Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 3203728, 5 pages
http://dx.doi.org/10.1155/2016/3203728

Research Article

Parallel Machine Scheduling with Nested Processing Set Restrictions and Job Delivery Times

Shuguang Li[1,2]

[1] *Key Laboratory of Intelligent Information Processing in Universities of Shandong (Shandong Institute of Business and Technology), Yantai 264005, China*
[2] *College of Computer Science and Technology, Shandong Institute of Business and Technology, Yantai 264005, China*

Correspondence should be addressed to Shuguang Li; sgliytu@hotmail.com

Received 7 June 2016; Accepted 4 September 2016

Academic Editor: Bruno G. M. Robert

The problem of scheduling jobs with delivery times on parallel machines is studied, where each job can only be processed on a specific subset of the machines called its processing set. Two distinct processing sets are either nested or disjoint; that is, they do not partially overlap. All jobs are available for processing at time 0. The goal is to minimize the time by which all jobs are delivered, which is equivalent to minimizing the maximum lateness from the optimization viewpoint. A list scheduling approach is analyzed and its approximation ratio of 2 is established. In addition, a polynomial time approximation scheme is derived.

1. Introduction

The problems of scheduling with processing set restrictions have been extensively studied in the past few decades [1, 2]. In this class of problems, we are given a set of n jobs $\mathcal{J} = \{1, 2, \ldots, n\}$ and a set of m parallel machines $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$. Each job j can only be processed on a certain subset $\mathcal{M}_j$ of the machines, called its *processing set*, and, on those machines, it takes p_j time units of uninterrupted processing to complete. Each machine can process at most one job at a time. The goal is to find an optimal schedule where optimality is defined by some problem dependent objective.

There are several important special cases of processing set restrictions: *inclusive*, *nested*, *interval*, and *tree-hierarchical* [2]. In the case of inclusive processing set, for any two jobs j_1 and j_2, either $\mathcal{M}_{j_1} \subseteq \mathcal{M}_{j_2}$ or $\mathcal{M}_{j_2} \subseteq \mathcal{M}_{j_1}$. In the nested processing set case, either $\mathcal{M}_{j_1} \cap \mathcal{M}_{j_2} = \phi$, or $\mathcal{M}_{j_1} \subseteq \mathcal{M}_{j_2}$, or $\mathcal{M}_{j_2} \subseteq \mathcal{M}_{j_1}$. In the interval processing set case, the machines are linearly ordered, and each job j is associated with two machine indices a_j and b_j such that $\mathcal{M}_j = \{M_{a_j}, M_{a_j+1}, \ldots, M_{b_j}\}$. It is easy to see that the inclusive processing set and the nested processing set are two special cases of the interval processing set. In the tree-hierarchical processing set case, each machine is represented by a vertex of a tree, and each job j is associated with a machine index a_j such that $\mathcal{M}_j$ consists of the machines on the unique path from M_{a_j} to the root of the tree.

In this paper, we consider the problem of scheduling jobs with nested processing set restrictions on parallel machines. Besides its *processing time* p_j and processing set $\mathcal{M}_j$, each job j requires an additional *delivery time* q_j after completing its processing. If S_j denotes the time job j starts processing, it has been delivered at time $L_j = S_j + p_j + q_j$, which is called its *delivery completion time*. All jobs are available for processing at time 0. The objective is to minimize the time by which all jobs are delivered, that is, the *maximum delivery completion time*, $L_{\max} = \max_j L_j$. Minimizing the maximum delivery completion time is equivalent to minimizing the maximum lateness from the optimization viewpoint [3]. Following the classification scheme for scheduling problems by Graham et al. [4], this problem is noted: $P \mid M_j(\text{nested}) \mid L_{\max}$.

