An Improved Approximation Algorithm for Scheduling under Arborescence Precedence Constraints

Nguyễn Kim Thắng
IBISC, Univ Evry, University Paris Saclay
Evry, France
kimthang.nguyen@univ-evry.fr

Abstract

We consider a scheduling problem on unrelated machines with precedence constraints. There are m unrelated machines and n jobs and every job has to be processed non-preemptively in some machine. Moreover, jobs have precedence constraints; specifically, a precedence constraint $j \prec j'$ requires that job j' can only be started whenever job j has been completed. The objective is to minimize the total completion time.

The problem has been widely studied in more restricted machine environments such as identical or related machines. However, for unrelated machines, much less is known. In the paper, we study the problem where the precedence constraints form a forest of arborescences. We present a $O\big((\log n)^2/(\log\log n)^3\big)$-approximation algorithm — that improves the best-known guarantee of $O\big((\log n)^2/\log\log n\big)$ due to Kumar et al. [12] a decade ago. The analysis relies on a dual-fitting method in analyzing the Lagrangian function of non-convex programs.

2012 ACM Subject Classification Theory of Computation → Approximation Algorithms Analysis

Keywords and phrases Scheduling, Precedence Constraints, Lagrangian Duality

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.73

Funding Research supported by the ANR project OATA n° ANR-15-CE40-0015-01.

1 Introduction

In this paper, we consider a classic scheduling problem on unrelated machines with precedence constraints. There are m unrelated machines and n jobs. Each job j has a processing time p_{ij} if it is processed on machine i. A job must be executed non-preemptively in some machine i (i.e., in an interval of length p_{ij} in machine i). Jobs have *precedence constraints* which are represented by a partial order $\prec$. Specifically, a dependence constraint $j \prec j'$ requires that job j' can only be started whenever job j has been completed. Hence, we need to assign jobs to machines and process them in some order consistent with the precedence constraints. The objective is to minimize the total completion time, i.e., $\sum_j C_j$ where C_j is the completion time of job j. In the standard three field notion, the problem is denoted as $R|prec|\sum_j C_j$.

The weighted version of this problem is a similar one where additionally jobs have weights and the objective is to minimize the total weighted completion time, denoted as $R|prec|\sum_j w_jC_j$. Little is known for both problems $R|prec|\sum_j C_j$ and $R|prec|\sum_j w_jC_j$ in the unrelated machine environments. However, the problem has been widely considered in more restricted machine environments such as identical parallel machines or related parallel machines. The problem $P|prec|\sum_j w_jC_j$ corresponding to the setting of identical machines ($p_{ij} = p_j \ \forall i$) has been extensively studied. Many algorithms and techniques have been designed for the latter over decades [13, 9, 4, 16, 6, 10, 5, 2, 19, 18]. The problem $P|prec|\sum_j w_jC_j$ has been revived with significant progresses recently. Li [15] provided a $(2 + 2\ln 2 + \epsilon)$-approximation by a subtle rounding based on a time-index LP. Later on, Garg et al. [8] gave a $(2 + \epsilon)$-approximation algorithm when the number of machines is a constant.

45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020).
Editors: Javier Esparza and Daniel Král'; Article No. 73; pp. 73:1–73:12
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Their result relies on a lift and project approach developed by Levey and Rothvoss [14] and Garg [7]. This approximation ratio matches to the lower bound of 2 proved by Bansal and Khot [2] assuming a variant of the Unique Game Conjecture.

In the more general setting of related machines (in which $p_{ij} = p_j/s_i$ where s_i is the speed of machine i), the corresponding problem $Q|prec|\sum_j w_jC_j$ does not admit any constant approximation assuming a (stronger) variant of the Unique Game Conjecture [3]. On the positive side, Chudak and Shmoys [6] showed an $O(\log m)$-approximation algorithm. This approximation ratio remained the best known upper bound until recently Li [15] gave an improved $O(\log m/\log\log m)$-approximation algorithm.

Despite progress in more restricted machine environments, there is still a large gap in the understanding of the problems $R|prec|\sum_j C_j$ and $R|prec|\sum_j w_jC_j$. When the precedence constraints are a collection of node-disjoint chains, the problems become the job shop scheduling problems [17, 11] — again a classic problem with a long history. A particular interesting case of the problem $R|prec|\sum_j w_jC_j$ is the setting where the precedence constraints form a forest (i.e., the underlying undirected graph of the constraints is a forest), denoted as $R|forest|\sum_j w_jC_j$. This problem is motivated by several applications such as evaluating large expression-trees and tree-shaped parallel processes. Kumar et al. [12] gave an $O\big(\log^3 n/(\log\log n)^2\big)$-approximation algorithm for $R|forest|\sum_j w_jC_j$. When the forests are out-trees or in-trees, the approximation ratio can be improved to $O\big(\log^2 n/\log\log n\big)$. It has remained the best-known result for a decade until now in both unweighted job and weighted job settings.

1.1 Our contribution and approach

We study the special setting of $R|prec|\sum_j C_j$ where the precedence constraints form a forest of *arborescences/out-trees*. (An arborescence/out-tree is a directed acyclic graph where the in-degree of every vertex is at most 1.) We denote the problem by $R|arborescences|\sum_j C_j$. The main result of the paper is the following.

