Theoretical Computer Science 803 (2020) 144–151

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Open-shop scheduling for unit jobs under precedence constraints [☆]

Yong Chen [a,b], Randy Goebel [b], Guohui Lin [b,*], Bing Su [c], An Zhang [a,b,**]

[a] *Department of Mathematics, Hangzhou Dianzi University, Hangzhou, China*
[b] *Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E8, Canada*
[c] *School of Economics and Management, Xi'an Technological University, Xi'an, China*

ARTICLE INFO

Article history:
Received 24 April 2019
Received in revised form 14 August 2019
Accepted 25 September 2019
Available online 1 October 2019

Keywords:
Open-shop scheduling
Precedence constraint
Directed acyclic graph
Approximation algorithm

ABSTRACT

We study open-shop scheduling for unit jobs under precedence constraints, where if one job precedes another job then it has to be finished before the other job can start to be processed. For the three-machine open-shop to minimize the makespan, we first present a simple 5/3-approximation algorithm based on a partition of the job set into agreeable layers using the natural layered representation of the precedence graph, which is directed acyclic. We then show a greedy algorithm to reduce the number of singleton-job layers, resulting in an improved partition, which leads to a 4/3-approximation algorithm. Both approximation algorithms apply to the general m-machine open-shops too.

1. Introduction

Machine scheduling with precedence constraints on the jobs has received much attention in the past few decades, and several algorithmic techniques such as the *critical path method* and the *project evaluation and review technique* [9] have been developed from the line of research. Job precedence constraints are common in construction and manufacturing industries, for example, the bicycle assembly problem is an earliest precedence constrained scheduling application introduced by Graham [7].

Precedence constraints describe the job processing order in a way that one or more jobs have to be finished before another job is allowed to start its processing. Such relationships together are usually represented as a *directed acyclic graph* (DAG) $G = (V, E)$, called the *precedence graph*, where V is the set of jobs and a directed edge $(v_i, v_j) \in E$ states that the job v_i precedes the job v_j, that is, v_i needs to be finished before v_j can start to be processed.

In this paper, we discuss the open-shop scheduling environment and use Om to denote the m-machine open-shop for some constant m, and O to denote the open-shop in which the number of machines is part of the input. In either Om or O, every job needs to be processed non-preemptively by each machine, in any machine order, and it is *finished* (or said *completed*) when it has been processed by all the machines. Note that the usual scheduling rules apply to a feasible schedule, that is, at any time point, a job can be processed by at most one machine and each machine can be processing at most one job. The *makespan* of the schedule is the maximum job completion time. The open-shop scheduling to minimize the

[☆] An extended abstract appears in the *Proceedings of COCOA 2018*. LNCS 11346, pages 329–340.
[*] Corresponding author.
[**] Corresponding author at: Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E8, Canada.
E-mail addresses: chenyong@hdu.edu.cn (Y. Chen), rgoebel@ualberta.ca (R. Goebel), guohui@ualberta.ca (G. Lin), subing684@sohu.com (B. Su), anzhang@hdu.edu.cn (A. Zhang).

https://doi.org/10.1016/j.tcs.2019.09.046
0304-3975/© 2019 Elsevier B.V. All rights reserved.

makespan is denoted as $Om \mid\mid C_{\max}$ or $O \mid\mid C_{\max}$, which has received much study [6,15,11,12,9]. In particular, $O2 \mid\mid C_{\max}$ is solvable in $O(n)$-time, where n denotes the number of jobs [6,9]; $Om \mid\mid C_{\max}$ becomes weakly NP-hard when $m \geq 3$ [6] but admits a polynomial-time approximation scheme (PTAS) [11,12]; $O \mid\mid C_{\max}$ is strongly NP-hard and cannot be approximated within 1.25 [15].