The motivation for this problem is the scenario in which the jobs (with nested processing set restrictions) are first processed on the machines and then delivered to their respective customers. In order to be competitive, the jobs are needed to be delivered as soon as possible to their customers. Thus, the industry practitioners are required to coordinate job production and job delivery. In manufacturing and distribution systems, finished jobs are delivered by vehicles such as trucks. Since there are sufficient vehicles for delivering jobs, delivery is a nonbottleneck activity. Therefore, we assume that all jobs may be simultaneously delivered. Considering job production and job delivery as one system, we choose the cost function to measure the customer service level. In particular, we are interested in the objective of minimizing the time by which all jobs are delivered.

The problem as stated is a natural generalization of strongly NP-hard problem $P \parallel C_{\max}$, which corresponds to the special case where all $\mathscr{M}_j = \mathscr{M}$ and all $q_j = 0$ [5]. For NP-hard problems, the research focuses on developing polynomial time approximation algorithms. Given instance $\mathscr{I}$ of a minimization problem and approximation algorithm A, let $A(\mathscr{I})$ and $\mathrm{OPT}(\mathscr{I})$ denote the objective value of the solution obtained by algorithm A and the optimal solution value, respectively, when applied to $\mathscr{I}$. If $A(\mathscr{I})/\mathrm{OPT}(\mathscr{I}) \leq \rho$ for all $\mathscr{I}$, then we say that algorithm A has *approximation ratio* ρ, and A is called ρ-approximation algorithm for this problem. Ideally, one hopes to obtain a family of polynomial time algorithms such that, for any given $\varepsilon > 0$, the corresponding algorithm is $(1+\varepsilon)$-approximation algorithm; such a family is called a *polynomial time approximation scheme* (PTAS) [6].

When all $\mathscr{M}_j = \mathscr{M}$, $P \mid M_j(\text{nested}) \mid L_{\max}$ reduces to the classic scheduling problem $P \parallel L_{\max}$. Woeginger [7] provided three fast heuristics for $P \parallel L_{\max}$ with approximation ratios 2, $2-1/m$, and $2-2/(m+1)$, respectively. For the single machine case with release times $1 \mid r_j \mid L_{\max}$, Potts [8] presented a 1.5-approximation algorithm, and Hall and Shmoys [9, 10] proposed two PTASs whose running times are $O((n/\varepsilon)^{O(1/\varepsilon)})$ and $O(n \log n + n(1/\varepsilon)^{O(1/\varepsilon^2)})$. For the parallel machines case with release times $P \mid r_j \mid L_{\max}$, Hall and Shmoys [10] obtained a PTAS with running time $O(n^{\text{poly}(1/\varepsilon)})$, and Mastrolilli [11] developed a PTAS that runs in $O(n + f(1/\varepsilon))$, where $f(1/\varepsilon)$ is a constant that depends exponentially on $1/\varepsilon$. Mastrolilli [11] also presented an improved PTAS for $1 \mid r_j \mid L_{\max}$ that runs in $O(n + (1/\varepsilon)^{O(1/\varepsilon)})$ time.

When all $q_j = 0$, $P \mid M_j(\text{nested}) \mid L_{\max}$ reduces to the problem of minimizing makespan with nested processing set restrictions, $P \mid M_j(\text{nested}) \mid C_{\max}$. There are a number of approximation algorithms for $P \mid M_j(\text{nested}) \mid C_{\max}$: $(2-1/m)$-approximation algorithm [12], 7/4-approximation algorithm [13], 5/3-approximation algorithm [14], and two PTASs [15, 16].

As mentioned above, the classic scheduling problem (without processing set restrictions) of minimizing the maximum delivery completion time and the problem of minimizing makespan with nested processing set restrictions have been studied in the literature. However, to the best of our knowledge, the problem of minimizing the maximum delivery completion time with nested processing set restrictions, $P \mid M_j(\text{nested}) \mid L_{\max}$, has not been studied to date. In this paper, we first use Graham's *list scheduling* [17] to get a simple and fast 2-approximation algorithm. We then derive a polynomial time approximation scheme, which is heavily built on the ideas of [15]. The PTAS result generalizes the approximation schemes of [15, 16], both of which deal with only the special case where all $q_j = 0$.