Theorem. *There exists an $O\big((\log n)^2/(\log\log n)^3\big)$-approximation algorithm for the problem $R|arborescences|\sum_j C_j$ where n is the number of jobs.*

Approach. In our approach, instead of directly dealing with the problem $R|arborescences|\sum_j C_j$, we consider first a related problem in the speed-scaling model. In the latter, machines can execute jobs with different speeds and that consumes energy. The objective of the new problem is to minimize the total completion time plus energy (under the same precedence constraints). Intuitively, this problem can be considered as a smooth and relaxed version of the original problem where the energy plays the role of a regularizer. More specifically, in the original problem, at any time every machine either executes some job or do not execute any job; these cases correspond to the speed of 1 or 0, respectively. In the related problem, one is allowed to choose an arbitrary (non-negative) speed. Moreover, the role of the energy function is to prevent the speed from being chosen too high or too low — both situations would lead to a large approximation ratio when converting a solution of the related speed-scaling problem to that of the original one. (Low speed results in a large total completion time whereas high speed yields a large factor in order to convert that speed to 0-1 speed.) Finally, given a solution for the problem of minimizing the total completion time plus energy, we show that one can transform that solution to a feasible schedule of the problem $R|arborescences|\sum_j C_j$ with some reasonable loss factor depending on the energy function.

In the paper, we choose the energy function of the form z^α where $\alpha = \Theta(\log n / \log\log n)$ in order to minime the loss.

Following the strategy described above, we focus on the design of an algorithm for the problem of minimizing the total completion time plus energy and analyze its performance by using tools in mathematical programming. In previous works on scheduling under precedence constraints, the most successful techniques are LP-based roundings [15, 12] or lift-and-project methods [7, 8]. In this paper, we take a different approach that relies non-convex mixed-integer formulations and weak duality presented in [20]. With this approach, we can construct a formulation that is convenient for the design and analysis of our algorithm since the formulation does not need to be either linear or convex. Moreover, one can work directly with integral variables without relaxing them, so avoiding serious integrality gap issue. Specifically, we consider a non-convex formulation for the problem of minimizing the total completion time plus energy and analyze the corresponding Lagrangian function, using the dual-fitting method, in order to bound the dual. The approach allows us to prove an approximation guarantee. That algorithm subsequently is used to derive the improved $O\big((\log n)^2/(\log\log n)^3\big)$-approximation algorithm for the problem $R|\text{arborescences}|\sum_j C_j$.

2 Preliminaries

Given a set of n jobs, the precedence contraints $\prec$ can be represented succinctly by a directed dependence graph. In this graph, there are n vertices, each represents a job, and there is an arc (j, j') if $j \prec j'$. Note that if in the graph there is a directed path $j_1, j_2, \ldots, j_k$ and an arc (j_1, j_k) then one can simply remove the arc (j_1, j_k) in the graph while always maintaining the job dependences. In the paper, we consider dependence graph as a collection of *arborescences*. An *arborescence* is a directed acyclic graph where the in-degree of every vertex is at most 1. The problem, as defined earlier, is to schedule jobs on unrelated machines in order to minimize the total completion time under the arborescence constraints, i.e., $R|arborescences|\sum_j C_j$.

Total Completion Time plus Energy. In order to design algorithm for the problem $R|arborescences|\sum_j C_j$, we study the following related problem in the speed-scaling model. In the problem, there are m unrelated machines and n jobs. An algorithm can choose speeds $s_i(t)$ for every machine i at every time t in order to execute jobs. That incurs the total energy of $\int_0^\infty s_i(t)^\alpha dt$ where $\alpha \geq 2$ is a fixed parameter. Each job j has a volume p_{ij} if it is executed on machine i. A job can be processed preemptively in a machine but *without migration*, i.e., every job must be assigned to some single machine. A job j assigned to some machine i is completed at time C_j if the total volume executed by machine i on this job up to time C_j is equal to p_{ij}. Moreover, jobs have precedence constraints $\prec$ which are represented by a collections of arborescences. A job j cannot be executed before the completion of every job j' where $j' \prec j$. In this problem, an algorithm needs to assign jobs to machines, decide the running speeds and execute jobs in some order consistent with the precedence constraints. The objective is to minimize the total completion time plus energy, which is $\sum_j C_j + \sum_i \int_0^\infty s_i(t)^\alpha dt$. In the paper, we first design an algorithm for this problem and subsequently derive an algorithm for the problem $R|arborescences|\sum_j C_j$.

Weak Duality. A property of mathematical programming, which holds for non-convex optimization and is crucial in our analysis, is the weak duality, stated as follows. For completeness, we incorporate also its (short) proof.

▶ **Lemma 1** (Weak duality). *Consider a possibly non-convex optimization problem* $p^* := \min_x f_0(x) \quad : \quad f_i(x) \leq 0, \quad i = 1, \ldots, m$ *where* $f_i : \mathbb{R}^n \to \mathbb{R}$ *for* $0 \leq i \leq m$. *Let* $\mathcal{X}$ *be the feasible set of* x. *Let* $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ *be the Lagrangian function* $L(x, \lambda) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x)$. *Define* $d^* = \max_{\lambda \geq 0} \min_{x \in \mathcal{X}} L(x, \lambda)$ *where* $\lambda \geq 0$ *means* $\lambda \in \mathbb{R}^m_+$. *Then* $p^* \geq d^*$.

Proof. We observe that, for every feasible $x \in \mathcal{X}$, and every $\lambda \geq 0$, $f_0(x)$ is bounded below by $L(x, \lambda)$:

$$\forall x \in \mathcal{X},\ \forall \lambda \geq 0 : \ f_0(x) \geq L(x, \lambda)$$

Define a function $g : \mathbb{R}^m \to \mathbb{R}$ such that

$$g(\lambda) := \min_z L(z, \lambda) = \min_z f_0(z) + \sum_{i=1}^m \lambda_i f_i(z)$$

As g is defined as a point-wise minimum, it is a concave function.

