Operations Research Letters 43 (2015) 530–533

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Sequential scheduling on identical machines

Refael Hassin *, Uri Yovel †

Department of Statistics and Operations Research, School of Mathematical Sciences, Tel-Aviv University, Tel-Aviv 69978, Israel

ARTICLE INFO

Article history:
Received 5 December 2013
Received in revised form
13 July 2015
Accepted 7 August 2015
Available online 17 August 2015

Keywords:
Sequential price of anarchy
Machine scheduling
Congestion games
Load balancing
Subgame-perfect equilibrium
Makespan minimization

ABSTRACT

We study a sequential version of the KP-model: Each of n agents has a job to be processed on any of m machines. Agents *sequentially* select a machine for processing their jobs. The goal of each agent is to minimize the completion time of his machine. We study the *sequential price of anarchy* for m identical machines under arbitrary and LPT orders, and suggest insights into the case of two unrelated machines.

1. Introduction

We study the following dynamic game. There are n agents, denoted $A_1, \ldots, A_n$, and m machines. Agent A_j has a job that takes $p_j > 0$ time units if processed by any of the machines. Agents *sequentially* select one of the machines for processing their jobs, starting with A_1 and ending with A_n. While choosing a machine, an agent knows the choices made by his predecessors. Once a machine completes processing all the jobs assigned to it, they are instantaneously delivered to their agents. The goal of each agent is to have his job delivered at the earliest possible time. We study the corresponding *sequential price of anarchy*, denoted **SPoA**, which is the cost-ratio of the worst subgame-perfect equilibria of such games to the solution that minimizes the overall makespan of the system.

The above model is a *sequential* version of the well-known KP-model introduced by Koutsoupias & Papadimitriou [7]. While the KP-model has been widely studied and extended, it seems that in many cases, a more appropriate model should include some form of sequentiality, since agents may *arrive at different times*. Consequently, a new line of algorithmic research has been initiated recently, which studies sequential versions of games whose simultaneous counterparts are well-studied. In their paper [8], Leme et al. define the notion of *sequential price of anarchy* (**SPoA**). The agents are indexed by their "order of arrival" and they choose their actions *sequentially*, knowing only the choices made by their predecessors.

We analyze the price of anarchy for m identical machines. Specifically, we prove that **SPoA** is at most $2 - \frac{1}{m}$, and this bound is tight. If the agents are ordered in nonincreasing order of their job's processing times (this is the well-known LPT rule), then this bound on **SPoA** is reduced to $\frac{4}{3} - \frac{1}{3m}$. These bounds coincide with the approximation ratios of the Greedy algorithm (i.e., each agent chooses a least loaded machine) in the classical *List Scheduling model* of Graham [5]; however, the proof is inherently different. Essentially, it is because in our model, the agents are selfish, so they need not choose a least loaded machine. In fact, we demonstrate that the greedy strategy may be bad for an agent. We also discuss **SPoA** for two unrelated machines, for which we conjecture that it is bounded by 3.

We believe that our sequential framework can be developed for other combinatorial optimization problems.

Leme et al. [8] prove that for *unrelated* machines, the worst **SPoA** is bounded between $\Omega(n)$ and $O(m \cdot 2^n)$; they also study the **SPoA** for other games, and in [9], they study **SPoA** of sequential auctions. In [2], Biló et al. improve the above bounds to $2^{\Omega(\sqrt{n})}$ and 2^n, respectively. In [1], Angelucci et al. study the **SPoA** of Isolation Games, and in [3] de Jong et al. study a sequential decision variation of their main model where each player controls a set of machines and wishes to maximize the value of jobs that can be feasibly scheduled on its machines.

Fiat et al. [4] consider another situation where selfish agents choose their time of transmission in a shared communication media. Transmission is successful only if there is no simultaneous

* Corresponding author.
E-mail address: hassin@post.tau.ac.il (R. Hassin).
† Uri Yovel deceased June 2014.

http://dx.doi.org/10.1016/j.orl.2015.08.003
0167-6377/

transmission at this time. The main difference between this model and ours is that decisions are made simultaneously and when transmission fails the agent can repeat trying in a later time, whereas in our case decisions are made sequentially and no regret is possible.

