Operations Research Letters 43 (2015) 522–525

Contents lists available at ScienceDirect

Operations Research Letters

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

On the optimality of the LP-based algorithm for online scheduling with GoS eligibility constraints

An Zhang

Department of Mathematics, Hangzhou Dianzi University, Hangzhou 310018, PR China

ARTICLE INFO

Article history:
Received 11 December 2014
Received in revised form
31 July 2015
Accepted 31 July 2015
Available online 7 August 2015

Keywords:
Online scheduling
Grade of Service eligibility
Competitive ratio
Linear programming

ABSTRACT

We consider the online scheduling problem on m identical machines subject to the Grade of Service (GoS) eligibility constraints. The goal is to minimize the makespan. For fractional jobs that can be arbitrarily split between machines and can be processed in parallel, we provide an optimal online algorithm based on the solution of linear programming.

1. Introduction

In recent years, scheduling problems with machine eligibility constraints have received a lot of attention. In such problems, a job cannot be processed by any one of the machines. Instead, it can only be processed by a machine that belongs to a specific subset of the machines, namely *the processing set* of the job. For the problem with arbitrary processing sets, which is also called the restricted assignment problem, Azar et al. [1] studied the online version of minimizing the makespan on m identical machines, where 'online' means that jobs are revealed one by one along some list that might be terminated at any time. Thus any online algorithm has to immediately assign the current job to a machine without any knowledge of the remaining jobs in the list. In [1], Azar et al. provided a greedy-like algorithm $\mathcal{AW}$ to solve the problem and showed that its competitive ratio is no more than $\lceil \log_2 m \rceil + 1$. In addition, they presented a lower bound $\lceil \log_2(m+1) \rceil$ for the problem, which indicates that $\mathcal{AW}$ is optimal when m is a power of 2. Hwang et al. [10] observed that a further analysis on the algorithm can imply a slightly smaller competitive ratio of $\log_2 m + 1$. Recently, Lim et al. [17] improved both the competitive ratio of the algorithm $\mathcal{AW}$ and the lower bound of the problem. They obtained a new competitive ratio $\lfloor \log_2 m \rfloor + \frac{m}{2^{\lfloor \log_2 m \rfloor}}$ and showed that the gap between the competitive ratio and the lower bound is no more than an irrational number which is approximately 0.1967. It is also observed that algorithm $\mathcal{AW}$ is optimal when the number of machines can be written as a sum of two powers of 2, i.e., $m = 2^k + 2^{k'}$ for $k \neq k'$. Consequently, the algorithm AW is optimal for any $m \leq 10$ except $m = 7 = 2^1 + 2^2 + 2^0$, for which, Lim et al. [17] further improved the gap to $\frac{1}{180} \approx 0.00556$.

Hwang et al. [11] addressed a scheduling problem in service provision application, where the processing sets are inclusively structured to distinguish different grades of service level customers. Their problem is called machine scheduling subject to the Grade of Service (GoS) eligibility constraints or simply the hierarchical scheduling in the literature. Various discussion on the complexity and the offline algorithms have been made, where we refer the reader to Leung and Li's survey paper [16]. For online scheduling, Bar-Noy et al. [2] presented a $e + 1$-competitive algorithm for the identical machines, where $e \approx 2.71828$ is the base of the natural logarithm. Another algorithm is shown to be e-competitive for the fractional case that each job can be arbitrarily split between machines and different parts of the same job can be processed in parallel. They also provided a matching lower bound for the fractional case. However, these bounds hold only when the number of machines goes to infinity. Thus, it deserves to be considered the problems with a specific number of machines. Tan and Zhang gave such an attempt in [22], where a new algorithm for the fractional case is designed. By constructing a linear programming (LP) formulation, their algorithm can handle any finite number of machines. As it turns out, the LP-based algorithm is optimal by numerical calculation. For the original problem without fractional jobs, they developed a de-fractional LP algorithm which improves the result in [2] as well.

