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*
[b]*Department of Electrical and Computer Engineering, University of Toronto*

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 $M = \{1, 2, \ldots, m\}$ and $J = \{1, 2, \ldots, n\}$ denote the set of machine indices and job indices, respectively. Each job j requires processing on $k_j \leq m$ 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:

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

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 $N_j \geq k_j$ code blocks. Each of the N_j code blocks are stored 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 comprises of the constraints in (1), constraints in (2) with $C_{\max}$ replaced by T, and the relaxed constraints $0 \leq x_{ij} \leq 1$, 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.

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

for some $T \geq \max_j p_j$. 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 $n \geq m$, 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 $\bar{j}$ that is in $\bar{G}$ should have at least two non-saturated edges under f. To see this, note that since $\bar{j}$ is in $\bar{G}$, we should have $0 < f(\bar{j}, i_1) < p_{\bar{j}}$ for some $i_1 \in M_{\bar{j}}$. Now, there should be at least one edge $0 < f(\bar{j}, i_2) < p_{\bar{j}}$ for some $i_2 \in M_{\bar{j}}$ to satisfy $\sum_{i \in M} f(\bar{j}, i) = k_{\bar{j}} p_{\bar{j}}$, the total flow out of $\bar{j}$.

If there are cycles in $\bar{G}$, we use the following procedure to open the cycles. Let $C = \{(j_1, i_1),(i_1, j_2), (j_2, i_2) \ldots,(j_l, i_l),(i_l, j_1)\}$ be an undirected cycle in $\bar{G}$. Since $\bar{G}$ is a bipartite graph, the cycle has an even number of edges. For each edge in C, modify the flow f in $\mathcal{N}$ as follows. For some small $\epsilon > 0$, which will be determined later, increase the flow on edge (j_1, i_1) by ϵ, reduce the flow on edge (j_2, i_1) by ϵ and repeat this procedure for consecutive edges in the cycle. This is demonstrated in an example graph $\bar{G}$ in Figure 1. Note that it only contains one cycle, given by $\{(1, 1), (1, 2), (2, 2), (2, 3), (3, 3), (3, 1)\}$ and the flow on alternative edges are increased/decreased by ϵ. It is easy to see that the modified flows on the edges of the cycle do not affect the value of the flow f.

We choose the minimum ϵ value such that the modified flow on one of the edges in cycle C either is reduced to zero or saturates the edge. Such ϵ value can be derived by using the fact that the flow on any edge is non-negative and cannot exceed the edge capacity. We have

$$\begin{aligned} f(j_1, i_1) + \epsilon &\leq p_{j_1} \\ f(j_2, i_1) - \epsilon &\geq 0 \\ &\vdots \\ f(j_l, i_l) + \epsilon &\leq p_{j_l} \\ f(j_1, i_l) - \epsilon &\geq 0. \end{aligned}$$

From the above inequalities, we obtain

$$\epsilon = \min_{(j_q, i_q) \in C} \min\{f(j_{q+1}, i_q), p_{j_q} - f(j_q, i_q)\}.$$

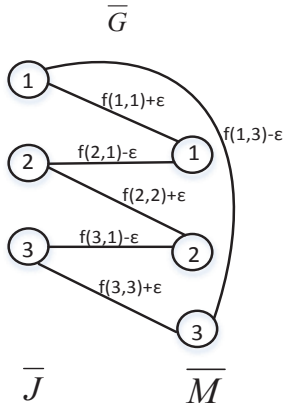

Figure 1: An example of an undirected graph $\bar{G}$ with one cycle

As a numerical illustration for the example given in Figure 1, consider the case that the processing times of tasks $1, 2$, and 3 are given by $2, 4$, and 5 units, respectively. Further suppose the solution to the maximum-flow problem resulted in the following flow values: $f(1,1) = 1.6, f(2,1) = 2.8, f(2,2) = 1.2, f(3,2) = 2.5, f(3,3) = 2.5$, and $f(1,3) = 0.4$. This implies, $\bar{x}_{11} = 0.8, \bar{x}_{12} = 0.7, \bar{x}_{22} = 0.3, \bar{x}_{23} = 0.5, \bar{x}_{33} = 0.5$ and $\bar{x}_{31} = 0.2$. Using the above values we compute $\epsilon = 0.4$, which is the minimum value that reduces the flow on edge $(1,3)$ to zero (and also saturates edge $(1,1)$). The new flow values are $f(1,1) = 2, f(2,1) = 2.4, f(2,2) = 1.6, f(3,2) = 2.1, f(3,3) = 2.9$, and $f(1,3) = 0$.

