *best response* $R_j(x_{-j})$ is the set of strategies which produces the most utility for the job j, given the other job strategies x_{-j}, i.e. $x'_j \in R_j(x_{-j})$ iff $u_j(x'_j, x_{-j}) \geq u_j(x_j, x_{-j})$ for all $x_j \in M$. We can redefine a Nash solution using best response as follows: A strategy profile x' is a Nash solution iff $\forall j \in J, x'_j \in R_j(x'_{-j})$. Given a strategy profile (x_j, x_{-j}), a *selfish move* by a job j changes the profile to (x'_j, x_{-j}) such that $u_j(x'_j, x_{-j}) > u_j(x_j, x_{-j})$ and $x'_j \in R_j(x_{-j})$. Note that the definition requires a selfish move to be strictly increasing the job's utility.

3 Related Work

In this section, we present known results on the price of anarchy for different types of scheduling models. For a more comprehensive survey, refer to [13]. The price of anarchy for congestion models is:

- $2 - 1/m$ for m identical machines [1].
- $\Theta\left(\frac{\log m}{\log \log m}\right)$ for m related (restricted) machines [7, 11].

The price of anarchy for sequencing models when SPT policy is used is:

- $2 - 2/(m+1)$ for m identical machines [18].
- $\Theta(\log m)$ for m related (restricted) machines [13].
- at most m for m unrelated machines [13].

And the price of anarchy when LPT policy is used is:

- $4/3 - 1/(3m)$ for m identical machines [6].
- at most $2 - 2/m$ for m related machines [13].
- $\Theta(\log m)$ for m restricted machines [3, 13].

The price of stability in network design game with fair cost allocation is studied by [2] and is at most $1 + \frac{1}{2} + ... + \frac{1}{n}$, with n number of players.

On complexity of finding a Nash equilibrium the following results are known. Computing mixed Nash equilibrium in 2, 3 and 4-player general normal form game is is PPAD-complete [5, 9, 8]. For computing pure Nash equilibrium, the results are known for a class of games called ordinal (generalized) potential game which is in PLS. And computing a pure Nash equilibrium in potential games which is a subset of ordinal potential games is PLS-Complete [10]. Recent algorithmic attempts to find a Nash equilibrium from AI community includes using mixed-integer programming [17], search methods [16] and continuation method [4].

4 Congestion Model

In this section, we present the result on congestion model first specifically to scheduling games and then its generalization to potential games.

4.1 Congestion Model for Scheduling Games

We present the result on congestion model in scheduling games, starting with $m = 2$ machines. For simplicity, the proof for the following result assumes identical machines, but it holds for related machines as well.

Proposition 1. *The optimum solution for congestion model with 2 machines is a Nash solution.*

Proof. Let L_1 and L_2 be the total processing time of the first and second machine respectively given the optimum solution x^*. If $L_1 = L_2$ then x^* is a Nash solution because no job can improve its utility by changing its strategy. Now, w.l.o.g. assume that $L_1 > L_2$ and let $\delta = L_1 - L_2$. If x^* is not a Nash solution, then there exist a job j that can change its strategy and obtained a better utility. This can happen iff j is in machine 1 and $l_j < \delta$. If job j is moved to machine 2, then we obtained a new schedule x' with cost $F(x') = \max(L_1 - l_j, L_2 + l_j)$. Since $L_2 + l_j < L_2 + \delta = L_1$, we have $F(x') < L_1 = F(x^*)$, thus contradicting the optimality of x^*. Therefore, x^* is a Nash solution. □

This also implies that the best Nash solution is an optimum solution, and there is no loss of efficiency if the best Nash solution can be achieved, i.e. the POS is 1. For more than 2 machines, the optimum solution is not always a Nash solution, but the best Nash solution is still an optimum solution. The following sequence of results are derived for $m > 2$ machines.

Lemma 2. *In congestion model, the cost of the new schedule x' resulting from a selfish move on an initial schedule x is at most the cost of the initial schedule, i.e. $F(x') \leq F(x)$.*

Proof. From proposition 1, we know that if a selfish move is made by a job from any machine i, with total processing time L_i, to any machine k with total processing time L_k, the result of the move is a new schedule with total processing time L'_i and L'_k for machine i and k respectively such that $\max(L'_i, L'_k) < \max(L_i, L_k)$. Since the cost of the schedule x is $F(x) = \max(L_1, ..., L_m)$, if i is the only critical machine in x then we have $F(x') = \max(L_1, ..., L'_i, ..., L'_k, ..., L_m) < \max(L_1, ..., L_m) = F(x)$, otherwise $F(x') = F(x)$. □

Note that in congestion model, although a selfish move affects the utility of other agents, it does not reduce the overall quality of the schedule; And for 2 machines, a selfish move even strictly reduces the cost of the schedule. The next result shows that any arbitrary schedule can be turned into a Nash solution by repetitive application of selfish moves. Again for simplicity, we assume identical machines in the argument of the proof.

