Theoretical Computer Science 570 (2015) 40–54

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Coordinating oligopolistic players in unrelated machine scheduling

Fidaa Abed *, Chien-Chung Huang

Max-Planck-Institut für Informatik, Saarbrücken, Germany

ARTICLE INFO

Article history:
Received 28 May 2014
Received in revised form 16 September 2014
Accepted 31 December 2014
Available online 9 January 2015
Communicated by X. Deng

Keywords:
Unrelated machine scheduling
Coordination mechanisms
Price of anarchy

ABSTRACT

We consider the following machine scheduling game. Jobs, controlled by selfish players, are to be assigned to unrelated machines. A player cares only about the finishing time of his job(s), while disregarding the welfare of other players. The outcome of such games is measured by the makespan. Our goal is to design coordination mechanisms to schedule the jobs so as to minimize the price of anarchy.

We introduce *oligopolistic players*. Each such player controls a set of jobs, with the aim of minimizing the sum of the completion times of his jobs. Our model of oligopolistic players is a natural generalization of the conventional model, where each player controls only a single job.

In our setting, previous mechanisms designed for players with single jobs are inadequate, e.g., having large price of anarchy, or not guaranteeing pure Nash equilibria. To meet this challenge, we design three mechanisms that are adapted/generalized from Caragiannis' ACOORD. All our mechanisms induce pure Nash equilibria while guaranteeing relatively small price of anarchy.

1. Introduction

We consider the game-theoretic version of the following machine scheduling problem. A set $\mathcal{I}$ of jobs and a set $\mathcal{M}$ of machines are given. A job $i \in \mathcal{I}$ has weight w_{ij} on machine $j \in \mathcal{M}$. Jobs are to be assigned to machines and the goal is to minimize the makespan. This is the classical *unrelated machine scheduling* problem with the makespan objective [21].

A natural question from the angle of game theory is what happens if the jobs are controlled by selfish players. The strategy space of a player is the set of machines. A player cares only about the finishing time of his job while disregarding the welfare of other players. In such games, researchers especially focus on the *Nash equilibrium* [22], a stable situation where no player can unilaterally change his strategy to *strictly* improve the finishing time of his job. In case that all players use pure strategies only, the Nash equilibrium resulted is called the pure Nash equilibrium (PNE). In this paper, we consider only pure strategies and PNEs.

A central question in algorithm game theory is the quality of equilibria. In particular, people analyze the *price of anarchy* (PoA) [20], which in our context is defined as the worst ratio between the makespan of a PNE against that of the optimal solution.

* Corresponding author.
E-mail addresses: fabed@mpi-inf.mpg.de (F. Abed), villars@mpi-inf.mpg.de (C.-C. Huang).

http://dx.doi.org/10.1016/j.tcs.2014.12.022
0304-3975/

Table 1
Summary of the properties of known mechanisms in our model. $m = |\mathcal{M}|$ is the number of machines, and C is the largest number of jobs controlled by a player. The results marked by $\star$ are proved in Appendix A. For the last three mechanisms, $p \geq 1$, and ϵ is some small constant where $\epsilon > 0$. If $p = \Theta(\log m)$ and $C = 1$, then the PoAs for ACOORD, BCOORD, and CCOORD are $\Theta(\log m)$, $\Theta(\frac{\log m}{\log \log m})$, and $O(\log^2 m)$ respectively.

Mechanisms	PoA		PNE	
	$C = 1$	$C > 1$	$C = 1$	$C > 1$
ShortestFirst [19]	$\Theta(m)$	$\Omega(m)$	Yes	No$\star$
LongestFirst [19]	Unbounded	Unbounded	No	No
Makespan [19]	Unbounded	Unbounded	Yes	No$\star$
RANDOM [19]	$\Theta(m)$	$\Omega(m)$	No	No
EQUI [9]	$\Theta(m)$	$\Omega(m)$	Yes	Yes
AJM-1 [5]	$\Theta(\log m)$	$\Omega(\log m)$	No	No
AJM-2 [5]	$\Theta(\log^2 m)$	$\Omega(\log^2 m)$	Yes	No$\star$
BALANCE [10]	$\Theta(\log m)$	$\Omega(\log m)$	Yes	No$\star$
ACOORD [7]	$O(p \cdot m^{1/p})$	$\Omega(C^{(1-\epsilon)(p+1)} m/p^2)\star$	Yes	Yes
BCOORD [7]	$O(p \cdot m^{1/p}/\log p)$	$\Omega(C^{(1-\epsilon)(p+1)} m/p^2)\star$	No	No$\star$
CCOORD [7]	$O(p^2 \cdot m^{1/p})$	$\Omega(C^{(1-\epsilon)(p+1)} m/p^2)$ when $p = 1\star$	Yes	No$\star$