Open-shop scheduling with precedence constraints, denoted as $Om \mid prec \mid C_{\max}$ or $O \mid prec \mid C_{\max}$, is more general (and thus more difficult) than its classical counterparts, which can be considered as scheduling without precedence constraints. Several special classes of precedence graphs have been investigated in the literature. For instance, if every job has at most one predecessor and at most one successor, the precedence graph is referred to as *chains*. If every job has at most one successor (one predecessor, respectively), the precedence graph is referred to as an *intree* (an *outtree*, respectively). The fact that the precedence graph belongs to a particular class may change the computational complexity of the scheduling problem. In general, one can expect that the precedence constraints increase the problem complexity. For example, $O2 \mid chains \mid C_{\max}$ becomes NP-hard [13], as opposed to the polynomial-time solvable $O2 \mid\mid C_{\max}$. For more complexity results on precedence constrained scheduling, the interested readers can refer to the surveys by Lenstra and Rinnooy Kan [8] and by Prot and Bellenguez-Morinea [10].

Unlike most past results which are on computational complexity, in this paper we aim to develop algorithmic positive results for open-shop scheduling with precedence constraints, from the approximation algorithm perspective. We focus on the problems restricted to unit jobs, that is, the jobs have the same processing times on all the machines (*i.e.*, the processing time of the job J_i on the machine M_j is $p_{ij} = 1$, for every couple i and j); most of these problems remain NP-hard, or their complexity are still open. To name a few, for an arbitrary precedence graph, the problem $O \mid p_{ij} = 1, prec \mid C_{\max}$ was shown to be strongly NP-hard by Timkovsky [14]; when the precedence graph is an out-tree, then the problem $O \mid p_{ij} = 1, outtree \mid C_{\max}$ becomes polynomially solvable [1]; for a more general objective of minimizing the maximum lateness, Timkovsky proved that $O \mid p_{ij} = 1, outtree \mid L_{\max}$ is weakly NP-hard [14], while the problem $O \mid p_{ij} = 1, intree \mid L_{\max}$ is polynomial solvable [3,2]. We note that, however, there are polynomial time algorithms for $O2 \mid p_{ij} = 1, prec \mid L_{\max}$, even if the jobs have different release times [3,2].

The problem we study in this paper is the m-machine open-shop for unit jobs under arbitrary precedence constraints, $Om \mid p_{ij} = 1, prec \mid C_{\max}$, where $m \geq 3$. To our surprise, for this fundamental problem in scheduling theory, there is no known computational complexity result in the literature. In fact, even when $m = 3$, whether or not $O3 \mid p_{ij} = 1, prec \mid C_{\max}$ is NP-hard is an open question explicitly listed in the websites maintained by Brucker and Knust [4] and Dürr [5], and in the survey paper by Prot and Bellenguez-Morinea [10].

We first introduce a natural layered representation for the precedence graph in Section 2, based on which we can construct a partition of the job set into agreeable subsets. We then construct a schedule using the partition and show that it is a 5/3-approximation algorithm for the problem $O3 \mid p_{ij} = 1, prec \mid C_{\max}$. In Section 3, we propose a greedy algorithm to reduce the number of singleton-job subsets in the earlier partition, resulting in an improved partition, which leads to a 4/3-approximation algorithm. We also show that both approximation algorithms apply to the general m-machine open-shops.

2. Preliminaries

We study the problem $O3 \mid p_{ij} = 1, prec \mid C_{\max}$, in which the unit jobs should be processed under the given precedence constraints. These precedence constraints are described as a directed acyclic graph (DAG), the *precedence graph*, in which a vertex corresponds to a job and a directed edge represents a precedence relationship between a pair of jobs. In the rest of the paper, we use a job and a vertex interchangeably. Due to all jobs having unit processing times, we assume without loss of generality that in any feasible schedule the starting processing time of every job is an integer.