Now, the cycle C is opened by deleting an edge from $\bar{G}$ where the flow either is reduced to zero or saturates the edge. We repeat this procedure of opening cycles until there is no cycle in $\bar{G}$. We denote $\bar{G}$ with no cycles by $\bar{G}_0$. Note that $\bar{G}_0$ is a forest graph. Since opening each cycle results in deleting an edge, it takes polynomial time to open all cycles.

After opening all cycles in $\bar{G}$, the modified flow f in $\mathcal{N}$ still has the value $\sum_{j=1}^{n} k_j p_j$. Using this modified flow f, we recompute the schedule $\{\bar{x}_{ij}\}$. Note that the recomputed $\{\bar{x}_{ij}\}$ is a feasible solution for (4). Now, to obtain a feasible solution $\{\hat{x}_{ij}\}$ for $\mathcal{P}_I$, we proceed as follows. Initialize all $\hat{x}_{ij}$ to 0. Under the modified flow f, if an edge from a job vertex to a machine vertex is saturated, then assign the job to that vertex, i.e., for all $(j,i) \in E$, if $f(j,i) = p_j$, then $\hat{x}_{ij} = 1$. According to the above assignment, we can infer that $\hat{x}_{ij} = 1$ if and only if $\bar{x}_{ij} = 1$.

Lemma 1. *Let $\bar{M}_{0,j}$ denote the set of machine vertices adjacent to the job vertex j in the forest graph $\bar{G}_0$, then the degree of a job vertex j in $\bar{G}_0$ satisfies the following inequality:*

$$|\bar{M}_{0,j}| \geq k_j - \sum_{i \in M_j \setminus \bar{M}_{0,j}} \hat{x}_{ij} + 1.$$

Proof. By noting the initial construction of $\bar{G}$ and the procedure of opening cycles, the job vertex j in forest graph $\bar{G}_0$ should have $0 < f(j,i) < p_j$, for all $i \in \bar{M}_{0,j}$. Since f is a maximum flow, we have

$$\begin{aligned}
&\sum_{i \in M_j} f(j,i) = k_j p_j \\
\Rightarrow &\sum_{i \in \bar{M}_{0,j}} \frac{f(j,i)}{p_j} + \sum_{i \in M_j \setminus \bar{M}_{0,j}} \frac{f(j,i)}{p_j} = k_j \\
\Rightarrow &\sum_{i \in \bar{M}_{0,j}} \frac{f(j,i)}{p_j} + \sum_{i \in M_j \setminus \bar{M}_{0,j}} \hat{x}_{ij} = k_j \qquad (5) \\
\Rightarrow &|\bar{M}_{0,j}| \geq k_j - \sum_{i \in M_j \setminus \bar{M}_{0,j}} \hat{x}_{ij} + 1.
\end{aligned}$$

In the third equality above we have used the fact that $\hat{x}_{ij} = \frac{f(j,i)}{p_j}$, for all $i \in M_j \setminus \bar{M}_{0,j}$, because in this case $f(j,i)$ is either equal to p_j or zero. In the last inequality above we have used $0 < f(j,i) < p_j$, for all $i \in \bar{M}_{0,j}$. □

The following corollary can be deduced from Lemma 1.

Corollary 1. *The degree of a job vertex j in the forest graph $\bar{G}_0$ is at least 2.*

Proof. To prove the corollary it is sufficient to show that

$$k_j - \sum_{i \in M_j \setminus \bar{M}_{0,j}} \hat{x}_{ij} \geq 1.$$

Noting that $f(j,i) > 0$, for all $i \in \bar{M}_{0,j}$ and all $\hat{x}_{ij}$ are either 0 or 1, the above inequality follows from equation (5). □

Next, we describe how to match the remaining jobs to machines in the forest graph $\bar{G}_0$.