The paper is organized into sections as follows. Section 2 presents a 2-approximation algorithm which uses list scheduling as a subroutine. The next three sections are devoted to designing the polynomial time approximation scheme. Section 3 shows how to simplify the input instance to get a so-called rounded instance. Section 4 shows how to solve the rounded instance optimally. Section 5 wraps things up to derive the polynomial time approximation scheme. The discussion in Section 6 completes the paper.

2. A 2-Approximation Algorithm

In this section, we will present a simple and fast 2-approximation algorithm for $P \mid M_j(\text{nested}) \mid L_{\max}$.

As observed in [12], nested processing sets have a partial ordering defined by the inclusion relationship and, thus, offer a natural topological sort on jobs, taking more constrained jobs first.

We now explore the behavior of Graham's list scheduling algorithm [17], with jobs sorted to respect nestedness. It does not depend on the delivery times. The algorithm is called Nested-LS.

Nested-LS

Step 1. Place all the jobs in a list in the order of the topological sort on jobs, taking more constrained jobs first. Set $\text{Load}_i = 0$, $i = 1, 2, \ldots, m$.

Step 2. For the first unscheduled job j in the list, select a machine $M_i \in \mathscr{M}_j$ for which Load_i is the smallest (ties broken arbitrarily). Assign job j to machine M_i. Set $\text{Load}_i = \text{Load}_i + p_j$. Repeat this step until all the jobs are scheduled.

The *load* on a machine is defined to be the total processing time of the jobs assigned to this machine. The quantity Load_i represents the current load on machine M_i during the run of Nested-LS, $i = 1, 2, \ldots, m$.

Theorem 1. *Nested-LS is a 2-approximation algorithm for $P \mid M_j(\text{nested}) \mid L_{\max}$ that runs in $O(n \log mn)$ time.*

Proof. Let OPT be the objective value of an optimal schedule. Denote by $L_{\max}$ the objective value of the schedule generated by Nested-LS. Let j_1 be the first job in the list generated in Step 2 for which $L_{\max} = S_{j_1} + p_{j_1} + q_{j_1}$ holds, where S_{j_1} denotes the time job j_1 starts processing. Remove all the following jobs from the list. Remove all the machines in $\mathscr{M} \setminus \mathscr{M}_{j_1}$ and all the jobs assigned to these machines. This cannot increase OPT and does not decrease $L_{\max}$.

A straightforward lower bound on OPT is $\mathrm{OPT} \geq p_j + q_j$ for any job j. By the rule of Nested-LS, at the time when job j_1 is assigned to its machine M_i, M_i is the least-loaded machine among the machines in $\mathscr{M}_{j_1}$. It follows that $S_{j_1} < \mathrm{OPT}$. Consequently, we get $L_{\max} = S_{j_1} + p_{j_1} + q_{j_1} < 2\mathrm{OPT}$.

The initial step of sorting the jobs to respect nestedness requires $O(n \log n)$ time. Assigning a job to the least-loaded eligible machine in Step 2 runs in $O(\log m)$ time. Thus, Nested-LS runs in $O(n \log mn)$ time. □

3. Simplifying the Input

The nested structure of processing sets can be depicted by rooted tree $T = (V, E)$, in which each processing set is represented by a vertex, and the predecessor relationship is defined by inclusion of the processing sets. Each machine can be regarded as a one-element processing set and thus it corresponds to a leaf in T, even if there are no jobs associated with this processing set. The root of T corresponds to $\mathscr{M} = \{M_1, M_2, \ldots, M_m\}$.

For each vertex $v \in V(T)$, let $\mathscr{M}(v)$ denote the set of machines associated with v, which is the disjoint union of all the processing sets associated with the sons of v. Let $\mathscr{J}(v)$ denote the set of jobs j for which $\mathscr{M}_j$ coincides with $\mathscr{M}(v)$.