Let $V = \{v_1, v_2, \ldots, v_n\}$ be the given set of unit jobs. If v_i precedes v_j, that is, we can start processing the job v_j only if the job v_i is finished by the three-machine openshop $O3$, then there is a directed path beginning from v_i and ending at v_j. Such a directed path is a directed edge (v_i, v_j) in the simplest case, in the DAG precedence graph $G = (V, E)$.

A subset $X \subseteq V$ of jobs is *agreeable* if none of the jobs in X precedes another job in X. In particular, two jobs are *agreeable* if none of them precedes the other, and thus they can be processed concurrently on different machines in a feasible schedule.

Lemma 2.1. *An agreeable subset $X \subseteq V$ of jobs can be processed by the three-machine openshop $O3$ in $|X|$ units of time if $|X| \geq 3$, or in 3 units of time if $|X| = 1, 2$.*

Proof. Let the jobs of X be $v_1, v_2, \ldots, v_k$. When $k = 1$, at any time point T, v_1 can be processed on the first machine M_1 (the second machine M_2, the third machine M_3, respectively) starting at T ($T + 1$, $T + 2$, respectively), and thus finished within 3 units of time.

When $k = 2$, at any time point T, v_1 can be processed on the first machine M_1 (the second machine M_2, the third machine M_3, respectively) starting at T ($T + 1$, $T + 2$, respectively); v_2 can be processed on the third machine M_3 (the first machine M_1, the second machine M_2, respectively) starting at T ($T + 1$, $T + 2$, respectively). Thus both of them are finished within 3 units of time.

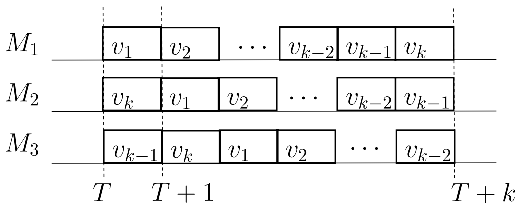

Fig. 1. A sub-schedule to process an agreeable subset $X \subseteq V$ of jobs in $|X|$ units of time when $k = |X| \geq 3$.

When $k \geq 3$, at any time point T, for $j = 1, 2, \ldots, k-2$, v_j can be processed on the first machine M_1 (the second machine M_2, the third machine M_3, respectively) starting at $T + j - 1$ ($T + j$, $T + j + 1$, respectively); v_{k-1} can be processed on the third machine M_3 (the first machine M_1, the second machine M_2, respectively) starting at T ($T + k - 2$, $T + k - 1$, respectively); v_k can be processed on the second machine M_2 (the third machine M_3, the first machine M_1, respectively) starting at T ($T + 1$, $T + k - 1$, respectively). See Fig. 1 for an illustration. Thus all of them are finished within k units of time. $\square$

Given two disjoint agreeable subsets X_1 and X_2, if a job in X_1 precedes a job in X_2, then we say X_1 *precedes* X_2. A collection of mutual disjoint agreeable subsets is *acyclic* if the precedence relations among the subsets do not contain any cycle. A subset of k jobs is called a k-subset, for $k = 1, 2, \ldots$. For simplicity, a 1-subset is also called a *singleton*.

Corollary 2.2. *Let $\mathcal{C}$ be an acyclic partition of V into agreeable subsets, in which there are b 2-subsets and c singletons. Then a schedule π can be constructed to achieve the makespan $C^{\pi}_{\max} = n + b + 2c$, where $n = |V|$.*

Proof. Using Lemma 2.1, all the $n - 2b - c$ jobs outside of those 2-subsets and singletons can be finished in $n - 2b - c$ units of time, and each 2-subset and each singleton can be finished in 3 units of time, respectively. Putting them together, we have a schedule π of makespan $C^{\pi}_{\max} = (n - 2b - c) + 3b + 3c = n + b + 2c$. $\square$

By Corollary 2.2, we wish to solve the problem $O3 \mid p_{ij} = 1, prec \mid C_{\max}$ by partitioning the jobs into acyclic agreeable subsets such that the quantity $b + 2c$ is minimized. Our main contribution is an algorithm that produces an acyclic partition achieving a number of singletons no more than the number of isolated jobs (to be defined) in the optimal schedule. That is, we manage to "sort of" minimize c.