The PoA is obviously determined by the rules of the game. Here the "rules" mean the scheduling policies of the machines. In the literature, the rules are formally called the *coordination mechanisms* [8]. Ideally, we would like to have good coordination mechanisms so as to minimize the PoA.

The design space of coordination mechanism depends on a number of parameters, e.g., whether preemption is allowed, whether jobs have unique IDs and so on. However, the following two conditions on coordination mechanisms are (implicitly) assumed by all previous works (and the current one).

1. **Physical Feasibility**. Suppose that a set of jobs $\mathcal{I}' \subseteq \mathcal{I}$ are assigned to machine j. At any point of time t, if a subset of jobs $\mathcal{I}'' \subseteq \mathcal{I}'$ are finished by machine j, then $\sum_{i \in \mathcal{I}''} w_{ij} \leq t$.
2. **Locality of Scheduling Decision**. A machine decides its schedule based only on the information of the incoming jobs, while, where the other jobs go to, and how the other machines schedule them is irrelevant.

The first condition is self-evident; the second condition is motivated by the fact that in a fluid environment, such as the Internet, a machine may not be able to coordinate with other machines in a timely manner. Azar, Jain, and Mirrokni [5] differentiate two classes of mechanisms: a mechanism is *local* if a machine schedules its job based only on the information of the incoming jobs (but notice that a machine is allowed to look at the weights of its jobs on other machines); a mechanism is *strongly local* if a machine schedules its jobs only based on the weights of the incoming job on it. It is known that the PoAs of strongly local mechanisms and of local mechanisms can be significantly different when preemption is disallowed [5,15].

Oligopolistic Players. All previous works assume that a player controls a single job. A natural and more realistic extension is to assume a player can control multiple jobs and we refer to such players as *oligopolistic* players. A question that arises in our model is: what would be the local objective of an oligopolistic player? This is a non-issue when a player controls a single job. However, when he has multiple jobs, several objectives are possible. For instance, it could be his makespan (the latest finishing time of his jobs), or it could be the sum of completion times of his jobs.

In this work, we assume that each player aims to minimize the sum of the completion times of his jobs. This assumption is motivated by the observation that a player would care about the *collective welfare* of his jobs. If moving a job from one machine to another machine decreases the finishing time of that job, the controlling player would have incentive to do so—even if the latest finishing time of his jobs is not really decreased.

To evaluate the overall system performance, there can be two natural candidates: makespan (the latest finishing time of a job), or the weighted completion times of the jobs (jobs are given weights and the cost is computed as the weighted sum of their completion times.) In a companion paper of this work [1], we use the weighted completion times of the jobs to measure the system performance. In this work, we instead consider the makespan.

In general, in terms of PoA, it is harder to design mechanisms when the global objective is the makespan than when it is the weighted sum of completion times of all jobs. In the original model where each player controls a single job, with weighted sum global objective, Cole et al. [11] show several mechanisms achieving constant PoA; on the other hand, when the global objective is the makespan, it is known [2,5,15] that constant PoA is impossible. As our model is a generalization of the single-job player model, we also cannot hope to achieve constant PoA.

We observe that previous mechanisms designed for players with single jobs are inadequate in our model of oligopolistic players. In some cases (ACOORD/BCOORD/CCOORD), the PoA becomes significantly worse; in some cases, they no longer guarantee PNEs (ShortestFirst/AJM-2/CCOORD/BALANCE). See Table 1 for a summary of the properties of the known mechanisms in our new model. Our challenge here is to design coordination mechanisms that simultaneously guarantee the existence of PNEs *and* still maintain small PoA.