Following [15], we transform tree T into a binary tree as follows. If there is vertex v with at least three sons, create new vertex u as the new father of two sons of v and as a new son of v. Repeat this procedure until we reach a binary tree, which consists of m leaves and of $m-1$ nonleaf vertices. Therefore, we can assume without loss of generality that T is a binary tree.

Given instance I of $P \mid M_j(\text{nested}) \mid L_{\max}$, let $\text{OPT}(I)$ be the objective value of an optimal schedule for I. Let A be the objective value of the schedule for I generated by Nested-LS. We have

$$\text{OPT}(I) \le A \le 2\text{OPT}(I). \tag{1}$$

Let α be some fixed positive integer. Classify jobs as big and small according to their processing times. Job j is *big*, if $p_j > A/(\alpha(\alpha+1))$, and otherwise it is *small*.

Modify I to get rounded instance $I^{\#}(\alpha)$ as follows:

(i) For each job j, round its delivery time q_j up to the nearest integer multiple of A/α; that is, set rounded delivery time $q_j^{\#} = \lceil q_j/(A/\alpha) \rceil \cdot (A/\alpha)$. Note that there are at most $\alpha + 1$ different delivery times in the rounded instance. Let $\xi_l = (l-1)A/\alpha$ denote lth delivery time, $l = 1, 2, \ldots, \alpha+1$.

(ii) For each big job j, round its processing time p_j up to the nearest integer multiple of $A/(\alpha^2(\alpha+1))$. Let $p_j^{\#}$ denote the rounded processing time of big job j. Note $p_j \le p_j^{\#} \le (1 + 1/\alpha)p_j$.

(iii) For each vertex $v \in V(T)$, let $TS_l(v)$ be the total processing time of the small jobs with delivery time ξ_l in $\mathscr{J}(v)$. Let $TS_l^{\#}(v)$ be the value of $TS_l(v)$ rounded up to the nearest integer multiple of $A/(\alpha(\alpha+1))$. The small jobs with delivery time ξ_l in $\mathscr{J}(v)$ are replaced with $TS_l^{\#}(v) \cdot \alpha(\alpha+1)/A$ new jobs each of which has delivery time ξ_l and processing time $A/(\alpha(\alpha+1))$, $l = 1, 2, \ldots, \alpha+1$.

Lemma 2. *There is schedule $\Sigma^{\#}$ for $I^{\#}(\alpha)$ with objective value $L^{\#} \le (1 + 1/\alpha) \cdot OPT(I) + 4A/\alpha$.*

Proof. Let Σ be an optimal schedule for instance I with objective value $\text{OPT}(I)$. Recall that the single machine case $1 \parallel L_{\max}$ can be solved optimally by Jackson's rule [18]: process the jobs successively in order of nondecreasing delivery times. Hence, we can assume that in Σ on each machine the jobs with the same delivery time are processed together in succession. Let TS_{il} be the total processing time of the small jobs with delivery time ξ_l processed on machine M_i in Σ. Let $TS_{il}^{\#}$ be the value of TS_{il} rounded up to the nearest integer multiple of $A/(\alpha(\alpha+1))$. Replace the small jobs with delivery time ξ_l processed on machine M_i in Σ by $2 + TS_{il}^{\#} \cdot \alpha(\alpha+1)/A$ slots each of which has delivery time ξ_l and size $A/(\alpha(\alpha+1))$, $i = 1, 2, \ldots, m$, $l = 1, 2, \ldots, \alpha+1$.