We have, for any x and λ, $L(x, \lambda) \geq g(\lambda)$. Combining with the previous inequality, we get

$$\forall x \in \mathcal{X} : \ f_0(x) \geq g(\lambda)$$

Taking the minimum over x, we obtain $\forall \lambda \geq 0 : \ p^* \geq g(\lambda)$. Therefore,

$$p^* \geq \max_{\lambda \geq 0} g(\lambda) = d^*.$$

◀

Notations. Given a collection of arborescences G, for every job j, define $\texttt{prev}(j)$ to be the job j' if there exists an arc (j', j) in the graph G; and $\texttt{prev}(j) = \emptyset$ if the in-degree of j is 0. Note that as in G the in-degree of every vertex is at most one, $\texttt{prev}(j)$ is well-defined. Intuitively, $\texttt{prev}(j)$ is the last job on which j depends. Let C_j be the completion time of job j. Moreover, define the *available time* A_j of job j as $C_{\texttt{prev}(j)}$ if $\texttt{prev}(j) \neq \emptyset$; and $A_j = 0$ otherwise. Informally, A_j is the earliest time where j can be executed. The *pending-time* of job j is defined as $C_j - A_j$, that represents the duration from the moment j is available to be executed until its completion. Note that this definition is different (but has some flavour) to the notion of flow-time in scheduling. Additional, a job j is *pending* if it is available but has not been completed.

3 Approximation Algorithm for Completion Time plus Energy Minimization

In this section, we consider the problem of minimizing the total completion time plus energy defined in the previous section. Let G be a collection of arborescences representing job dependencies. For every job j, define the *weight* of job j as $w_j = \sum_{j' : j \preceq j'} 1$. In other words, w_j is the number of jobs which depends on job j (including j itself); equivalently, w_j is the number of nodes in the sub-arborescences rooted at j.

We first make the following observation.

$$\sum_j w_j (C_j - A_j) = \sum_j \Bigg(\sum_{j' : j \preceq j'} 1 \Bigg) (C_j - A_j) = \sum_j \sum_{j' \preceq j} (C_{j'} - A_{j'}) = \sum_j C_j.$$

The last equality holds due to the structure of arborescences: the set $\{j' : j' \preceq j\}$ forms a path $(j_1, j_2, \ldots, j_k)$ where $j_k = j$ and j_1 is the root of the arborescence containing j; so

$A_{j_\ell} = C_{j_{\ell-1}}$ for $2 \leq \ell \leq k$ and $A_{j_1} = 0$. Hence, the total job completion time is equal to the total weighted pending-time of jobs with respect to the weight w_j's defined above. So in order to consider the total completion time, we will rather consider the total weighted pending-time.

Before presenting the algorithm, we define some notions. At a time t, the *remaining volume* of a job j assigned to machine i is denoted as $q_{ij}(t)$. The *density* of job j in machine i is $\delta_{ij} = w_j/p_{ij}$. The *residual density* of a pending job j assigned to machine i at time t is $\delta_{ij}(t) = w_j/q_{ij}(t)$. (As j is pending, $q_{ij}(t) > 0$.)

Our algorithm, named **Algorithm 1**, consists of scheduling and assignment policies described as follows.

1. **Scheduling policy.** At any time t, every machine i sets its speed $s_i(t) = \beta W_i(t)^{1/\alpha}$ where $W_i(t)$ is the total weight of jobs assigned to machine i which are still pending at time t; and $\beta > 0$ is a constant to be chosen later. Moreover, at every time, every machine i processes the highest residual density job among the pending ones assigned to i.
2. **Assignment policy.** Whenever any job j is available, i.e., all jobs $j' \prec j$ have been completed, immediately assign job j to some machine. Note that different assignments of j (to different machines) give rise to different marginal increases of the total weighted pending-time (with respect to the scheduling policy). Here, among all machines, assign (immediately) job j to the one that minimizes the marginal increase of the total weighted pending-time.

Formulation. Let $s_{ij}(t)$ be the variable that represents the speed of job j on machine i at time t. Variables A_j and C_j denote the available time and the completion time of job j, respectively. Let x_{ij} be the variable indicating whether job j is assigned to machine i. The problem could be relaxed as the following formulation. We emphasize that in the formulation, we do *not* relax the integrality of variables x_{ij}'s.

$$
\begin{aligned}
\text{minimize} \quad & \sum_i \int_0^\infty \Big(\sum_j s_{ij}(t)\Big)^\alpha dt + \sum_{i,j} \Big(\int_{A_j}^{C_j} s_{ij}(t)dt\Big) \delta_{ij} x_{ij} (C_j - A_j) \\
& + \frac{\alpha}{\beta(\alpha-1)} \sum_{i,j} \Big(\int_{A_j}^{C_j} s_{ij}(t)dt\Big) x_{ij} w_j^{\frac{\alpha-1}{\alpha}} \\
\text{subject to} \quad & \sum_i x_{ij} = 1 && \forall j \\
& x_{ij} \int_{A_j}^{C_j} s_{ij}(t)dt = p_{ij} x_{ij} && \forall j \\
& A_j = C_{\texttt{prev}(j)} && \forall j : \texttt{prev}(j) \neq \emptyset \\
& A_j = 0 && \forall j : \texttt{prev}(j) = \emptyset \\
& x_{ij} \in \{0,1\} && \forall i,j \\
& s_{ij}(t) \geq 0 && \forall i,j,t \\
& C_j \geq 0 && \forall j
\end{aligned}
$$

The first constraint ensures that every job is assigned to some machine. The second constraint guarantees that if a job j is assigned to some machine i then it will be fully processed during the interval $[A_j, C_j]$ in machine i. In the objective, the first term represents the energy cost. The second term stands for the weighted pending-time of jobs, i.e.,

$$\left(\int_{A_j}^{C_j} s_{ij}(t)dt\right)\delta_{ij}x_{ij}(C_j - A_j) = p_{ij}x_{ij}\delta_{ij}(C_j - A_j) = x_{ij}w_j(C_j - A_j)$$

by the second constraint. The last term in the objective, inspired by [1], is added in order to reduce the integrality gap. In this term, β is a parameter (depending on α) to be chosen later. Note that, in order to minimize the objective function under the above constraints, every algorithm will set $s_{ij}(t) = 0 \ \forall i, j, \forall t \notin [A_j, C_j]$.