Other research on scheduling with the GoS eligibility constraints mostly focuses on either a small number of machines or

E-mail address: anzhang@hdu.edu.cn.

http://dx.doi.org/10.1016/j.orl.2015.07.008
0167-6377/

a relatively simple setting on the processing sets. Park et al. [20] and Jiang et al. [13] independently proposed an optimal algorithm for two identical machines, which has a competitive ratio of $\frac{5}{3}$. For $m = 3$, Zhang et al. [26] claimed an optimal algorithm with competitive ratio 2. Tan and Zhang [22] presented algorithms for $m = 4$ and $m = 5$ with competitive ratios of 2.333 and 2.610, respectively. Lim et al. [17] developed improved algorithms with competitive ratios 2.294 and 2.501 for $m = 4$ and $m = 5$, respectively. Chassid and Epstein [3], Dosa and Epstein [5], and Tan and Zhang [21] considered the online problem on two uniform machines, with the fractional case, the preemptive case and the general case included, where optimal algorithms, as well as the associated competitive analysis with respect to the speeds of the machines, are proposed. In [12], Jiang introduced a problem where there are only two inclusive processing sets for all jobs, i.e., each job can either be processed by all machines or by only the first k machines. He provided a $\frac{12+4\sqrt{2}}{7} \approx 2.522$-competitive algorithm. Zhang et al. [27] further designed an improved algorithm with competitive ratio $1 + \frac{m^2-m}{m^2-km+k^2} < \frac{7}{3}$ and proved a lower bound of 2 when $k \geq 3$ and $m \geq \frac{3}{2}(k+1)$. Hou and Kang extended the models to m uniform machines with two different speeds [8,9], where both the general case and the fractional case are considered. There is also research considering the semi-online problems that some partial information on jobs are known in advance, see e.g., [19,18,14,25,4,24]. For more online results on scheduling with the GoS eligibility constraints, we refer the reader to [15,23].

In this paper, we consider online scheduling on m identical machines subject to the GoS eligibility constraints. Formally, we are given a set of m identical machines $\{M_1, M_2, \ldots, M_m\}$ and a sequence of jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ that arrive online one by one. Once a job is presented to the online algorithm, it has to be placed immediately on the machines. A job J_j with processing time p_j can be assigned to a machine M_i only if the machine belongs to the job's processing set. Due to the GoS eligibility constraints, we assume that there are exactly m different processing sets. And for simplicity, let us denote them by $\mathcal{M}_k = \{M_1, M_2, \ldots, M_k\}$ for $1 \leq k \leq m$. Our goal is to minimize the makespan, i.e., the maximum completion time of all jobs. Though in a previous work [22], Tan and Zhang have designed a LP-based algorithm for the fractional case and showed its optimality by numerical calculation, it is still left as an open question whether the algorithm is optimal in the theoretical sense. This paper first provides a slightly different linear programming formulation to acquire the LP-based algorithm. Then a new lower bound for the problem is constructed by another linear programming formulation. The optimality of the LP-based algorithm is confirmed by duality theory of linear programming.

The rest of the paper is organized as follows. In Section 2, we give the LP-based algorithm and prove its competitive ratio. In Section 3, a new lower bound is constructed to show the optimality of the algorithm.

2. The LP-based algorithm

The idea in [22] for designing an online algorithm for the fractional case is very simple. It just assigns jobs with the same processing set to the machines in the same way. The proportion of a job assigned to a machine only depends on its processing set and the specific machine, regardless of the processing time of itself. We use the same idea but a slightly different linear programming formulation to determine the proportion values.

For any fixed $m \geq 2$, we construct the following linear programming:

$$\min \ \gamma$$

$$\text{s.t.} \ \sum_{i=1}^{j} x_{ij} = 1, \quad j = 1, \ldots, m, \tag{1}$$

$$(LP(m)) \qquad i x_{ii} + \sum_{j=i+1}^{m} x_{ij} = \gamma, \quad i = 1, \ldots, m, \tag{2}$$

$$x_{ij} \geq x_{i,j+1}, \quad j = i, \ldots, m-1, \ i = 1, \ldots, m, \tag{3}$$

$$x_{ij} \geq 0, \quad j = i, \ldots, m, \ i = 1, \ldots, m. \tag{4}$$

In fact, we use the variable x_{ij} to determine the proportion of a job with processing set $\mathcal{M}_j$ to the machine M_i, $i = 1, 2, \ldots, j$. Constraints (1) and (4) ensure that each job must be completely split between its admissible machines. Constraints (2) and (3) are technical requirements that will be used in the competitive analysis. Note that the above linear programming is almost the same with the one arising in [22] except that we replace inequalities by equality constraints in (2). With such linear programming, our algorithm can be described as follows.