In the rest of the section, we introduce a representation for the DAGs which is used in our algorithm design and analysis.

2.1. A DAG representation

Let $G = (V, E)$ be the precedence graph describing all the given precedence constraints, where a directed path from v_i to v_j suggests that the job v_i precedes the job v_j (that is, v_j cannot be processed unless v_i is finished by the three-machine openshop). Through out the paper, we let $n = |V|$ and $m = |E|$.

If $(v_i, v_j) \in E$ and there exists a path from v_i to v_j not involving the edge (v_i, v_j), then we call (v_i, v_j) a *redundant* edge, in the sense that the precedence constraint between every pair of jobs is still there after we remove the edge (v_i, v_j) from the graph. We may thus simplify the graph G by removing all redundant edges, which can be executed in $O(m)$ time by a *breadth-first-search* (BFS). Afterwards, for each edge $(v_i, v_j) \in E$, we call v_i a *parent* of v_j and v_j a *child* of v_i. Note that a job can have multiple parents, and multiple children as well; see for example Fig. 2.

In the following layered representation of the graph $G = (V, E)$, each job will be associated with a level (a positive integer). The first layer consists of all the jobs with in-degree 0, and these jobs are the *level*-1 jobs. Iteratively, after the level-ℓ jobs are determined, they and the edges (these are out-edges) incident at them are removed from the graph; then the $(\ell + 1)$-st layer consists of all the jobs with in-degree 0 in the remainder graph, and these jobs are the *level*-$(\ell + 1)$ jobs. The process terminates when all the jobs of the original graph G have been partitioned into their respective layers. We assume that there are $\ell_{\max}$ layers in total. The entire layer partitioning process is executed in $O(m)$ time. In the sequel, without loss of generality, a DAG $G = (V, E)$ is always represented in this way, in which every job is associated with a level and L_i denotes the subset of all the level-i jobs, for $i = 1, 2, \ldots, \ell_{\max}$. See Fig. 2 for an illustration.

Lemma 2.3. *Given a DAG $G = (V, E)$, L_i is agreeable for every i, and a level-i job has at least one level-$(i - 1)$ parent for every $i \geq 2$.*

Proof. The lemma holds by how the layers are constructed. $\square$

Lemma 2.4. *Given a DAG $G = (V, E)$, the partition $\mathcal{C} = \{L_1, L_2, \ldots, L_{\ell_{\max}}\}$ is an acyclic collection of agreeable subsets.*

Proof. The lemma holds by how the layers are constructed, and by Lemma 2.3, L_i precedes L_j if and only if $i < j$. $\square$

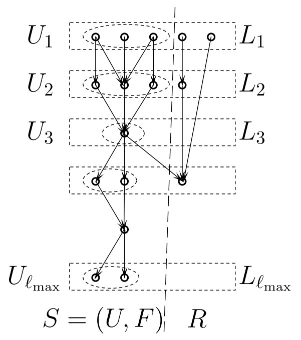

Fig. 2. A layered representation of the precedence graph $G=(V,E)$, in which there are $\ell_{\max}$ layers (each as a dashed rectangle) in total, denoted as $L_1, L_2, \ldots, L_{\ell_{\max}}$. U denotes the subset of all the vertices on the longest paths in G, $U_i = L_i \cap U$, for $i = 1, 2, \ldots, \ell_{\max}$ (each as a dashed oval), $S=(U,F)$ denotes the induced subgraph $G[U]$ by U, $R = V - U$ and H denotes the induced subgraph $G[R]$ by R.

Lemma 2.5. *Given a DAG $G=(V,E)$, the minimum makespan $C^*_{\max} \geq \max\{n, 3\ell_{\max}\}$.*

Proof. Since we are dealing with unit jobs, $C^*_{\max} \geq n$.