The following lemma shows that the objective value of any feasible schedule is within a constant factor of the *cost* of the schedule, which is the sum of the completion times and the energy consumed. The proof follows the scheme of a similar lemma in [1]. For completeness, we give the proof in the appendix.

▶ **Lemma 2.** *Consider a feasible schedule $\mathcal{S}$ for an instance $\mathcal{I}$ of the problem. Let x_{ij} and $s_{ij}(t)$ be the corresponding solution to the mathematical program. Then the objective value of such solution for the mathematical program is at most $(1 + \frac{\alpha}{\beta(\alpha-1)})$ the cost of $\mathcal{S}$.*

Proof. Let C_j be the completion time of job j in schedule $\mathcal{S}$. In the objective of the formulation, the first term clearly captures the consumed energy. Due to the constraints, the second term is $\sum_j w_j\big(C_j - A_j\big)$ — the total weighted pending-time (which equals the total completion time).

In the remaining, we show that the last term in the objective is bounded by $\frac{\alpha}{\beta(\alpha-1)}$ the cost of $\mathcal{S}$. The arguments follow the ones in [1]. In schedule $\mathcal{S}$, assume that job j is executed during $[A_j, C_j]$ in machine i. Then the average speed $\widetilde{s}_{ij}$ of j during $[A_j, C_j]$ is $p_{ij}/(C_j - A_j)$. Thus, $C_j - A_j \geq p_{ij}/\widetilde{s}_{ij}$. The total energy consumed to complete job j is at least $(C_j - A_j)\widetilde{s}_i^{\alpha} \geq p_{ij}\widetilde{s}_i^{\alpha-1}$. Therefore,

$$\begin{aligned}
w_j(C_j - A_j) + p_{ij}\widetilde{s}_i^{\alpha-1} &\geq w_j p_{ij}/\widetilde{s}_i + p_{ij}\widetilde{s}_i^{\alpha-1} \\
&\geq p_{ij}w_j^{\frac{\alpha-1}{\alpha}}\left((\alpha-1)^{\frac{1}{\alpha}} + (\alpha-1)^{-\frac{\alpha-1}{\alpha}}\right) \\
&\geq p_{ij}w_j^{\frac{\alpha-1}{\alpha}} = \sum_{i'}\left(\int_{A_j}^{C_j} s_{i'j}(t)dt\right)x_{i'j}w_j^{\frac{\alpha-1}{\alpha}}.
\end{aligned}$$

The second inequality is due to the first order condition. In the last term, note that $x_{i'j} = 1$ if $i' = i$ and $x_{i'j} = 0$ if $i' \neq i$. As the energy function is convex, the total energy consumed of a schedule is larger than the sum of energy consumed on each individual job. Summing the above inequality for all jobs j, we deduce that the third term in the objective function is bounded by factor $\frac{\alpha}{\beta(\alpha-1)}$ the cost of $\mathcal{S}$. ◀

Dual program and variable setting. The dual of that program is $\max \min_{x,s,C} L$ where L is the Lagrangian function associated to the above mathematical program and the maximum is taken over dual variables. Let λ_{ij} be the dual variable corresponding to the second constraint. Set all dual variables except λ_{ij}'s equal to 0, the Lagrangian function becomes

$$\begin{aligned}
&\sum_i \int_0^{\infty}\left(\sum_j s_{ij}(t)\right)^{\alpha} dt + \sum_j \int_{A_j}^{C_j} \delta_{ij}(C_j - A_j)x_{ij}s_{ij}(t)dt \\
&\quad + \frac{\alpha}{\beta(\alpha-1)}\sum_{i,j}\left(\int_{A_j}^{C_j} s_{ij}(t)dt\right)x_{ij}w_j^{\frac{\alpha-1}{\alpha}} + \sum_{i,j}\lambda_{ij}x_{ij}\left(p_{ij} - \int_{A_j}^{C_j} s_{ij}(t)dt\right)
\end{aligned}$$

Hence, the dual program is

$$\min_{x,s,C}\Bigg\{\sum_{i,j}\lambda_{ij}p_{ij}x_{ij}$$

$$-\sum_{i,j}\int_{A_j}^{C_j}x_{ij}s_{ij}(t)\bigg(\lambda_{ij}-s_i(t)^{\alpha-1}-\frac{\alpha}{\beta(\alpha-1)}w_j^{\frac{\alpha-1}{\alpha}}-\delta_{ij}(C_j-A_j)\bigg)dt\Bigg\}$$

$$\geq \min_{x}\sum_{i,j}\lambda_{ij}p_{ij}x_{ij}$$

$$-\max_{x,s,C}\sum_{i,j}\int_{A_j}^{C_j}x_{ij}s_{ij}(t)\bigg(\lambda_{ij}-s_i(t)^{\alpha-1}-\frac{\alpha}{\beta(\alpha-1)}w_j^{\frac{\alpha-1}{\alpha}}-\delta_{ij}(C_j-A_j)\bigg)dt$$

Choose λ_{ij} such that $\lambda_{ij}p_{ij}$ equals the increase of the total weighted pending-time of jobs (different to j) assigned to machine i plus the weighted pending-time of job j if the latter is assigned to i. In other words, $\lambda_{ij}p_{ij}$ equals the marginal increase in the total weighted pending time if job j is assigned to machine i. Recall that by the assignment policy of the algorithm, job j is assigned to machine i that minimizes $\lambda_{ij}p_{ij}$.

