Information Processing Letters 139 (2018) 39–43

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

2-Approximation algorithm for a generalization of scheduling on unrelated parallel machines

Yossi Azar [a], Jaya Prakash Champati [b,*], Ben Liang [b]

[a] Blavatnik School of Computer Science, Tel-Aviv University, Israel
[b] Department of Electrical and Computer Engineering, University of Toronto, Canada

ARTICLE INFO

Article history:
Received 23 March 2017
Received in revised form 23 February 2018
Accepted 9 July 2018
Available online xxxx
Communicated by Prudence Wong

Keywords:
Unrelated machines
Makespan
Approximation algorithms
Scheduling
Open cycles

ABSTRACT

In their seminal work [8], Lenstra, Shmoys, and Tardos proposed a 2-approximation algorithm to solve the problem of scheduling jobs on unrelated parallel machines with the objective of minimizing makespan. In contrast to their model, where a job is processed to completion by scheduling it on any one machine, we consider the scenario where each job j requires processing on k_j different machines, independently. For this generalization, we propose a 2-approximation algorithm based on the ρ-relaxed decision procedure [8] and open cycles used in [3,2].

1. Introduction

We consider a system of m parallel machines. At time zero, n independent and non-preemptible jobs are given. Let $M = \{1, 2, \dots, m\}$ and $J = \{1, 2, \dots, n\}$ denote the set of machine indices and job indices, respectively. Each job j requires processing on $k_j \le m$ different machines and the processing of the job can be performed independently on different machines. The processing time required by a job j on machine $i \in M$ is p_{ij}. For each job j and machine $i \in M$, let x_{ij} denote a binary variable such that $x_{ij} = 1$ if job j is assigned to machine i, and $x_{ij} = 0$ otherwise. A schedule is then determined by the set $\{x_{ij} : x_{ij} \in \{0, 1\}, \forall i \in M, \forall j \in J\}$. The schedule is feasible if and only if $\sum_{i \in M} x_{ij} = k_j$ for all $j \in J$.

Given a schedule, the completion time on a machine i is determined by the sum of processing times of jobs assigned to it. The makespan of the jobs, denoted by $C_{\max}$, is the maximum completion time over all machines. Given $\{k_j\}$ and $\{p_{ij}\}$ for all j and i, our objective is to find a feasible schedule that minimizes the makespan. We formulate the problem $\mathcal{P}$ as an ILP below:

$$\begin{aligned}
\text{minimize} \quad & C_{\max} \\
\text{subject to} \quad & \sum_{i \in M} x_{ij} = k_j, \quad \forall j \in J & (1) \\
& \sum_{j \in J} x_{ij} p_{ij} \le C_{\max}, \quad \forall i \in M & (2) \\
& x_{ij} \in \{0, 1\}, \quad \forall i \in M, \forall j \in J. & (3)
\end{aligned}$$

We note that the above formulation is general and can be used for the case where jobs have placement constraints, i.e., a job j can only be processed on a subset of machines. In this case, we assign $p_{ij} = \infty$, for every machine i on which the job j cannot be processed.

Our motivation for studying $\mathcal{P}$ is the following model for data retrieval in a coded memory storage system [10,7]. A data file j is divided into k_j blocks that are encoded into $N_j \ge k_j$ code blocks. Each of the N_j code blocks are stored

* Corresponding author.
E-mail address: jayaprakash.nitw@gmail.com (J.P. Champati).

https://doi.org/10.1016/j.ipl.2018.07.005

on N_j different storage units. A read request for the data file j can be served by retrieving any k_j code blocks. Given m storage units and n data file read requests, the problem of minimizing the total time to retrieve the files from the storage system can be formulated using $\mathcal{P}$.

For the special case where $k_j = 1$ for all $j \in J$, $\mathcal{P}$ is equivalent to the classical problem of minimizing makespan on unrelated parallel machines, denoted by $R||C_{\max}$ [5,6,4,9,8]. Horowitz and Shani [6] provided a fully polynomial time approximation algorithm for any fixed number of unrelated machines. A list scheduling algorithm having $2\sqrt{m}$ approximation ratio was proposed by Davis and Jaffe [4]. Later, Potts [9] proposed a 2-approximation algorithm by solving a relaxed linear program and doing enumeration for the non-integral part of the solution. However, due to the enumeration step, Potts' algorithm has $O(m^{m-1})$ time complexity.

Lenstra, Shmoys, and Tardos (LST) [8] extended the solution approach of Potts by providing a polynomial time algorithm for rounding the fractional solution of the linear program. The LST algorithm is based on finding a *ρ-relaxed decision procedure* as follows. Given P, an instance of $R||C_{\max}$, and a deadline T, the ρ-relaxed decision procedure outputs 'no' if there is no schedule with makespan at most T for an integer relaxation of P, else it outputs a schedule with makespan at most ρT for P. The LST algorithm finds a 2-relaxed decision procedure and uses a simple binary search to obtain a 2-approximation solution to $R||C_{\max}$.