Select one job v_i from L_i, for every i, such that v_i is a child of the job v_{i-1}. (That is, v_1-v_2-...-$v_{\ell_{\max}}$ is a longest path in G.) One clearly sees that in any feasible schedule, the job v_i starts processing after the job v_{i-1} is finished by the three-machine openshop; the makespan of the schedule is thus at least $3\ell_{\max}$. This proves the lemma. □

Theorem 2.6. *A schedule π can be constructed from the partition $\mathcal{C} = \{L_1, L_2, \ldots, L_{\ell_{\max}}\}$ to achieve the makespan $C^\pi_{\max} \leq \frac{5}{3} C^*_{\max}$.*

Proof. Let b and c denote the number of 2-subsets and the number of singletons among $L_1, L_2, \ldots, L_{\ell_{\max}}$, respectively. By Corollary 2.2 a schedule π can be constructed from $\mathcal{C}$ to achieve the makespan $C^\pi_{\max} = n + b + 2c$.

Using the trivial bound $\ell_{\max} \geq b + c$ in Lemma 2.5, we have $C^*_{\max} \geq \max\{n, 3(b+c)\}$. It follows that

$$C^\pi_{\max} = n + b + 2c \leq C^*_{\max} + \frac{2}{3} C^*_{\max} = \frac{5}{3} C^*_{\max}.$$

This proves the theorem. □

3. A 4/3-approximation algorithm for $O3 \mid prec, p_{ij} = 1 \mid C_{\max}$

Clearly, from the layered representation of the graph $G=(V,E)$, we see that every longest path in G begins with a level-1 job and ends at a level-$\ell_{\max}$ job, and it passes through every intermediate layer. That is, every longest path contains exactly $\ell_{\max}$ jobs (and $\ell_{\max} - 1$ edges). Let U denote the subset of all the jobs on the longest paths and F denote the subset of edges inherited by U (*i.e.*, $F = E[U]$). We call $S=(U,F)$ the *spine* of the graph $G=(V,E)$. Let $R = V - U$ denote the remaining subset of jobs and $H = G[R]$ denote the subgraph of G induced by R. See Fig. 2 for an illustration.

We define a connected component in a DAG in the usual way by ignoring the direction of the edges. If the spine $S=(U,F)$ has more than one connected component, then we can safely conclude that every layer of the graph $G=(V,E)$ contains at least two jobs (at least one job from each component), that is, $|L_i| \geq 2$ for $i = 1, 2, \ldots, \ell_{\max}$. Recall that our goal is to partition all the jobs into acyclic agreeable subsets to minimize the number of singletons. We call such partitions with the minimum number of singletons the *good partitions* or *good collections* of acyclic agreeable subsets. We assume in the rest of the paper that the spine $S=(U,F)$ of the input graph $G=(V,E)$ is connected (*i.e.*, S contains only one connected component) and there are singleton layers in $S=(U,F)$, as otherwise we trivially achieve a good partition without any singletons. Let U_i denote the subset of level-i jobs of U, that is, $U_i = L_i \cap U$, for $i = 1, 2, \ldots, \ell_{\max}$. If $|U_i| = 1$, then the job in U_i, denoted as s_i, is called a *singleton job* in U.

Lemma 3.1. *Given a DAG $G=(V,E)$ and its spine $S=(U,F)$, any acyclic partition of agreeable subsets contains at least $\ell_{\max}$ subsets.*

Proof. Select one job u_i from U_i, for every i, such that u_i is a child of the job u_{i-1}. (Again, u_1-u_2-...-$u_{\ell_{\max}}$ is a longest path in G.) One clearly sees that in an acyclic partition of agreeable subsets, the jobs u_i and u_j do not belong to any common subset when $i \neq j$. This suggests that there are at least $\ell_{\max}$ subsets in the partition, and thus proves the lemma. □