Analysis

The strategy of the analysis is to show that, with the chosen dual variables, the dual has value at least some factor (smaller than 1) times the cost of the algorithm schedule. Then, by weak duality, we derive an approximation ratio for the algorithm.

We first show that the algorithm admits some monotone property. Consider two sets of jobs $\mathcal{I}$ and $\mathcal{I}'$ assigned to machine i such that they are identical except that there is only a job $j \in \mathcal{I} \setminus \mathcal{I}'$ (i.e., $\mathcal{I} = \mathcal{I}' \cup \{j\}$). Moreover, assume that all jobs in $\mathcal{I}'$ have available times earlier than that of j. For every job k, define the *fractional* weight of k in machine i at time t as $w_k q_{ik}(t)/p_{ik}$. Let $V_i(t)$ be the total *fractional* weight of pending jobs assigned to machine i. The following lemma, which has been proved in [1], shows a property of $V_i(t)$.

▶ **Lemma 3** ([1]). *Let I be a set of jobs and $\mathcal{I}' = \mathcal{I} \setminus \{j\}$ where $j \in \mathcal{I}$ is the job with maximum available time (among ones in $\mathcal{I}$). Fix an arbitrary machine i. Let $V_i^{\mathcal{I}}(t)$ and $V_i^{\mathcal{I}'}(t)$ be the total fractional weights of pending jobs at time t in machine i if the sets of jobs assigned to machine i are $\mathcal{I}$ and $\mathcal{I}'$, respectively. Then, $V_i^{\mathcal{I}'}(t) \leq V_i^{\mathcal{I}}(t)$ for every time t.*

Informally, Lemma 3 shows that for every machine i, $V_i(t)$ is monotone w.r.t the set of jobs assigned to machine i. In fact, Lemma 3 is proved by Anand et al. [1] in the online setting. The proof remains exactly the same by replacing the available times A_j's in our setting by the release times r_j's of jobs in the online setting.

We are now proving a crucial lemma relating the dual variables and the fractional pending weights.

▶ **Lemma 4.** *It holds that $\lambda_{ij} - \delta_j(t-A_j) - \frac{\alpha}{\beta(\alpha-1)}w_j^{\frac{\alpha-1}{\alpha}} \leq \frac{\alpha}{\beta(\alpha-1)}V_i(t)^{\frac{\alpha-1}{\alpha}}$ for every machine i and every time $t \geq A_j$.*

Proof. By Lemma 3, it is sufficient to prove the inequality for a fixed machine i assuming that no new job will be assigned to i after A_j. For simplicity of the notations, as machine i is fixed, in the remaining of the proof, we drop the index of the machines in all the parameters (e.g., $\delta_j(t)$ stands for $\delta_{ij}(t)$, etc). Moreover, denote again $q_k = q_k(A_j)$ and $\delta_k = \delta_k(A_j)$ for every pending job k. At A_j, rename jobs in non-increasing order of their residual densities, i.e., $q_1/w_1 \leq \ldots \leq q_n/w_n$ (note that q_k/w_k is the inverse of job k's residual density). Denote

$W_k = w_k + \ldots + w_n$ for $1 \leq k \leq n$. The marginal increase in the total weighted pending-time due to the assignment of job j is

$$w_j\left(\frac{q_1}{\beta W_1^{1/\alpha}} + \ldots + \frac{q_j}{\beta W_j^{1/\alpha}}\right) + W_{j+1}\frac{q_j}{\beta W_j^{1/\alpha}}$$

where the first term is the weighted pending-time of job j and the second one is the increase of the weighted pending-time of other jobs (note that only jobs with density smaller than that of j has their completion times increased). Let C_j^* be the completion time of job j if it is assigned to machine i. We consider different cases of time t.

Case 1: $t \leq C_j^*$**.** Let k be the pending job at t with the smallest index. In other words, the machine has processed all jobs $1, \ldots, k-1$ and a part of job k in interval $[A_j, t]$. By the definition of λ_j, we have that

$$\begin{aligned}
\lambda_j - \delta_j(t - A_j) &= \delta_j\left(\frac{q_k(t)}{\beta W_k^{1/\alpha}} + \frac{q_{k+1}}{\beta W_{k+1}^{1/\alpha}} + \ldots + \frac{q_j}{\beta W_j^{1/\alpha}}\right) + \frac{W_{j+1}}{\beta W_j^{1/\alpha}} \\
&= \delta_j\left(\frac{w_k(t)}{\delta_k \beta W_k^{1/\alpha}} + \frac{w_{k+1}}{\delta_{k+1}\beta W_{k+1}^{1/\alpha}} + \ldots + \frac{w_j}{\delta_j \beta W_j^{1/\alpha}}\right) + \frac{W_{j+1}}{\beta W_j^{1/\alpha}} \\
&\leq \frac{1}{\beta}\left(\frac{w_k(t)}{W_k^{1/\alpha}} + \frac{w_{k+1}}{W_{k+1}^{1/\alpha}} + \ldots + \frac{w_j}{W_j^{1/\alpha}} + \frac{w_{j+1}}{W_{j+1}^{1/\alpha}} + \ldots + \frac{w_n}{W_n^{1/\alpha}}\right) \\
&\leq \frac{1}{\beta}\int_{w_n}^{V(t)+w_j} \frac{dz}{z^{1/\alpha}} \leq \frac{\alpha}{\beta(\alpha-1)}(V(t) + w_j)^{\frac{\alpha-1}{\alpha}} \\
&\leq \frac{\alpha}{\beta(\alpha-1)}\left(V(t)^{\frac{\alpha-1}{\alpha}} + w_j^{\frac{\alpha-1}{\alpha}}\right).
\end{aligned}$$

The second equality is due to the definition of the residual density. The first inequality holds since $\delta_j \leq \delta_{k'}$ for every job $k' \leq j$ and $W_j \geq W_{j+1} \geq \ldots \geq W_n$. The second inequality holds since function $z^{-1/\alpha}$ is decreasing. The last inequality holds because $0 < (\alpha-1)/\alpha < 1$.