2.1.3. Matching in Forest Graph $\bar{G}_0$

Since all the job vertices in $\bar{G}_0$ have degree at least 2, only machine vertices can be leafs in $\bar{G}_0$. Now, consider each connected component, in the forest graph $\bar{G}_0$. Use any job vertex as a root in the tree. Since any job vertex has at least one child, we may assign each job vertex to one of its children machine vertices, delete those machine vertices, and set the corresponding $\hat{x}_{ij}$ to 1. If for any job j, $\sum_{i \in M_j} \hat{x}_{ij} = k_j$, then delete the job vertex as well. The above step is repeated till all job vertices are deleted.

Theorem 1. *The schedule $\{\hat{x}_{ij}\}$, obtained after matching in the forest graph $\bar{G}_0$, is feasible for $\mathcal{P}_I$ and has a makespan at most $2T$.*

Proof. Note that all $\hat{x}_{ij}$ are either 0 or 1. Before performing the above matching in the forest, any job vertex j not in the forest graph $\bar{G}_0$ should have $\sum_{i \in M_j} \hat{x}_{ij} = k_j$. Otherwise, for some $i \in M_j$, we would have $0 < f(j,i) < p_j$, and j would have been in $\bar{G}_0$. From Lemma 1, the degree of a job j in $\bar{G}_0$ is at least $(k_j - \sum_{i \in M_j \setminus \bar{M}_{0,j}} \hat{x}_{ij} + 1)$, which implies that it has at least $(k_j - \sum_{i \in M_j \setminus \bar{M}_{0,j}} \hat{x}_{ij})$ children in the tree that contains it. Therefore, in the matching procedure job j will be matched to $(k_j - \sum_{i \in M_j \setminus \bar{M}_{0,j}} \hat{x}_{ij})$ children in the tree, i.e.

$$\sum_{i \in \bar{M}_{0,j}} \hat{x}_{ij} = k_j - \sum_{i \in M_j \setminus \bar{M}_{0,j}} \hat{x}_{ij}.$$

Thus, by the end of the matching procedure the constraints $\sum_{i\in M_j} \hat{x}_{ij} = k_j$, for all $j \in J$, are satisfied.

Recall that the schedule $\{\bar{x}_{ij}\}$ is computed based on the modified flow f obtained after opening all cycles in $\bar{G}$. Also, matching in the forest graph $\bar{G}_0$ results in each machine being assigned to at most one job. This implies that for each machine i, the vectors $(\bar{x}_{ij}, j = 1, 2, \ldots, n)$ and $(\hat{x}_{ij}, j = 1, 2, \ldots, n)$ differ in at most one value. Therefore,

$$\begin{aligned}\sum_{j\in J_i} \hat{x}_{ij} p_j &\le \sum_{j\in J_i} \bar{x}_{ij} p_j + \max_j p_j, \quad \forall i \in M \\ &\le T + T = 2T, \quad \forall i \in M.\end{aligned}$$

The second inequality above follows from the fact that $\{\bar{x}_{ij}\}$ is a feasible solution for (4) and the value of T is chosen to be at least $\max_j p_j$. □

2.2. Binary Search

Given the 2-relaxed decision procedure for $\mathcal{P}_I$, we use the following binary search algorithm to obtain a 2-approximation solution. Let

$$\begin{aligned}a &= \max\{\frac{1}{m}\sum_{j=1}^{n} k_j p_j, \max_j p_j\} \\ b &= \sum_{j=1}^{n} k_j p_j.\end{aligned}$$

Clearly, a and b are lower and upper bounds for the optimal makespan. Set $T = \lfloor \frac{a+b}{2} \rfloor$. For this T if (4) is not feasible, then output 'no', and set $a = T + 1$; else set $b = T$. Again, set $T = \lfloor \frac{a+b}{2} \rfloor$ and repeat the above procedure until $a = b$. The output is a 2-approximation solution for $\mathcal{P}_I$ [Lemma 1, [8]].

3. Related Machines with Assignment Restrictions

For the case of related machines with assignment restrictions, the processing time of job j is p_j/v_i, for all $i \in M_j$, where v_i is the speed factor of machine i. The 2-relaxed decision procedure for this case is the same as that of the case of identical machines except the following changes to be made in constructing graph $\mathcal{N}$:

- We assign the capacity of an edge from machine vertex i to the sink to be $v_i T$.
- For any edge from job j to machine i, if $p_j/v_i > T$, delete the edge.