Lemma 3.2. *Given a DAG $G=(V,E)$ and its spine $S=(U,F)$, a singleton job in U cannot be processed concurrently with any other job in U in any feasible schedule.*

Proof. The lemma holds because the singleton job is not agreeable with any other job in U. □

```
The $j$-th iteration of the algorithm Approx ($j = 2, 3, \dots, k$):

  1. for $i = i_{k+2-j} - 1, i_{k+2-j} - 2, \dots, i_{k+1-j} + 1$,
     1.1. set $D_i = L_i$;
     1.2. remove the jobs in $L_i - U_i$ from $R$;
  2. for $i = i_{k+1-j}$, if $|L_{i_{k+1-j}}| \geq 2$,
     2.1. set $D_{i_{k+1-j}} = L_{i_{k+1-j}}$;
     2.2. remove the jobs in $L_{i_{k+1-j}} - s_{i_{k+1-j}}$ from $R$;
     2.3. end the iteration;
  3. if exists $v_i \in L_i - U_i$, with maximum $i$ using Lemma 3.3, agreeable with $s_{i_{k+1-j}}$,
     3.1. set $D_{i_{k+1-j}} = \{s_{i_{k+1-j}}, v_i\}$;
     3.3. remove the job $v_i$ from $R$;
     3.3. end the iteration;
  4. 4.1. set $D_{i_{k+1-j}} = \{s_{i_{k+1-j}}\}$;
     4.2. end the iteration.
```

Fig. 3. A high-level description of a typical j-th iteration of the algorithm Approx.

Assume there are in total k singleton jobs in U, which are $s_{i_1}, s_{i_2}, \dots, s_{i_k}$, that is, $U_{i_j} = \{s_{i_j}\}$, where $1 \leq i_1 < i_2 < \dots < i_k \leq \ell_{\max}$.

Lemma 3.3. *Given a DAG $G = (V, E)$ and its spine $S = (U, F)$, for a singleton job $s_{i_j} \in U$, if there is a job in $V - U$ agreeable with s_{i_j}, then there is a level-i job in $L_i - U_i$ with $i \leq i_j$ which is agreeable with s_{i_j}.*

Proof. Let $v_i \in L_i - U_i$ be a level-i job agreeable with s_{i_j}. If $i \leq i_j$, then we are done, and thus we assume below that $i > i_j$.

Since v_i is agreeable with s_{i_j}, none of the jobs in U_{i-1} can be a parent of v_i by precedence transitivity; it follows from Lemma 2.3 that v_i has a parent $v_{i-1} \in L_{i-1} - U_{i-1}$. If $i - 1 > i_j$, v_{i-1} must also be agreeable with s_{i_j} again by precedence transitivity, and we may repeat the above argument to conclude that there is a job $v_{i_j} \in L_{i_j} - s_{i_j}$ which is a predecessor of v_i. Since both s_{i_j} and v_{i_j} are in L_{i_j}, they are agreeable by Lemma 2.3. We thus have proved the lemma. □

3.1. The algorithm

We have shown in Theorem 2.6 that we can construct a schedule π from the partition $\mathcal{C} = \{L_1, L_2, \dots, L_{\ell_{\max}}\}$ to achieve the makespan $C^{\pi}_{\max} \leq \frac{5}{3} C^*_{\max}$, suggesting that the $O3 \mid prec, p_{ij} = 1 \mid C_{\max}$ problem admits a linear time 5/3-approximation algorithm. In this section, we present an improved 4/3-approximation algorithm.

For each singleton job s_{i_j} of U, our algorithm searches for a job in $V - U$ which is agreeable with s_{i_j} such that they can be processed concurrently, and the search is mostly based on the above Lemma 3.3. The algorithm is greedy and iterative, and is denoted as Approx.