2. Notation

Let $N \equiv \{1, \ldots, n\}$, $M \equiv \{1, \ldots, m\}$. Thus, as we introduced above, the n agents are A_j, $j \in N$. We denote the m machines (or processors) by M_i, $i \in M$. Agent A_j has a job, denoted j, that takes $p_j > 0$ time units on any of the machines (i.e., they are identical). We denote the list of processing times of all the agents by $p := (p_1, \ldots, p_n)$. Each agent selects one of the machines for processing his job, thus, the action set for each A_j is M. In step j of the game, A_j observes the current loads on all the machines, i.e., he knows the actions chosen by $A_1, \ldots, A_{j-1}$, and chooses a machine for processing his job. Hence, the strategy for A_j is a function $s_j : M^{j-1} \to M$. We denote $j \in M_i$ if $s_j = i$, i.e., A_j chooses M_i for his job, and we say that *M_i is A_j's machine*. After all agents chose their machines, i.e., the *strategy profile* $s \equiv (s_1, \ldots, s_n) \in M^n$ has been determined, a complete job schedule is obtained. For a given such schedule, we denote by S_j and C_j the start time and the completion time of j, respectively ($j \in N$), and by L_i the (final) load on M_i, i.e., $L_i \equiv \sum_{j \in M_i} p_j$; our schedule will always be clear from the context, so we do not write $S_j(s)$, $C_j(s)$ etc. We denote the *average load* by $\bar{L} := \frac{1}{m}\sum_{i=1}^{m} L_i = \frac{1}{m}\sum_{j=1}^{n} p_j$. We denote by $C_{\max}$ the *makespan* of the schedule, i.e., $C_{\max} = \max_{j \in N} C_j = \max_{i \in M} L_i$.

Once a machine completes processing all the jobs assigned to it, they are (instantaneously) delivered to their agents, hence the cost of A_j is the (final) load of his machine, i.e., it is L_i satisfying $j \in M_i$. The goal of each agent is to minimize this cost, i.e., to have his job delivered at the earliest possible time. This is an *extensive form game*, and so it always possesses (pure) *subgame perfect equilibria*, which can be calculated by *backward induction*. Consequently, we refer to any schedule of the jobs which was obtained by the sequential decision process as a *subgame perfect equilibrium*, or *equilibrium* for short. We denote by *SPE* the set of these equilibria (corresponding to the given game in context).

We denote by $C^*_{\max}$ the makespan of an *optimal schedule*, i.e., it is the minimum possible value of the makespan of the system, ideally achieved if a central authority were to schedule the entire set of jobs.

We study the corresponding *sequential price of anarchy* of the game, denoted **SPoA**, which is the cost-ratio of the worst subgame-perfect equilibrium to the optimal makespan, that is:

Definition 2.1 (*Sequential Price of Anarchy [8]*)**.**

$$\textbf{SPoA} = \max_{s \in SPE} \frac{C_{\max}(s)}{C^*_{\max}}.$$

($C_{\max}(s)$ is $C_{\max}$ in the schedule corresponding to the strategy profile $s \in M^n$.)

3. SPoA for identical machines

We start with three simple examples with $n = 3$ agents and $m = 2$ machines. In these examples we use a sufficiently small $\epsilon > 0$ in one of the processing times, so no ties ever occur. However, the conclusions are also valid when $\epsilon = 0$, and the bounds obtained are exact and not asymptotic. If $\epsilon = 0$ the obtained solution is just one of several possibilities, so any implementation should include a tie-breaking rule. Thus, we use ϵ in the examples for convenience, but when we refer to **SPoA**, we substitute $\epsilon = 0$, since this always yields the highest possible value.