We note that the LST algorithm cannot be directly extended to solve $\mathcal{P}$. To see this, consider the underlying feasibility problem for finding a ρ-relaxed decision procedure for $\mathcal{P}$, for a given deadline T. It consists of the constraints in (1), constraints in (2) with $C_{\max}$ replaced by T, and the relaxed constraints $0 \le x_{ij} \le 1$, for all i and for all j. Let r be the number of variables in this feasibility problem, then the number of constraints are $2r+m+n$. This is in contrast to the number of constraints $r+m+n$ present in the corresponding feasibility problem for the classical unrelated parallel machines problem [8]. Therefore, the counting argument used in [8] to claim that only m jobs will have non-integral x_{ij} values is not applicable to the feasibility problem at hand.

In this work, we present a 2-approximation solution to $\mathcal{P}$. Our solution approach closely follows [8] with an exception that we use open cycles in a bipartite graph [3,2] to round the solution of the feasibility problem for $\mathcal{P}$. For ease of exposition, in Section 2 we first present our solution to a special case of $\mathcal{P}$, where the processing time of any job is the same on any eligible machine, i.e., the case of identical machines with assignment restrictions. This is then extended in Section 3 to the case of related machines with assignment restrictions. Finally, in Section 4 we detail the additional steps required for solving $\mathcal{P}$.

2. Identical machines with assignment restrictions

Let $\mathcal{P}_I$ denote the special case of $\mathcal{P}$, where the processing time of a job j on any machine is either p_j or ∞. Let M_j denote the set of eligible machines of job j on which its processing time is p_j. Similarly, let J_i denote the set of jobs which have finite processing time on machine i. In the following we present a 2-relaxed decision procedure for $\mathcal{P}_I$.

2.1. 2-relaxed decision procedure

The 2-relaxed decision procedure for $\mathcal{P}_I$ is based on the following feasibility problem.

$$\begin{aligned}
&\sum_{i\in M_j} x_{ij} = k_j, \quad \forall j \in J \\
&\sum_{j\in J_i} x_{ij} p_j \le T, \quad \forall i \in M \\
&0 \le x_{ij} \le 1, \quad \forall i \in M, \forall j \in J \\
&x_{ij} = 0, \quad \forall i \notin M_j, \forall j \in J,
\end{aligned} \tag{4}$$

for some $T \ge \max_j p_j$. If (4) is not feasible, then there is no schedule for $\mathcal{P}_I$ with makespan at most T. If (4) is feasible, then we round the fractional solution using open cycles followed by a simple matching in a forest graph. We show that the resulting schedule has makespan at most $2T$ for $\mathcal{P}_I$, thus establishing a 2-relaxed decision procedure for $\mathcal{P}_I$. In the following we solve (4) by reducing it to a maximum flow problem.

2.1.1. Maximum flow problem

Consider the bipartite graph $G = \{J \cup M, E\}$, where $E = \{(j,i) : j \in J, i \in M_j\}$. Using G we construct a flow network $\mathcal{N}$ as follows:

- Introduce a source and add directed edges from the source to all vertices in J. Assign capacity $k_j p_j$ to the edge from the source to vertex j.
- If $(j,i) \in E$, then direct the edge from j to i and assign capacity p_j to the edge.
- Introduce a sink and add directed edges from all vertices in M to the sink. Assign capacity T to all these edges.

It is easy to establish that solving the maximum flow problem in $\mathcal{N}$ results in a feasible solution for (4). This is stated in the following proposition.

Proposition 1. *For any given T, (4) is feasible if and only if there exists a maximum flow f with value $\sum_{j=1}^{n} k_j p_j$ in $\mathcal{N}$. Further, if such flow f exists, then the schedule $\{\bar{x}_{ij} = f(j,i)/p_j$, for all $(j,i) \in E\}$ is a solution for (4).*

Assuming $n \ge m$, the maximum flow problem in $\mathcal{N}$ can be solved efficiently by a bipartite preflow-push algorithm with run time $O(m^3 n)$ [1]. Next, we assume that for a given T a maximum flow f with value $\sum_{j=1}^{n} k_j p_j$ exists in $\mathcal{N}$. We round the non-integral part of $\{\bar{x}_{ij}\}$ using open cycles and matching in a forest graph.

2.1.2. Open cycles

We construct an undirected bipartite graph $\bar{G} = \{\bar{J} \cup \bar{M}, \bar{E}\}$, such that $\bar{J} \subseteq J$, $\bar{M} \subseteq M$, and (j,i) is in $\bar{E}$ if and only if $0 < f(j,i) < p_j$ for all $j \in J$ and $i \in M$. A job vertex