The LP-based algorithm

Step 1. Solve the linear programming $LP(m)$ and let $\{x_{ij}^{(m)}, \ i \leq j, \ i, j = 1, 2, \ldots, m, \gamma^{(m)}\}$ be the optimal solution, where $\gamma^{(m)}$ is the optimal objective value.

Step 2. When a job J_j with the processing set $\mathcal{M}_k$ arrives, assign $x_{ik}^{(m)} p_j$ part of the job to machine M_i for any $i = 1, \ldots, k$.

For the sake of completeness, we also provide a proof for the competitive ratio of the LP-based algorithm (which is seen in [22]). Denote by T_k the total processing time of jobs with the processing set $\mathcal{M}_k$, $k = 1, \ldots, m$, and let C_f^{LP} and C_f^* be the makespan generated by the LP-based algorithm and the optimal offline algorithm, respectively.

Lemma 2.1 ([2]). *For the fractional case, the optimal makespan* $C_f^* = \max_{j=1,\ldots,m} \frac{1}{j} \sum_{k=1}^{j} T_k$.

Lemma 2.2 ([27]). *Let* $\mathbf{1} = (1, 1, \ldots, 1)^T$, $\mathbf{x} = (x_1, x_2, \ldots, x_q)^T$ *be* $q \times 1$ *matrices and* $\mathbf{c} = (c_1, c_2, \ldots, c_q)$ *be a* $1 \times q$ *matrix.* $\mathbf{A} = (a_{ij})_{q\times q}$ *is an invertible matrix, and the ith row vector of* $\mathbf{A}$ *is denoted as* α_i. *If* $\mathbf{c}\mathbf{A}^{-1} \geq \mathbf{0}$, *then* $\mathbf{c}\mathbf{x} \leq (\mathbf{c}\mathbf{A}^{-1}1) \max\{\alpha_1\mathbf{x}, \alpha_2\mathbf{x}, \ldots, \alpha_q\mathbf{x}\}$ *for any* $\mathbf{x}$.

Theorem 2.1. *The LP-based algorithm is feasible for the fractional case and has a competitive ratio of* $\gamma^{(m)}$ *for any* $m \geq 2$.

Proof. The feasibility of the algorithm is due to the constraints (1) in the linear programming. In fact, any job is assigned to the permitted machines by Step 2. In addition, for each job J_j with the processing set $\mathcal{M}_k$, the summation of its fractional parts equals exactly $\sum_{i=1}^{k} x_{ik}^{(m)} p_j = p_j \sum_{i=1}^{k} x_{ik}^{(m)} = p_j$ by (1). Thus each job must be finished.

Now let L_i be the completion time of machine M_i when the algorithm terminates, $i = 1, \ldots, m$. By the algorithm's rule in Step 2, only the jobs with processing set $\mathcal{M}_k$, $k \geq i$ can be assigned to the machine M_i, and the proportion of each job assigned to a machine is independent of its processing time, thus we can get

$$L_i = \sum_{k=i}^{m} x_{ik}^{(m)} T_k, \quad i = 1, \ldots, m. \tag{5}$$

Note that $C_f^{LP} = \max_{i=1}^{m} L_i$. For convenience, let $C_f^{LP} = L_i$ for some i. Then we denote $\mathbf{x}_{(i)} = (T_i, T_{i+1}, \ldots, T_m)^T$ and $\mathbf{c}_{(i)} = (x_{ii}^{(m)}, x_{i,i+1}^{(m)}, \ldots, x_{im}^{(m)})$. Consequently, we have $C_f^{LP} = \mathbf{c}_{(i)}\mathbf{x}_{(i)}$ by (5). Let $\mathbf{A}_{(i)}$ be