Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Scheduling selfish jobs on multidimensional parallel machines

Leah Epstein [a,*], Elena Kleiman [b]

[a] Department of Mathematics, University of Haifa, 3498838 Haifa, Israel
[b] Department of Software Engineering, ORT Braude College of Engineering, 2161002 Karmiel, Israel

ARTICLE INFO

Article history:
Received 13 March 2017
Received in revised form 14 June 2017
Accepted 21 June 2017
Available online xxxx
Communicated by C. Kaklamanis

Keywords:
Load balancing
Price of anarchy
Identical machines
Vector scheduling
Selfish agents

ABSTRACT

We study the multidimensional vector scheduling problem with selfish jobs, both in non-cooperative and in cooperative versions, in two variants. These variants differ in the private costs of the jobs: in the first variant, the cost is the ℓ_∞ norm (maximum over components) while in the second variant it is the ℓ_1 norm (sum of components) of the load. We show existence of assignments that are Nash, strong Nash, weakly and strictly Pareto optimal Nash equilibria in these settings. For the first variant, we improve upon the previous bounds on the price of anarchy for the non-cooperative case, and find tight bounds for every number of machines and dimension. For the cooperative case we provide tight bounds on the strong prices of anarchy and stability, as well as tight bounds on weakly and strictly Pareto optimal prices of anarchy and stability, for every number of machines and dimension. For the second variant, which was not considered before, we again consider all the aforementioned measures, and find tight bounds, each of which being a function of the number of machines and the dimension, showing cardinal differences in the behavior of these measures both between the two variants, and in comparison to the one-dimensional case.

1. Introduction

1.1. Motivation and framework

In this paper we study the d-dimensional vector scheduling problem, which is a generalization of the well-known job scheduling problem [23,19]. In this problem, each job has d distinct resource requirements, and each machine has d corresponding resources. Here, we focus on a game-theoretical setting. Such problems are often referred to as 'vector assignment' problems, as each job can be viewed as a d-dimensional vector, where each coordinate is associated with the corresponding requirement. This requirement can be available memory, disk space, CPU power, network bandwidth, or any other quality a computing unit may possess.[1]

Multidimensionality plays an important role in capturing incomparable characteristics of the jobs that are to be scheduled. For example, the memory requirements and the bandwidth requirements of a job in a distributed computer environment are incomparable. Multidimensionality also allows to differentiate between the costs that may be associated with each of these requirements.

* Corresponding author.
E-mail addresses: lea@math.haifa.ac.il (L. Epstein), elena.kleiman@gmail.com (E. Kleiman).

[1] In an accompanying paper [16], we consider also the d-dimensional vector packing problem, that can be seen as a scheduling problem where resources are limited on each component. Even though the motivations for both versions are related, the problems are of different flavors.

http://dx.doi.org/10.1016/j.tcs.2017.06.018
0304-3975/

Multidimensional assignment problems of the kind that we consider in this paper appear in practice. For example, the Google ROADEF/EURO challenge 2011–2012[2] involved a set of machines with several resources, such as RAM and CPU, running processes which consume those resources. Another application of this problem is for resource scheduling for query optimization in parallel databases [8]. The parallel database consists of multiple independent processing units, each having shareable resources. Each query executed on one of these units has requirements for each of these resources, and can be described as a multidimensional load vector. Work in [21,22] demonstrates the practical effectiveness of the multidimensional approach for this setting.

However, so far these problems have received relatively little attention in the literature (see e.g. [20,27,10,8]). In most work on scheduling it is assumed that the load of a job is described by a single aggregate measure. This assumption is typically done to simplify the analysis. But for large task systems, not accounting for the multidimensional nature of the jobs could result in a bad performance and reduced efficiency. With the rapidly growing popularity and importance of large-scale distributed computer environments such as the Internet, and more recently data centers implementing Cloud Computing [36,34,3], it becomes crucial to address various issues that are typical to such settings. Problems concerning efficient resource allocation for better resource utilization, and multidimensional job scheduling and packing in particular, are being two of the most significant problems in Cloud Computing. This fact has rekindled the interest of researchers in these classical optimization problems that have been studied since the early 1970's. A game-theoretic framework is a natural choice for the purpose of studying such problems, as the users of these systems act selfishly, and often even in an uncoordinated manner. The importance of studying scheduling and packing problems under a game-theoretical framework is by now very well established and widely acknowledged (see e.g. [29,11,31,1,18,7,14]). We attempt to add to this body of knowledge by considering the multidimensional variant. A preliminary version of this paper appeared as [15].