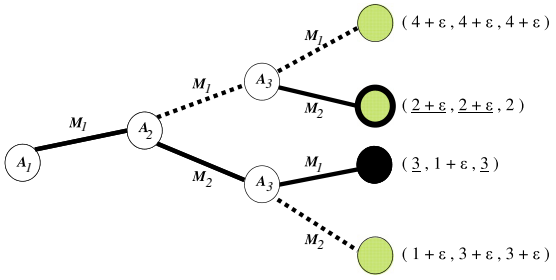

Fig. 1. The game-tree associated with Example 3.1 ($n = 3, m = 2$).

Example 3.1. There are $n = 3$ agents and $m = 2$ machines. The processing times are $p = (p_1, p_2, p_3) = (1, 1+\epsilon, 2)$. Without loss of generality, assume that A_1 chooses M_1 for his job. Then A_2 chooses M_2 for his job, since he realizes that in this case A_3 will choose M_1, so A_2's cost will be $L_2 = p_2 = 1+\epsilon$, which is best possible. Thus, in the resulting equilibrium, jobs 1 and 3 select M_1 (we write this as $1, 3 \in M_1$) $2 \in M_2$, hence the loads are $L_1 = 3$, $L_2 = 1+\epsilon$, so $C_{\max} = 3$. However, the optimal schedule (with minimum possible makespan) is $1, 2 \in M_1$ and $3 \in M_2$, achieving $C^*_{\max} = 2+\epsilon$. Consequently, **SPoA** $= \frac{3}{2}$.

Fig. 1 depicts the game-tree associated with Example 3.1. Vectors at the leaves are the cost vectors (i.e., the loads). The solid lines show the subgame-perfect strategies, and the (unique) path from the root to the leaf corresponding to the black circle is the equilibrium solution. The bold circle corresponds to an optimal solution. The values of $C_{\max}$ and $C^*_{\max}$ are underlined in their corresponding cost vectors.

Note that in the above example, each agent acts *greedily*, i.e., chooses a least loaded machine. However, the next example shows that this strategy need not yield a minimal makespan for that agent.

Example 3.2. Consider the previous example with a small change to the processing times so that $p = (1, 1-\epsilon, 2)$. Let A_1 choose M_1. Now A_2 reasons as follows: if he chooses M_2 for his job (which is currently empty), then A_3 will choose M_2 as well, hence A_2's cost will be $3-\epsilon$. However, if he chooses M_1, A_3 will choose M_2, and so A_2's cost will be $2-\epsilon$. Thus, in the resulting equilibrium, $1, 2 \in M_1$ and $3 \in M_2$, hence the loads are $L_1 = 2-\epsilon$, $L_2 = 2$, so $C_{\max} = 2$. It is easily verified that this is also an optimal schedule, i.e., $C^*_{\max} = 2$. Consequently, **SPoA** $= 1$.

Examples 3.1 and 3.2 also demonstrate that an agent may prefer to have his job longer, *ceteris paribus*: the fact that p_2 is $1+\epsilon$ (in 3.1) rather than $1-\epsilon$ (in 3.2) gives A_2 considerable advantage.

Observe that the order of agents crucially affects the outcome:

Example 3.3. Consider again Example 3.1 but change the order to $p = (2, 1, 1+\epsilon)$. Then, the equilibrium is $1 \in M_1$, $2, 3 \in M_2$, yielding $C^*_{\max} = C_{\max} = 2+\epsilon$; consequently, **SPoA** $= 1$.

We now show that the sequential price of anarchy is at most $2 - \frac{1}{m}$. This bound matches the bound on the approximation ratio of the greedy algorithm in Graham's *List Scheduling*, however, the additional step of Lemma 3.1 is required because the agents are selfish, so (when $m > 2$) they may apply a strategy other than the greedy one, that is, *a-priori they need not choose a least loaded machine*.

The following example demonstrates this fact:

Example 3.4. Consider $m \geq 3$ machines and $n = m+1$ jobs, whose processing times are $(p_1, \ldots, p_{m+1}) = (K, 1-\epsilon, 1, 1, \ldots, 1, K+\epsilon)$. Let A_1 choose M_1. We claim that in the resulting schedule, A_2 will schedule his job (with $p_2 = 1-\epsilon$) on M_1 (starting at