Case 2: $t > C_j^*$**.** Let k be the pending job at t with the smallest index. We have

$$\begin{aligned}
\lambda_j - \delta_j(t - A_j) &= \frac{W_{j+1}}{\beta W_j^{1/\alpha}} - \delta_j(t - C_j^*) = \frac{1}{\beta W_j^{1/\alpha}}\Big(w_{j+1} + \ldots + w_n\Big) - \delta_j(t - C_j^*) \\
&\leq \delta_{j+1}\frac{q_{j+1}}{\beta W_{j+1}^{1/\alpha}} + \ldots + \delta_n \frac{q_n}{\beta W_n^{1/\alpha}} - \delta_j(t - C_j^*) \\
&\leq \delta_k \frac{q_k(t)}{\beta W_k^{1/\alpha}} + \delta_{k+1}\frac{q_{k+1}}{\beta W_{k+1}^{1/\alpha}} + \ldots + \delta_n \frac{q_n}{\beta W_n^{1/\alpha}} \\
&= \delta_k \frac{w_k(t)}{\delta_k \beta W_k^{1/\alpha}} + \delta_{k+1}\frac{w_{k+1}}{\delta_{k+1}\beta W_{k+1}^{1/\alpha}} + \ldots + \delta_n \frac{w_n}{\delta_n \beta W_n^{1/\alpha}} \\
&\leq \frac{1}{\beta}\int_{w_n}^{V(t)} \frac{dz}{z^{1/\alpha}} \leq \frac{\alpha}{\beta(\alpha-1)} V(t)^{\frac{\alpha-1}{\alpha}}
\end{aligned}$$

where the first inequality holds since $W_j \geq W_{j+1} \geq \ldots \geq W_n$; the second inequality is due to $\delta_j \geq \delta_{k'}$ for every job $k' > j$.

Combining both cases, the lemma follows. ◀

▶ **Theorem 5.** *Algorithm 1 is* $8(1 + \frac{\alpha}{\ln \alpha})$*-approximation for* $\beta = \frac{1}{\alpha-1}(\alpha - 1 + \ln(\alpha-1))^{\frac{\alpha-1}{\alpha}}$.

Proof. Let $\mathcal{P}^*$ be the total weighted pending-time due to the algorithm (that also equals the total completion time). By the choice of dual variables, we have

$$
\begin{aligned}
\min_{x,s,C} L = \min_{x} & \sum_{i,j} \lambda_{ij} p_{ij} x_{ij} \\
& - \max_{x,s,C} \sum_{i,j} \int_{A_j}^{C_j} x_{ij} s_{ij}(t) \Big(\lambda_{ij} - s_i(t)^{\alpha-1} - \frac{1}{\beta} w_{ij}^{\frac{\alpha-1}{\alpha}} - \delta_j (C_j - A_j) \Big) dt \\
& \geq \mathcal{P}^* - \max_{x,s,C} \sum_{i,j} \int_{A_j}^{C_j} x_{ij} s_{ij}(t) \Big(\lambda_{ij} - s_i(t)^{\alpha-1} - \frac{1}{\beta} w_{ij}^{\frac{\alpha-1}{\alpha}} - \delta_j (t - A_j) \Big) dt \\
& \geq \mathcal{P}^* - \max_{x,s,C} \sum_{i,j} \int_{A_j}^{C_j} x_{ij} s_{ij}(t) \Big(\frac{\alpha}{\beta(\alpha-1)} V_i(t)^{\frac{\alpha-1}{\alpha}} - s_i(t)^{\alpha-1} \Big) dt \\
& = \mathcal{P}^* - \max_{x,s,C} \sum_{i} \int_{0}^{\infty} \Big(\sum_{j} x_{ij} s_j(t) \Big) \Big(\frac{\alpha}{\beta(\alpha-1)} V_i(t)^{\frac{\alpha-1}{\alpha}} - s_i(t)^{\alpha-1} \Big) dt \\
& \geq \mathcal{P}^* - \max_{x,s,C} \sum_{i} \int_{0}^{\infty} s_i(t) \Big(\frac{\alpha}{\beta(\alpha-1)} V_i(t)^{\frac{\alpha-1}{\alpha}} - s_i(t)^{\alpha-1} \Big) dt
\end{aligned}
$$

where the first inequality follows by the assignment policy (assign job j to machine i that minimizes $\lambda_{ij} p_{ij}$) and $t \leq C_j$; the second inequality is due to Lemma 4. By the first order condition, function $z(\frac{\alpha}{\beta(\alpha-1)} V^{\frac{\alpha-1}{\alpha}} - z^{\alpha-1})$ is maximized at $z_0 = \frac{V^{1/\alpha}}{((\alpha-1)\beta)^{1/(\alpha-1)}}$. We have

$$
\begin{aligned}
\min_{x,s,C} L & \geq \mathcal{P}^* - \frac{\alpha - 1}{((\alpha-1)\beta)^{\frac{\alpha}{\alpha-1}}} \sum_{i} \int_{0}^{\infty} V_i(t) dt \\
& \geq \mathcal{P}^* - \frac{\alpha - 1}{((\alpha-1)\beta)^{\frac{\alpha}{\alpha-1}}} \sum_{i} \int_{0}^{\infty} W_i(t) dt = \Big(1 - \frac{\alpha - 1}{((\alpha-1)\beta)^{\frac{\alpha}{\alpha-1}}} \Big) \mathcal{P}^*
\end{aligned}
$$

where the second inequality holds since $V_i(t) \leq W_i(t)$ for every i and t.