Note: If we do not delete the edges (j, i) where $p_j/v_i > T$, then under the maximum flow solution f, we may have $f(j, i) > 0$ for those edges. However, $f(j, i)$ will be zero for those edges after rounding. Therefore, one may improve the run time of the algorithm by simply deleting those edges.

4. Unrelated Machines

The solution approach for finding a 2-approximation solution for $\mathcal{P}$ is the same as that of $\mathcal{P}_I$ except the open cycles procedure, which is more involved for $\mathcal{P}$. In the following we present the 2-relaxed decision procedure for $\mathcal{P}$.

Given T, the feasibility problem for $\mathcal{P}$ is presented below.

$$\begin{aligned}&\sum_{i\in M} x_{ij} = k_j, \quad \forall j \in J \\ &\sum_{j\in J} x_{ij} p_{ij} \le T, \quad \forall i \in M \\ &x_{ij} = 0, \quad \forall i \in M, \forall j \in J \text{ such that } p_{ij} > T \\ &0 \le x_{ij} \le 1, \quad \forall i \in M, \forall j \in J.\end{aligned} \tag{6}$$

The third constraint above is crucial for the proof of our 2-approximation result. If we do not impose this constraint, a feasible solution may have $0 < x_{ij} < 1$ for some i and j such that $p_{ij} > T$. In contrast to (4), (6) cannot be reduced to a maximum flow problem. However, given a solution $\{\bar{x}_{ij}\}$ to (6) we obtain flow values $f(j, i) = \bar{x}_{ij} p_{ij}, \forall i \in M, \forall j \in J$. We then construct the undirected bipartite graph $\bar{G} = \{\bar{J} \cup \bar{M}, \bar{E}\}$, such that (j, i) is in $\bar{E}$ if and only if $0 < f(j, i) < p_{ij}$.

Let $C = \{(j_1, i_1),(i_1, j_2),(j_2, i_2) \ldots,(j_l, i_l),(i_l, j_1)\}$ be an undirected cycle in $\bar{G}$. We modify the flow f as follows:

$$\begin{aligned}f(j_1, i_1) &= f(j_1, i_1) + \epsilon \\ f(j_2, i_1) &= f(j_2, i_1) - \epsilon \\ f(j_2, i_2) &= f(j_2, i_2) + \epsilon \cdot \frac{p_{i_2 j_2}}{p_{i_1 j_2}} \\ f(j_3, i_2) &= f(j_3, i_2) - \epsilon \cdot \frac{p_{i_2 j_2}}{p_{i_1 j_2}} \\ f(j_3, i_3) &= f(j_2, i_2) + \epsilon \cdot \frac{p_{i_2 j_2}}{p_{i_1 j_2}} \cdot \frac{p_{i_3 j_3}}{p_{i_2 j_3}} \\ &\vdots \\ f(j_l, i_l) &= f(j_l, i_l) + \epsilon \cdot \frac{p_{i_2 j_2}}{p_{i_1 j_2}} \cdot \frac{p_{i_3 j_3}}{p_{i_2 j_3}} \cdots \frac{p_{i_l j_l}}{p_{i_{l-1} j_l}} \\ f(j_1, i_l) &= f(j_1, i_l) - \epsilon \cdot \frac{p_{i_l j_1}}{p_{i_1 j_1}}\end{aligned}$$

In the above modification, the flow entering any machine vertex $i \in \bar{M}$ is unchanged except for machine vertex i_l. The flow entering i_l is changed by $-\epsilon \cdot \frac{p_{i_l j_1}}{p_{i_1 j_1}} \cdot (1 - \Pi)$, where

$$\Pi = \frac{p_{i_1 j_1}}{p_{i_l j_1}} \cdot \frac{p_{i_2 j_2}}{p_{i_1 j_2}} \cdot \frac{p_{i_3 j_3}}{p_{i_2 j_3}} \cdots \frac{p_{i_l j_l}}{p_{i_{l-1} j_l}}$$