1.2. The model

We now define the multidimensional job scheduling problem. There are n jobs, denoted by $J = \{1, 2, \ldots, n\}$, each with $d \geq 1$ components, that are to be assigned to a set of m parallel machines $M = \{M_1, \ldots, M_m\}$ (also called machines $1, 2, \ldots, m$), each with $d \geq 1$ identical resources. A multidimensional job scheduling game, or simply, a vector scheduling game VS is characterized by a tuple

$$VS = \langle N, (\mathcal{M}_k)_{k\in N}, (c_k)_{k\in N} \rangle ,$$

where N is the set of players. Each selfish player $k \in N$ controls a single job and selects the machine to which it will be assigned. We associate each player with the job it wishes to run, that is, $N = J$. The set of strategies $\mathcal{M}_k$ for each job $k \in N$ is the set M of all machines. i.e. $\mathcal{M}_k = M$. Each job must be assigned to one machine only. The outcome of the game is an assignment

$$\mathcal{A} = (\mathcal{A}_k)_{k\in N} \in \times_{k\in N} \mathcal{M}_k$$

of jobs to the machines, where $\mathcal{A}_k$ for each $1 \leq k \leq n$ is the index of the machine that job k chooses to run on. Let $\mathcal{S}$ denote the set of all possible assignments.

Let the vector of a job $k \in \{1, 2, \ldots, n\}$ be denoted by

$$p_k = (p_k^1, \ldots, p_k^d) ,$$

where $p_k^i \geq 0$, $1 \leq i \leq d$ and $\sum_{i=1}^{d} p_k^i > 0$. The value p_k^i is the size or processing time of job k for the ith component. Let $P^i = \sum_{k=1}^{n} p_k^i$ be the total processing time for the ith component. For a fixed schedule $\mathcal{A}$, we let L_ℓ^i denote the load of the ith component for machine M_ℓ, that is $L_\ell^i = \sum_{k:\mathcal{A}_k=M_\ell} p_k^i$. The load of machine M_ℓ is defined to be the maximum load in any dimension observed by machine M_ℓ in this assignment, that is $L_\ell(\mathcal{A}) = \max_{1\leq i\leq d} L_\ell^i$.

The cost function of job $k \in N$ is denoted by $c_k : \mathcal{S} \to \mathbb{R}$. In the first variant of the game that we consider, the cost c_k charged from job k for running on machine M_ℓ in a given assignment $\mathcal{A}$ is defined to be the load of machine M_ℓ, that is $c_k(\ell, \mathcal{A}_{-k}) = L_\ell(\mathcal{A})$, where $\mathcal{A}_{-k} \in \mathcal{S}_{-k}$; here $\mathcal{S}_{-k} = \times_{j\in N\setminus\{k\}} \mathcal{M}_j$ denotes the actions of all players except for player k. Similarly, for $K \subseteq N$ we denote by $\mathcal{A}_{-K} \in \mathcal{S}_{-K}$ the set of strategies of players outside of K in an assignment $\mathcal{A}$, when $\mathcal{S}_{-K} = \times_{j\in N\setminus K} \mathcal{M}_j$ is the action space of all players except for players in K. Each selfish job is interested in reducing its cost by moving to a less loaded machine, if possible, which will result in better QoS. The social cost of an assignment $\mathcal{A}$ is defined as the maximum load on any dimension of a machine, over all the machines, also called makespan, and is denoted by

$$SC(\mathcal{A}) = \max_{1\leq\ell\leq m} L_\ell(\mathcal{A}) .$$

For the cost function defined here and the first variant of the game, the maximum can be taken over the players instead of the machines, as these are equivalent for a non-empty set of players; an empty machine will not achieve the maximum,

[2] http://challenge.roadef.org/2012/en/index.php.

Please cite this article in press as: L. Epstein, E. Kleiman, Scheduling selfish jobs on multidimensional parallel machines, Theoret. Comput. Sci. (2017), http://dx.doi.org/10.1016/j.tcs.2017.06.018