Besides, the total weighted pending-time plus energy is

$$
\mathcal{P}^* + \int_{0}^{\infty} s^{\alpha}(t) dt = \mathcal{P}^* + \sum_{i} \int_{0}^{\infty} \beta^{\alpha} W_i(t) dt = (1 + \beta^{\alpha}) \mathcal{P}^*.
$$

Therefore the primal objective is bounded by $\big((1+\beta^{\alpha}) + \frac{\alpha}{\beta(\alpha-1)}(1+\beta^{\alpha})\big)\mathcal{P}^*$ (Lemma 2). Thus, the approximation ratio is at most

$$
\frac{(1+\beta^{\alpha}) + \frac{\alpha}{\beta(\alpha-1)}(1+\beta^{\alpha})}{1 - \frac{\alpha-1}{((\alpha-1)\beta)^{\frac{\alpha}{\alpha-1}}}} \tag{1}
$$

Choose $\beta = \frac{1}{\alpha-1}(\alpha - 1 + \ln(\alpha-1))^{\frac{\alpha-1}{\alpha}}$. Observe that

$$
\begin{aligned}
& \Big(1 + \frac{\ln(\alpha-1)}{\alpha-1} \Big)^{\alpha-1} < e^{\ln(\alpha-1)} = \alpha - 1 \\
& \Rightarrow (\alpha - 1 + \ln(\alpha-1))^{\alpha-1} < (\alpha-1)^{\alpha} \Rightarrow \beta < 1
\end{aligned}
$$

Moreover, $\beta > (\alpha-1)^{-1/\alpha}$. With the chosen β, the denominator of (1) becomes $\frac{\ln(\alpha-1)}{\alpha-1+\ln(\alpha-1)}$ and the nominator is bounded by 8 (since $\alpha^{-1/\alpha} < \beta < 1$ and $\alpha \geq 2$). Hence, the approximation ratio is at most $8(1+\alpha/\ln\alpha)$. ◀

4 Approximation Algorithm for $R|arborescences|\sum_j C_j$

We are now considering the problem $R|arborescences|\sum_j C_j$. Fix the parameter α such that $\alpha^\alpha = n$, so $\alpha = \Theta\left(\frac{\log n}{\log \log n}\right)$. Notice that given an instance of $R|arborescences|\sum_j C_j$, there is a corresponding instance of the problem of minimizing the total completion time plus energy in which the energy function of every machine is $\int_0^\infty s_i(t)^\alpha dt$. Our algorithm **Algorithm 2** for the $R|arborescences|\sum_j C_j$ problem is the following.

1. Given an instance of $R|arborescences|\sum_j C_j$, consider the corresponding instance of the problem of minimizing the total completion time plus energy (defined in Section 2) with parameter α such that $\alpha^\alpha = n$. Solve the latter by Algorithm 1 and obtain a schedule $\mathcal{S}_1$ (with machine speeds).
2. Transform the schedule $\mathcal{S}_1$ to a schedule $\mathcal{S}_2$ such that at any time t where $s_i(t) > \alpha$ for some machine i, reduce the speed $s_i(t)$ to α. Note that this transformation might delay job completion times.
3. Given the schedule $\mathcal{S}_2$, transform to a unit-speed schedule $\mathcal{S}_3$ as follows. In the schedule $\mathcal{S}_3$, preserve the job-to-machine assignments as in schedule $\mathcal{S}_2$. In every machine, execute jobs non-preemptively (by unit-speed) in the non-decreasing order of their completion times in schedule $\mathcal{S}_2$. Return the non-preemptive schedule $\mathcal{S}_3$.

We first show some properties of schedules $\mathcal{S}_2$ and $\mathcal{S}_3$.

▶ **Lemma 6.** **1.** *The cost (i.e., total completion time plus energy) of the schedule $\mathcal{S}_2$ is at most that of schedule $\mathcal{S}_1$.*

2. *The total completion time of $\mathcal{S}_3$ is at most α times that of $\mathcal{S}_2$.*

Proof. **1.** Assume that the speed of some machine i at some time t is $s_i(t) > \alpha$. The increasing rate of energy cost in machine i at time t is

$$\frac{d(s_i(t))^\alpha}{dt} = \alpha s_i^{\alpha-1}(t) > \alpha^\alpha = n.$$

However, the increasing rate of the total completion time is at most n. Therefore, one can reduce the speed $s_i(t)$ to get a smaller cost. Hence, by operations of Step 2 in Algorithm 2, the total completion time plus energy of the schedule $\mathcal{S}_2$ is at most that of schedule $\mathcal{S}_1$.

2. If the speed of a machine is reduced by a factor α then the completion time of each job will be increased by at most a factor α. Therefore, the total completion time is increased by at most a factor α.

◀

▶ **Theorem 7.** *Algorithm 2 is $O\left(\frac{\log^2 n}{(\log \log n)^3}\right)$-approximation for the problem $R|arborescences|\sum_j C_j$.*

Proof. Let $\mathcal{C}(\mathcal{S})$ and $\mathcal{E}(\mathcal{S})$ be the total completion time and the energy of the schedule S, respectively. Let $\mathcal{S}^*$ be an optimal schedule for the problem of minimizing the total completion time plus energy. Let OPT be an optimal schedule for the problem $R|arborescences|\sum_j C_j$. Note that OPT is a feasible solution to the problem of minimizing the total completion time plus energy where at any time the machine speeds are unit (whenever there is still a pending job). We have

$$\begin{aligned}\mathcal{C}(\mathcal{S}_3) &\le \alpha \cdot \mathcal{C}(\mathcal{S}_2) \le \alpha \cdot (\mathcal{C}(\mathcal{S}_2) + \mathcal{E}(\mathcal{S}_2)) \le \alpha \cdot (\mathcal{C}(\mathcal{S}_1) + \mathcal{E}(\mathcal{S}_1)) \\ &\le 8\alpha\left(1 + \frac{\alpha}{\log \alpha}\right)(\mathcal{C}(S^*) + \mathcal{E}(S^*)) \le 8\alpha\left(1 + \frac{\alpha}{\log \alpha}\right)(\mathcal{C}(OPT) + \mathcal{E}(OPT)) \\ &\le 16\alpha\left(1 + \frac{\alpha}{\log \alpha}\right) \cdot \mathcal{C}(OPT)\end{aligned}$$