If $\Pi \le 1$, then the flow entering i_l cannot increase. By choosing sufficiently small $\epsilon > 0$, we can compute a new feasible solution for (6) by using the modified flow values. If $\Pi > 1$, then we modify the flows in the reverse cycle given by $\{(j_1, i_l), (i_l, j_l), \ldots, (j_2, i_1), (i_1, j_1)\}$. We start with reducing the flow $f(j_1, i_l)$ by ϵ and increasing the flow $f(i_l, j_l)$ by ϵ and repeat this procedure for consecutive edges in the reverse cycle. In this case, it can be verified that only the flow entering machine vertex i_1 will be changed. It will be changed by

$-\epsilon \cdot \frac{p_{i_1 j_1}}{p_{i_l j_1}} \cdot (1 - 1/\Pi)$. Since $\frac{1}{\Pi} < 1$, the flow entering machine vertex i_1 will be reduced. Again, choosing sufficiently small $\epsilon > 0$, we can compute a new feasible solution for (6) by using the modified flow values.

Again, choose the maximum ϵ value such that the modified flow value $f(j, i)$ on some edge either equals p_{ij} or is reduced to 0. We open the cycle by deleting that edge. After opening all cycles, we initialize all $\hat{x}_{ij}$ values to 0, and assign $\hat{x}_{ij} = 1$ if and only if $f(j, i) = p_{ij}$. Matching in the forest graph $\bar{G}_0$ is performed exactly the same as before, leading to the solution $\{\hat{x}_{ij}\}$.

Theorem 2. *The schedule $\{\hat{x}_{ij}\}$ is feasible for $\mathcal{P}$ and has a makespan at most $2T$.*

Proof. The proof is similar to the proof of Theorem 1, where the approximation bound holds using also the fact that $p_{ij} \leq T$ for $\hat{x}_{ij} > 0$, for all i, j. □

Again, given the 2-relaxed decision procedure for $\mathcal{P}$, we can find a 2-approximation solution by binary search with lower and upper bounds for T chosen as follows:

$$a = \max\{\frac{1}{m}\sum_{j=1}^{n} k_j \min_i p_{ij}, \max_j\{\min_i p_{ij}\}\}$$

$$b = \sum_{j=1}^{n} k_j \max_i p_{ij}.$$

Acknowledgement

We would like to thank Allan Borodin for a helpful discussion that led to the present collaboration. This work has been supported in part by the Natural Sciences and Engineering Research Council of Canada, the Israel Science Foundation (grant No. 1506/16), and the ICRC Blavatnik Fund.

References

[1] Ahuja, R. K., Magnanti, T. L., Orlin, J. B., 1993. Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Englewood Cliffs, NJ.

[2] Andelman, N., Mansour, Y., 2004. Auctions with budget constraints. In: Algorithm Theory - SWAT 2004. Vol. 3111 of Lecture Notes in Computer Science. Sringer, Berlin, Heidelberg.

[3] Azar, Y., Epstein, L., Richter, Y., Woeginger, G. J., 2004. All-norm approximation algorithms. J. Algorithms 52 (2), 120–133.

[4] Davis, E., Jaffe, J. M., 1981. Algorithms for scheduling tasks on unrelated processors. J. ACM 28 (4), 721–736.

[5] Graham, R. L., Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of discrete mathematics 5 (2), 287–326.

[6] Horowitz, E., Sahni, S., 1976. Exact and approximate algorithms for scheduling nonidentical processors. JACM 23 (2), 317–327.

[7] Joshi, G., Liu, Y., Soljanin, E., 2014. On the delay-storage trade-off in content download from coded distributed storage systems. IEEE Journal on Selected Areas in Communications 32 (5), 989–997.

[8] Lenstra, J. K., Shmoys, D. B., Tardos, E., 1990. Approximation algorithms for scheduling unrelated parallel machines. Math. Program. 46 (3), 259–271.

[9] Potts, C. N., 1985. Analysis of a linear programming heuristic for scheduling unrelated parallel machines. Discrete Applied Mathematics 10 (2), 155–164.

[10] Rashmi, K., Shah, N. B., Gu, D., Kuang, H., Borthakur, D., Ramchandran, K., 2014. A"hitchhiker's" guide to fast and efficient data reconstruction in erasure-coded data centers. In: proceedings ACM SIGCOMM. pp. 331–342.