We next explain how to assign all small jobs in $I^{\#}(\alpha)$ to these slots of size $A/(\alpha(\alpha+1))$. Starting from the leaves of tree T, we work our way towards the root in a bottom-up fashion. Suppose that we are handling vertex $v \in V(T)$. At this point, all the descendants of v except v itself have already been handled and some slots have been occupied. Let T_v be the subtree of T rooted at v which contains exactly $2m(v) - 1$ vertices, where $m(v) = |\mathscr{M}(v)|$ denotes the number of machines (leaves) in T_v. Let β_l denote the total processing time of the small jobs with delivery time ξ_l in all descendants of v (including v itself) in instance I, and let $\beta_l^{\#}$ denote the total processing time of the corresponding small jobs in $I^{\#}(\alpha)$, $l = 1, 2, \ldots, \alpha+1$. Let $\gamma_l^{\#}$ denote the total size of the slots with delivery time ξ_l and size $A/(\alpha(\alpha+1))$ on all machines in descendant leaves of v. We have the following two inequalities: $\beta_l^{\#} \le \beta_l + (2m(v) - 1)A/(\alpha(\alpha+1))$ and $\gamma_l^{\#} \ge \beta_l + 2m(v)A/(\alpha(\alpha+1))$. Therefore, we get $\beta_l^{\#} \le \gamma_l^{\#}$.

Let $\mathcal{SJ}_1^{\#}(v)$ denote the set of small jobs $j^{\#}$ in $I^{\#}(\alpha)$ with delivery time $q_{j^{\#}} = \xi_l$, processing time $p_{j^{\#}} = A/(\alpha(\alpha+1))$, and processing set $\mathscr{M}_{j^{\#}} = \mathscr{M}(v)$. Since $\beta_l^{\#} \le \gamma_l^{\#}$, when we handle vertex v, there are enough unoccupied slots of size $A/(\alpha(\alpha+1))$ to accommodate the jobs in $\mathcal{SJ}_l^{\#}(v)$. For each job in $\mathcal{SJ}_l^{\#}(v)$ $(l = 1, 2, \ldots, \alpha+1)$, we assign an unoccupied slot to it and then mark this slot as occupied.

After we handle the root of T, we fit all small jobs in $I^{\#}(\alpha)$ into the slots of size $A/(\alpha(\alpha+1))$ and thereby get an assignment of the small jobs in $I^{\#}(\alpha)$ to m machines. We schedule the small jobs in $I^{\#}(\alpha)$ assigned to machine M_i as follows. If $TS_{il} = 0$ (which means that in Σ machine M_i processes no small job with delivery time ξ_l) but the replacement procedure has assigned some small jobs with delivery time ξ_l to M_i, then we schedule these small jobs on M_i first, $l = 1, 2, \ldots, \alpha+1$, $i = 1, 2, \ldots, m$. We then schedule all the other small jobs assigned to M_i by Jackson's rule, that is, keeping their order in Σ. (In fact, scheduling all the small jobs assigned to M_i by Jackson's rule can improve the solution. We schedule them in this way only for ease of the subsequent analysis.)

The big jobs in $I^{\#}(\alpha)$ are easily scheduled. We simply replace every big job in Σ by its rounded counterpart in $I^{\#}(\alpha)$.

Let $\Sigma^{\#}$ denote the obtained schedule for $I^{\#}(\alpha)$. It remains only to analyze how the objective value changes, when we move from schedule Σ to schedule $\Sigma^{\#}$. Rounding up the delivery times may increase the objective value by A/α. Rounding up the processing times of the big jobs may increase the objective value by a factor of $1 + 1/\alpha$. Since there are at most $\alpha + 1$ different delivery times in $I^{\#}(\alpha)$, the replacement procedure for the small jobs increases the objective value by $(\alpha+1)\cdot 3A/(\alpha(\alpha+1)) = 3A/\alpha$. Hence, $L^{\#}$, the objective value of $\Sigma^{\#}$, is no more than $(1 + 1/\alpha)\cdot \mathrm{OPT}(I) + 4A/\alpha$. □

Lemma 3. *Let $\Sigma^{\#}$ be a schedule for $I^{\#}(\alpha)$ with objective value $L^{\#}$. Then, there is a schedule for I with objective value $L_{max} \le L^{\#} + A/\alpha$.*

Proof. We assume without loss of generality that, in $\Sigma^{\#}$ on each machine, the jobs are scheduled by Jackson's rule, and thus the small jobs with the same delivery time are processed together in succession on each machine. Let $TS^{\#}_{il}$ be the total processing time of the small jobs with delivery time ξ_l processed on machine M_i in $\Sigma^{\#}$, $l = 1, 2, \ldots, \alpha + 1$, $i = 1, 2, \ldots, m$.