Lemma 3. *In congestion model with m machines, the number of selfish moves needed to change an arbitrary schedule to a Nash solution is at most $O(mn)$, where n is the number of jobs.*

Proof. We first give a looser bound by allowing the sequence of best responses to be made by arbitrary jobs, and strengthen it later by giving a procedure for choosing the next job for selfish move. Given an arbitrary schedule x, let L_i be the total processing time of machine i, i.e. $L_i = \sum_{j:x_j=i} l_j$ and let $\delta = max_i L_i - min_i L_i$ be the difference of the longest processing time to the shortest processing time. Observe that every best response by a job j always moves j to the machine with the shortest processing time, and there are at most m machines with the shortest processing time. Thus, after at most m selfish moves the value $min_i L_i$ will increase by at least l' where l' is the length of the shortest job, which also means the value of δ will decrease by at least l'. When δ is reduced to less than l', no more best responses can be made and a Nash solution is reached. Let N be the number of best responses needed to reduce δ to less than l', we have:

$$\delta - \frac{N}{m} \times l' < l' \Rightarrow N > m(\delta - l')/l'.$$

Hence, the number of best responses needed to change x into a Nash solution is at most the smallest N that satisfies the above inequality which is $O(mL)$ when $\delta = L$ and $l' \ll L$. Now to strengthen the bound, instead of using an arbitrary job for the next best response, pick one from one of the critical machines. Note that after the move, the machine considered may no longer be a critical machine. Repeat this process for resulting new schedule and so on until no more best responses are possible for the jobs in a critical machine. This critical machine together with the jobs in it can then be removed from consideration. The process is then continued until all the machines are removed. Now consider all the removed machines with the jobs, the resulting schedule is a Nash solution and the number of selfish moves made is at most $m \times n$. □

The constructive proof of Lemma 3 gives us a polynomial time algorithm for constructing a Nash solution from an arbitrary schedule by using only selfish moves until convergence. The proof also shows why the procedure converges: every selfish move reduces the value of δ and since δ cannot be reduced beyond 0, the procedure converges. This procedure is sometimes called *Nashification*, e.g. in [11]. Using the 2 lemmas from above, we have the following:

Theorem 4. *In congestion model with m machines, the best Nash solution is an optimum solution.*

Proof. Starting with an optimum solution x^*, by Lemma 3 we can turn this into a Nash solution x' by repeated application of selfish moves, and by Lemma 2 we have $F(x') \leq F(x^*)$ since only selfish moves are used which means x' is also an optimum solution. □

As a consequence, because the best Nash solution is optimal, any algorithm that computes the best Nash can be used directly for computing the optimum. And since finding the optimum solution for identical machine scheduling is NP-hard even for $m = 2$ [12], we arrive at the following.

Corollary 5. *Computing the best Nash solution for congestion model in a m-identical machine scheduling game is NP-hard, even for $m = 2$.*

Another consequence is the following approximation result.

Corollary 6. *Any polynomial time ϵ-approximation algorithm for a given machine scheduling problem can be transformed into a polynomial time ϵ-approximation algorithm for computing the best Nash in the corresponding scheduling game.*

Since the best Nash and the optimum solution have the same cost, any polynomial time algorithm that approximate the optimum with ratio ϵ can also be used to approximate the best Nash solution with the same ratio by appending the Nashification procedure described above into the algorithm. Since the Nashification procedure requires polynomial number of steps and does not increase the cost of the initial schedule, the same ratio ϵ holds.

4.2 Generalization to Potential Games

The scheduling games we have considered thus far belong to a larger class of games called the *potential games*

[14]. A potential game is a game that exhibits a potential function $\phi(x)$ on the set of strategy profiles such that, if x' is a profile obtained by changing the strategy of one player j in x then $u_j(x') > u_j(x)$ implies $\phi(x') > \phi(x)$. The existence of this function is typically used to show the existence of pure Nash equilibrium.