The first and third inequalities follow from Lemma 6. The fourth inequality is due to Theorem 5. The last inequality holds since in OPT, at every time every machine runs with speed either 1 or 0, so the total energy incurred in a machine is bounded by the maximum completion time of a job in that machine. The theorem follows since $\alpha = \Theta\big(\frac{\log n}{\log \log n}\big)$. ◀

Remark. The weighted version $R|arborescences|\sum_j w_j C_j$ can be solved by a similar algorithm and the approximation ratio will be $O\big(\rho \cdot \frac{\log^2 n}{(\log \log n)^3}\big)$ where $\rho = \max_{j,j':w_{j'}>0} \frac{w_j}{w_{j'}}$.

5 Conclusion

In this paper, we present a new approach for the problem $R|arborescences|\sum_j C_j$ using non-convex formulations and a dual-fitting method. In high level, the consideration of a smooth variant of the problem helps to bypass a hard constraint of the problem (that every job has to be processed by unit speed). Moreover, the formulation of a non-convex program with mixed integer variables (assignment variables) and continuous variables (speed variables) allows us to get rid of the integrality gap issue while still benefit from several continuous aspects. Finally, the analysis holds by the simple yet powerful weak duality which holds even for non-convex programs. The approach enables an improvement, albeit rather small, over a long standing approximation. We hope that the approach would provide additional tools and a different point of view towards the design of algorithms with improved performance guarantees for the general problems $R|prec|\sum_j C_j$ and $R|prec|\sum_j w_j C_j$.

References

1 S. Anand, Naveen Garg, and Amit Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *Proc. 23rd ACM-SIAM Symposium on Discrete Algorithms*, pages 1228–1241, 2012.

2 Nikhil Bansal and Subhash Khot. Optimal long code test with one free bit. In *Proc. 50th Symposium on Foundations of Computer Science*, pages 453–462, 2009.

3 Abbas Bazzi and Ashkan Norouzi-Fard. Towards tight lower bounds for scheduling problems. In *Proc. 23rd European Symposium on Algorithms*, pages 118–129, 2015.

4 Soumen Chakrabarti, Cynthia A Phillips, Andreas S Schulz, David B Shmoys, Cliff Stein, and Joel Wein. Improved scheduling algorithms for minsum criteria. In *Proc. Colloquium on Automata, Languages, and Programming*, pages 646–657, 1996.

5 Chandra Chekuri and Sanjeev Khanna. Approximation algorithms for minimizing average weighted completion time. In *Handbook of Scheduling*, pages 220–249. Chapman and Hall/CRC, 2004.

6 Fabián A Chudak and David B Shmoys. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. *Journal of Algorithms*, 30(2):323–343, 1999.

7 Shashwat Garg. Quasi-ptas for scheduling with precedences using lp hierarchies. In *Proc. 45th Colloquium on Automata, Languages, and Programming*, 2018.

8 Shashwat Garg, Janardhan Kulkarni, and Shi Li. Lift and project algorithms for precedence constrained scheduling to minimize completion time. In *Proc. 30th Symposium on Discrete Algorithms*, pages 1570–1584, 2019.

9 Ronald L Graham, Eugene L Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

10 Han Hoogeveen, Petra Schuurman, and Gerhard J Woeginger. Non-approximability results for scheduling problems with minsum criteria. *INFORMS Journal on Computing*, 13(2):157–168, 2001.

11 Klaus Jansen, Roberto Solis-Oba, and Maxim Sviridenko. Makespan minimization in job shops: a polynomial time approximation scheme. In *Proc. Symposium on Theory of Computing*, volume 99, pages 394–399, 1999.

12 Anil Kumar, Madhav Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. Scheduling on unrelated machines under tree-like precedence constraints. *Algorithmica*, 55(1):205–226, 2009.

13 Jan Karel Lenstra and AHG Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22–35, 1978.

14 Elaine Levey and Thomas Rothvoss. A $(1 + epsilon)$-approximation for makespan scheduling with precedence constraints using lp hierarchies. In *Proc. 48th Symposium on Theory of Computing*, pages 168–177, 2016.

15 Shi Li. Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. In *Proc. 58th Symposium on Foundations of Computer Science (FOCS)*, pages 283–294, 2017.

16 Alix Munier, Maurice Queyranne, and Andreas S Schulz. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. In *Proc. Conference on Integer Programming and Combinatorial Optimization*, pages 367–382, 1998.

17 David B Shmoys, Clifford Stein, and Joel Wein. Improved approximation algorithms for shop scheduling problems. *SIAM Journal on Computing*, 23(3):617–632, 1994.

18 Martin Skutella. A 2.542-approximation for precedence constrained single machine scheduling with release dates and total weighted completion time objective. *Operations Research Letters*, 44(5):676–679, 2016.

19 Ola Svensson. Hardness of precedence constrained scheduling on identical machines. *SIAM Journal on Computing*, 40(5):1258–1274, 2011.

20 Nguyen Kim Thang. Lagrangian duality in online scheduling with resource augmentation and speed scaling. In *Proc. 21st European Symposium on Algorithms*, pages 755–766, 2013.