We now explain how to assign all small jobs in I to m machines. Starting from the leaves of tree T, we work our way towards the root in a bottom-up fashion, as in the proof of Lemma 2. Suppose that we are handling vertex $v \in V(T)$. At this point, all the descendants of v except v itself have already been handled and the associated small jobs have been assigned. Let β_l denote the total processing time of the small jobs with delivery time ξ_l in all descendants of v (including v itself) in instance I, and let $\beta^{\#}_l$ denote the total processing time of the corresponding small jobs in $I^{\#}(\alpha)$, $l = 1, 2, \ldots, \alpha + 1$. We have $\beta^{\#}_l \ge \beta_l$.

Let $\mathcal{SJ}_l(v)$ denote the set of small jobs j in I with delivery time $q_j = \xi_l$ and processing set $\mathcal{M}_j = \mathcal{M}(v)$, $l = 1, 2, \ldots, \alpha + 1$. For each machine $M_i \in \mathcal{M}(v)$, if $TS^{\#}_{il} > 0$ (which means that in $\Sigma^{\#}$ machine M_i processes at least one small job with delivery time ξ_l), we assign the small jobs in $\mathcal{SJ}_l(v)$ to M_i until the first time that the total processing time of the small jobs with delivery time ξ_l assigned to M_i exceeds $TS^{\#}_{il}$ (or until there are no unassigned small jobs in $\mathcal{SJ}_l(v)$). Since $\beta^{\#}_l \ge \beta_l$, each job in $\mathcal{SJ}_l(v)$ can be assigned to a machine in $\mathcal{M}(v)$.

After we handle the root of T, we assign all small jobs in I to m machines. The big jobs in I are easily scheduled. We simply replace every rounded big job in $\Sigma^{\#}$ with its original counterpart in I. We then schedule all the jobs assigned to M_i by Jackson's rule, that is, keeping their order in $\Sigma^{\#}$, $i = 1, 2, \ldots, m$.

Let $L_{\max}$ be the objective value of the obtained schedule for I. Since all small jobs in I have processing time at most $A/(\alpha(\alpha + 1))$ and there are at most $\alpha + 1$ different delivery times in I, the replacement procedure for the small jobs may increase the objective value by $(\alpha + 1)\cdot A/(\alpha(\alpha + 1)) = A/\alpha$. The replacement of big jobs will not increase the objective value. Hence, we get $L_{\max} \le L^{\#} + A/\alpha$. □

4. Solving the Rounded Instance Optimally

In this section, we will present an polynomial time optimal algorithm for rounded instance $I^{\#}(\alpha)$ obtained in the preceding section. The basic idea is to generalize the dynamic programming method used in [15] for solving problem $P \mid M_j(\text{nested}) \mid C_{\max}$.

Recall that all jobs in rounded instance $I^{\#}(\alpha)$ have processing times of form $kA/(\alpha^2(\alpha + 1))$, where $k \in \{\alpha, \alpha + 1, \ldots, \alpha^2(\alpha + 1)\}$. We thus can represent subsets of the jobs in $I^{\#}(\alpha)$ as vectors $\vec{\lambda} = (\lambda^1_\alpha, \ldots, \lambda^1_{\alpha^2(\alpha+1)}, \ldots, \lambda^{\alpha+1}_\alpha, \ldots, \lambda^{\alpha+1}_{\alpha^2(\alpha+1)})$, where λ^l_k denotes the number of jobs with delivery time ξ_l and processing time $kA/(\alpha^2(\alpha + 1))$, $l \in \{1, 2, \ldots, \alpha + 1\}$, $k \in \{\alpha, \alpha+1, \ldots, \alpha^2(\alpha+1)\}$. Let Λ be the set of all such vectors. Clearly, there are $O(n^{\alpha^2(\alpha+1)})$ different vectors in Λ. For each vertex $v \in V(T)$, set $\mathcal{J}(v)$ of jobs j in $I^{\#}(\alpha)$ with $\mathcal{M}_j = \mathcal{M}(v)$ is encoded by vector $\vec{\lambda}(v)$.