Interestingly, although the result in the previous section are derived for scheduling games, it also applies to the larger class of potential game, thus can be used to characterize the behavior of the best Nash equilibrium in this class. We define congestion model in potential games such that given a strategy profile, a utility-improving move on this profile does not reduce the overall quality of the profile. More precisely, the generalized congestion model is a game consisting of a cost function $F(x)$ and a potential function $\phi(x)$, such that if x' is a strategy profile obtained from x by one step utility-improving move on player j ($u_j(x') > u_j(x)$), then:

1. $F(x') \leq F(x)$, and
2. $\phi(x') > \phi(x)$.

The following lemma is known for potential games:

Lemma 7. *[14] Every finite potential game has the finite improvement property, and every maximal improvement path terminate in a Nash equilibrium.*

An improvement path is a sequence $(x^0, x^1, ...)$ of strategy profiles such that for every $k > 0$ there is a utility-improving move made by one job from x^{k-1} to x^k. A game has the finite improvement property if every improvement path is finite. We can then state the following:

Theorem 8. *In a potential game with congestion model, the best Nash solution is an optimum solution.*

Proof. Similar to the proof of theorem 4, let the optimum solution x^* be the initial point of a maximal improvement path. By Lemma 7, this maximal improvement path is finite and terminates in a Nash equilibrium. Each step in the path is a utility improvement move by one player and by definition, this move does not increase the cost of the overall profile. Thus the Nash equilibrium is also optimal. □

A point of clarification may be needed. Lemma 7 merely establishes that potential games have finite improvement property. Generally, the number of steps required by the maximal improvement move to terminate can be very large. Hence to prove NP-hardness for scheduling games under congestion model (Corollary 5), one still needs to show that there is a $polynomial$ reduction from optimum to the best Nash, and we did this in Lemma 3 (rather than directly using Lemma 7). Similarly, the polynomial reduction is also necessary to establish Corollary 6.

5 Sequencing Model

In contrast with the congestion model discussed above, in a sequencing model, a job is released by a machine immediately after it has been processed. This means that a sequencing policy is needed for a machine to determine the order of processing a given set of jobs on that machine. Examples of commonly used policies are: SPT policy (to process the jobs in order of nondecreasing processing time) and LPT policy (to process the jobs in order of nonincreasing processing time). A job j is said to precede a job k (denoted $j \preceq k$) iff job j will always be processed before or at the same time as job k. Assuming that all the machines have the same policy (e.g. all SPT or all LPT), then there is a total ordering on the jobs to be processed across all machines no matter whether the machines are identical or related (ties broken arbitrarily). Hence, we will use an ordering $\preceq$ on the set J to represent the sequencing policy used.

5.1 Sequencing Model in Identical Machines

We will start first by considering identical machines. Recall that the utility of a job j given a strategy profile x in identical machines is given by:

$$u_j(i, x_{-j}) = -\sum_{k:x_k=i, k\preceq j} l_k.$$

The expression above shows that the utility of a job j will only depend on the decisions of the jobs that precede j. For the first job in the ordering, its utility does not depend on the decisions of the other jobs because no matter what the decisions of the other jobs are, it will always be processed first and its completion time depends only on the machine it chooses. For the second job, its utility depends only on the decision of the first job and so on. Since each job wants to maximize its own utility, the set of strategies that the job will take is exactly its best response. By definition, the best response of a job j is the following:

$$R_j(x_{-j}) = \arg\min_{i\in M} \left(l_j + \sum_{k:k\neq j, x'_k=i, k\preceq j} l_k \right).$$

Like its utility, the best response of a job j depends only on the decisions of the jobs that precedes j. The following result states that under sequencing model with identical machines, all the Nash solutions have the same cost.

Theorem 9. *The Nash equilibrium solutions for scheduling games with identical machines using a sequencing policy $\preceq$ have a unique cost.*

Proof. Let π_k denotes the k^{th} job in the ordering given by $\preceq$, we can then construct the best responses of the jobs sequentially by assuming that each jobs will only choose the strategy from its best response and show that: each job's utility is uniquely determined and is independent of which of the best response strategies are chosen by the job and the previous jobs. This is shown by using strong induction as follows:

1. (base step) The best response of the first job in the ordering is $R_{\pi_1} = M$ since all the machines have the same speed. And this is the only best response of the first job no matter what the strategies of the other jobs are, its utility by choosing from its best response is $-l_{\pi_1}$.
2. (base step) The best response of the second job is $M \backslash \{x_{\pi_1}\}$ if $|M| > 1$. Generally, for $1 < k \leq |M|$, $R_{\pi_k} = M \backslash \{x_{\pi_1}, ..., x_{\pi_{k-1}}\}$ if all the jobs $\pi_1, .., \pi_{k-1}$ choose the strategy from their respective best response, i.e. the job will always choose the available empty machine. The utility achieved by job π_k by playing its best response is thus $-l_{\pi_k}$. This is true no matter which of the best response strategies are chosen by this job and the previous jobs.
3. (induction step) For $k > |M|$, by definition, the best response of job π_k is:

$$R_{\pi_k}(x_{\pi_1}, ..., x_{\pi_{k-1}}) = \overset{\arg\min}{i \in M} \sum_{j:1 \leq j \leq k-1, x_{\pi_j} = i} l_{\pi_j},$$

   i.e. its best response R_{π_k} is the set of machines with the minimum total processing time with all the previous jobs scheduled. $\forall i \in R_{\pi_k}$, let j_i be the latest job being processed by i in the current state (before π_k chooses its strategy). All these jobs have the same completion time thus the same utility. The utility of job π_k by choosing any strategy $i \in R_{\pi_k}$ is $u_{\pi_k} = (u_{j_i} - l_{\pi_k})$. Since $j_i \preceq \pi_k$, by induction hypothesis u_{j_i} is uniquely determined, thus u_{π_k} is also uniquely determined and is independent of the best response strategies chosen so far. This ends the induction step.

Since the jobs choose their strategy only from their respective best response, by definition, the set of possible resulting schedules (after the last job chooses its strategy) is exactly the set of Nash solutions. The cost of the Nash solution is the latest completion time among all jobs. Since this value is uniquely determined and is independent of the best response strategies chosen, the cost of the Nash solutions is uniquely determined. □

As a consequence of Theorem 9, the best Nash solution is no better than the worst Nash solution under sequencing model with identical machines, and hence the price of stability is exactly the price of anarchy. The proof of the theorem also shows how to compute the Nash solutions. Note that although the characteristic of the Nash solution derived from above does not depend on a particular sequencing policy used, the exact bound on the POS does. With Theorem 9, we obtain the following two POS results directly via the results on POA for identical machines, due respectively to [1] and [6]:

Corollary 10. *The price of stability in scheduling games with m identical machines using SPT sequencing policy is $2 - 1/m$.*

Corollary 11. *The price of stability in scheduling games with m identical machines using LPT sequencing policy is $4/3 - 1/(3m)$.*

Note that the POS of Theorem 10 is different from the one stated in [13] which is $2 - 2/(m + 1)$. The result in [13] is derived from the bound for the classical Ibarra-Kim algorithm which produces a locally optimum solution for the scheduling problem [18]. We like to point out that tere is a subtle but important difference between a locally optimal solution and a Nash solution. Consider the following example: 3 jobs with length l_1, l_2 and l_3 are to be scheduled on $m = 2$ identical machines with SPT sequencing policy. Let's assume that $l_1 = l_2 = l_3/2$, and that on the same machine, l_1 will be processed first before l_2. The optimal schedule in this case is to have job 1 and 2 in one machine and job 3 in the other, with optimal cost of l_3. The unique Nash solution has job 1 and 2 scheduled on different machines, and job 3 can be in either machines, for example we put job 3 on top of job 2. The Nash solution has the cost of $l_3/2 + l_3 = (3/2)l_3$. The loss of efficiency in term of ratio is $(3/2)l_3/l_3 = 3/2$, which is greater than $2-2/(m+1) = 4/3$. The Nash solution however is not locally optimal because job 2 can move to the other machine and improve the quality of the schedule. While a Nash solution is reached when no job can move and improve its *own* utility, a locally optimal solution is reached when no job can move and improve the *overall* quality of the schedule. In this case, as shown by the example, they do not imply each other.

5.2 Sequencing Model in Related Machines

For related machines, the result presented above does not follow because the argument in the proof of Theorem 9, that all job utilities can be uniquely determined regardless of the best response strategies chosen, does not hold. As a counter example consider the following game: 3 jobs with length $l_1 = 10$, $l_2 = 20$ and $l_3 = 30$ are to be scheduled on

two machines with speed $s_1 = 6$ and $s_2 = 4$ using SPT sequencing policy. The best response for each of the jobs can be derived as follows:

1. Job 1's utility does not depend on the strategies of the other 2 jobs, and its best response is $R_1 = \{1\}$ since $s_1 > s_2$. Its completion time is $5/3$ and is uniquely determined.
2. Job 2's utility depends only on job 1's strategy. If job 1 plays its best response, job 2's best response is $R_2(x_1 = 1) = \{1, 2\}$ since both machines give the same completion time: $(10 + 20)/6 = 20/4 = 5$ and is uniquely determined.
3. Job 3's utility depends on both job 1 and 2's strategies. Since job 2 can play either strategy from its best response, the best responses for job 3 are:
   (a) $R_3(x_1 = 1, x_2 = 1) = \{2\}$, with completion time $30/4 = 7.5$ which is smaller than $(10 + 20 + 30)/6 = 10$, its completion time if machine 1 is chosen.
   (b) $R_3(x_1 = 1, x_2 = 2) = \{1\}$, with completion time $(10 + 30)/6 = 6.67$ which is smaller than $(20 + 30)/4 = 12.5$ if machine 2 is chosen.

Thus the completion time for job 3 is not unique and depends on which strategy is played by job 2 from its best response. Consequently, the only 2 Nash solutions: $(1, 1, 2)$ and $(1, 2, 1)$ have different costs of 7.5 and 6.67 respectively and the gap between the best and the worst Nash solution in this example is $9/8$. This example also tells us that the gap between the best and the worst Nash in sequencing model with 2 related machines is at least $9/8$. One future direction for this work is to bound the gap between the best and the worst Nash solution and generalize it to m related machines.

6 Conclusion and Future Work

In this paper we studied a class of scheduling games, and provided results on the computational complexity for computing the best Nash equilibrium, as well as the quality guarantee (price of stability) for such a solution. While we have closed the problem on the congestion model, we only managed to obtain results for identical machines on the sequencing model. Although we managed to show a gap between the worst and the best Nash equilibrium for related machines, the bound on this gap is still unknown and left as future work. It is also interesting to extend the investigation on a more general class of congestion (potential) games.

Acknowledgments This research was supported partially by the Singapore Management University under grant number C220/MSS6C007.

References

[1] E. Angel, E. Bampis, and F. Pascual. Truthful algorithms for scheduling selfish tasks on parallel machines. In *Theoretical Computer Science*, 2006.

[2] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability in network design with fair cost allocation. In *Symposium on Foundations of Computer Science (FOCS)*, 2004.

[3] Y. Azar, J. Naor, and R. Rom. The competitiveness of online assignments. In *Journal of Algorithms*, volume 18, pages 221–237, 1995.

[4] B. Blum, C. R. Shelton, and D. Koller. A continuation method for nash equilibria in structured games. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[5] X. Chen and X. Deng. Settling the complexity of 2-player nash-equilibrium. In *ECCC Report*, 2005.

[6] G. Christodoulou, E. Koutsoupias, and A. Nanavati. Coordination mechanisms. In *International Colloquium on Automata, Languages and Programming (ICALP)*, 2004.

[7] A. Czumaj and B. Vocking. Tight bounds for worst-case equilibria. In *Symposium on Discrete Algorithms (SODA)*, 2002.

[8] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a nash equilibrium. In *Symposium on Theory of Computing (STOC)*, 2006.

[9] C. Daskalakis and C. H. Papadimitriou. Three-player games are hard. In *ECCC Report*, 2005.

[10] A. Fabrikant, C. H. Papadimitriou, and K. Talwar. The complexity of pure nash equilibria. In *Symposium on Theory of Computing (STOC)*, 2004.

[11] M. Gairing, T. Lucking, M. Mavronicolas, and B. Monien. Computing nash equilibria for scheduling on restricted parallel links. In *Symposium on Theory of Computing (STOC)*, 2004.

[12] M. R. Garey and D. S. Johnson. *Computer and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[13] N. Immorlica, L. Li, V. S. Mirrokni, and A. Schulz. Coordination mechanisms for selfish scheduling. In *Internet and Network Economics*, volume 3828 of *Lecture Notes in Computer Science*, pages 55–69. Springer, 2005.

[14] D. Monderer and L. S. Shapley. Potential games. In *Games and Economic Behavior*, volume 14, pages 124–143, 1996.

[15] C. Papadimitriou. Algorithms, games, and the internet. In *Symposim on Theory of Computing (STOC)*, pages 749–753, 2001.

[16] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a nash equilibrium. In *National Conf. on Artificial Intelligence (AAAI)*, 2004.

[17] T. Sandholm, A. Gilpin, and V. Conitzer. Mixed-integer programming methods for finding nash equilibria. In *National Conf. on Artificial Intelligence (AAAI))*, 2005.

[18] P. Schuurman and T. Vredeveld. Performance guarantees of local search for multiprocessor scheduling. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2001.