Let $v \in V(T)$ and $\vec{\lambda} \in \Lambda$, where vector $\vec{\lambda}$ represents a subset of jobs whose processing sets are proper supersets of $\mathcal{M}(v)$. Let $f(v, \vec{\lambda})$ denote the minimum objective value over all the schedules which process the jobs in the descendants of v and the additional jobs in $\vec{\lambda}$ on the machines in $\mathcal{M}(v)$. All jobs in the descendants of v must obey the processing set restrictions, whereas the jobs in $\vec{\lambda}$ can be assigned to any machine in $\mathcal{M}(v)$.

All values $f(v, \vec{\lambda})$ can be computed and tabulated in a bottom-up fashion. If v is a leaf, then $f(v, \vec{\lambda})$ is equal to the objective value of the single machine schedule which processes all the jobs in $\vec{\lambda}(v) + \vec{\lambda}$ by Jackson's rule. If v is a nonleaf vertex, then v has two sons v_1 and v_2, and

$$f(v, \vec{\lambda}) = \min\left\{\max\left\{f(v_1, \vec{\mu}_1), f(v_2, \vec{\mu}_2)\right\} : \vec{\mu}_1, \vec{\mu}_2 \in \Lambda,\ \vec{\mu}_1 + \vec{\mu}_2 = \vec{\lambda}(v) + \vec{\lambda}\right\}. \tag{2}$$

Since there are $O(n^{\alpha^2(\alpha+1)})$ different vectors in Λ and α is a fixed integer that does not depend on the input, all values $f(v, \vec{\lambda})$ can be computed in polynomial time. Finally, since, for root vertex r, there is no job whose processing set is a proper superset of $\mathcal{M}(r)$, the optimal objective value for $I^{\#}(\alpha)$ is achieved by computing $f(r, (0, \ldots, 0))$.

We establish the following theorem.

Theorem 4. *For any fixed integer α, rounded instance $I^{\#}(\alpha)$ can be solved optimally in polynomial time.*

5. The Approximation Scheme

We can now put things together.

Let ε be an arbitrarily small positive constant. Set $\alpha^* = \lceil 11/\varepsilon \rceil$. Given instance I of $P \mid M_j(\text{nested}) \mid L_{\max}$, we get rounded instance $I^{\#}(\alpha^*)$ as described in Section 3. Solve $I^{\#}(\alpha^*)$ optimally as described in Section 4. Denote by $\Sigma^{\#}$ the resulting optimal schedule with objective value $L^{\#}_{\max}$ for $I^{\#}(\alpha^*)$. Transform $\Sigma^{\#}$ into schedule Σ with objective value $L_{\max}$ for instance I as described in the proof of Lemma 3.

By Lemma 2, we have $L^{\#}_{\max} \le (1+1/\alpha^*)\cdot \mathrm{OPT}(I)+4A/\alpha^*$. Together with Lemma 3, we have $L_{\max} \le L^{\#}_{\max} + A/\alpha^* \le (1 + 1/\alpha^*)\cdot \mathrm{OPT}(I) + 5A/\alpha^*$. By inequality (1), we have $A \le 2\mathrm{OPT}(I)$. It follows that $L_{\max} \le (1 + 11/\alpha^*)\cdot \mathrm{OPT}(I) \le (1 + \varepsilon)\cdot \mathrm{OPT}(I)$. We have the following theorem.

Theorem 5. *Problem* $P \mid M_j(nested) \mid L_{max}$ *admits a polynomial time approximation scheme.*

6. Concluding Remarks

In this paper, we initiated the study of scheduling parallel machines with job delivery times and nested processing set restrictions. The objective is to minimize the maximum delivery completion time. For this strong NP-hard problem, we presented a simple and fast 2-approximation algorithm. We also presented a polynomial time approximation scheme. A natural open problem is to design fast algorithms with approximation ratios better than 2. It would also be interesting to study the problem with job release times.

Competing Interests

The author confirms that this article content has no competing interests.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (nos. 61373079, 61472227, 61272244, and 61672327), Key Project of Chinese Ministry of Education (no. 212101), and Shandong Provincial Natural Science Foundation of China (no. ZR2013FM015).

References

[1] J. Y.-T. Leung and C.-L. Li, "Scheduling with processing set restrictions: a survey," *International Journal of Production Economics*, vol. 116, no. 2, pp. 251–262, 2008.

[2] J. Y. Leung and C. Li, "Scheduling with processing set restrictions: a literature update," *International Journal of Production Economics*, vol. 175, pp. 1–11, 2016.

[3] H. Kise, T. Ibaraki, and H. Mine, "Performance analysis of six approximation algorithms for the one-machine maximum lateness scheduling problem with ready times," *Journal of the Operations Research Society of Japan*, vol. 22, no. 3, pp. 205–224, 1979.

[4] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, vol. 174, W. H. Freeman, New York, NY, USA, 1979.

[6] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Courier Dover, 1998.

[7] G. J. Woeginger, "Heuristics for parallel machine scheduling with delivery times," *Acta Informatica*, vol. 31, no. 6, pp. 503–512, 1994.

[8] C. N. Potts, "Analysis of a heuristic for one machine sequencing with release dates and delivery times," *Operations Research*, vol. 28, no. 6, pp. 1436–1441, 1980.

[9] L. A. Hall and D. B. Shmoys, "Jackson's rule for single-machine scheduling: making a good heuristic better," *Mathematics of Operations Research*, vol. 17, no. 1, pp. 22–35, 1992.

[10] L. A. Hall and D. B. Shmoys, "Approximation schemes for constrained scheduling problems," in *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pp. 134–139, 1989.

[11] M. Mastrolilli, "Efficient approximation schemes for scheduling problems with release dates and delivery times," *Journal of Scheduling*, vol. 6, no. 6, pp. 521–531, 2003.

[12] C. A. Glass and H. Kellerer, "Parallel machine scheduling with job assignment restrictions," *Naval Research Logistics*, vol. 54, no. 3, pp. 250–257, 2007.

[13] Y. Huo and J. Y.-T. Leung, "Parallel machine scheduling with nested processing set restrictions," *European Journal of Operational Research*, vol. 204, no. 2, pp. 229–236, 2010.

[14] Y. Huo and J. Y.-T. Leung, "Fast approximation algorithms for job scheduling with processing set restrictions," *Theoretical Computer Science*, vol. 411, no. 44–46, pp. 3947–3955, 2010.

[15] G. Muratore, U. M. Schwarz, and G. J. Woeginger, "Parallel machine scheduling with nested job assignment restrictions," *Operations Research Letters*, vol. 38, no. 1, pp. 47–50, 2010.

[16] L. Epstein and A. Levin, "Scheduling with processing set restrictions: PTAS results for several variants," *International Journal of Production Economics*, vol. 133, no. 2, pp. 586–595, 2011.

[17] R. L. Graham, "Bounds for certain multiprocessing anomalies," *Bell System Technical Journal*, vol. 45, no. 9, pp. 1563–1581, 1966.

[18] J. R. Jackson, "Scheduling a production line to minimize maximum tardiness," DTIC Document, 1955.