Recall that there are in total k singleton jobs in U, which are $s_{i_1}, s_{i_2}, \dots, s_{i_k}$ (that is, $U_{i_j} = \{s_{i_j}\}$), with $1 \leq i_1 < i_2 < \dots < i_k \leq \ell_{\max}$. There are $k + 1$ iterations in the algorithm Approx, which together construct an acyclic partition $\mathcal{D} = \{D_{\ell_{\max}}, D_{\ell_{\max}-1}, \dots, D_2, D_1\}$. Recall also that we initialize $R = V - U$.

In the first iteration, sequentially for $i = \ell_{\max}, \ell_{\max} - 1, \dots, i_k + 1$, we simply let $D_i = L_i$ and remove the jobs in $L_i - U_i$ from R. For $i = i_k$, if $|L_{i_k}| \geq 2$, then we let $D_{i_k} = L_{i_k}$ and remove the jobs in $L_{i_k} - s_{i_k}$ from R. Otherwise (that is, $L_{i_k} = U_{i_k} = \{s_{i_k}\}$), among all the jobs in R, we pick one job that is agreeable with s_{i_k} (*i.e.*, not a predecessor of s_{i_k}) and has the maximum level, using Lemma 3.3. Assume this job is $v_i \in L_i - U_i$ such that $i < i_k$. We let $D_{i_k} = \{s_{i_k}, v_i\}$ and remove the job v_i from R. If no job in R is agreeable with s_{i_k}, then we let $D_{i_k} = \{s_{i_k}\}$ and say that s_{i_k} *remains as a singleton job* in the partition $\mathcal{D}$. This ends the first iteration.

In general, in the j-th iteration ($j = 2, 3, \dots, k$), sequentially for $i = i_{k+2-j} - 1, i_{k+2-j} - 2, \dots, i_{k+1-j} + 1$, we simply let $D_i = L_i$ and remove the jobs in $L_i - U_i$ from R. We remark that here the set L_i might not be the original L_i, since some of its jobs might be picked in earlier iterations and thus have been removed. Nevertheless, since $|U_i| \geq 2$, we conclude that $|D_i| \geq 2$ too. For $i = i_{k+1-j}$, if $|L_{i_{k+1-j}}| \geq 2$, then we let $D_{i_{k+1-j}} = L_{i_{k+1-j}}$ and remove the jobs in $L_{i_{k+1-j}} - s_{i_{k+1-j}}$ from R. Otherwise (that is, $L_{i_{k+1-j}} = U_{i_{k+1-j}} = \{s_{i_{k+1-j}}\}$), among all the jobs in R, we pick one job that is agreeable with $s_{i_{k+1-j}}$ (*i.e.*, not a predecessor of $s_{i_{k+1-j}}$) and has the maximum level, using Lemma 3.3. Assume this job is $v_i \in L_i - U_i$ such that $i < i_{k+1-j}$. We let $D_{i_{k+1-j}} = \{s_{i_{k+1-j}}, v_i\}$ and remove the job v_i from R. If no job in R is agreeable with $s_{i_{k+1-j}}$, then we let $D_{i_{k+1-j}} = \{s_{i_{k+1-j}}\}$ and say that $s_{i_{k+1-j}}$ *remains as a singleton job* in the partition $\mathcal{D}$. This ends the j-th iteration.

A high-level description of such a typical j-th iteration of the algorithm Approx is depicted in Fig. 3, in which the detailed steps of operations are listed.

In the last (that is, the $(k+1)$-st) iteration, sequentially for $i = i_1 - 1, i_1 - 2, \dots, 2, 1$, we simply let $D_i = L_i$ and remove the jobs in $L_i - U_i$ from R. Again, we know that here the set L_i might not be the original L_i, since some of its jobs might be picked in earlier iterations. Nevertheless, since $|U_i| \geq 2$, we conclude that $|D_i| \geq 2$ too. This ends the last iteration and the construction of $\mathcal{D}$ is complete. See Fig. 4 for an illustration on $\mathcal{D}$ achieved on the graph $G = (V, E)$ shown in Fig. 2.

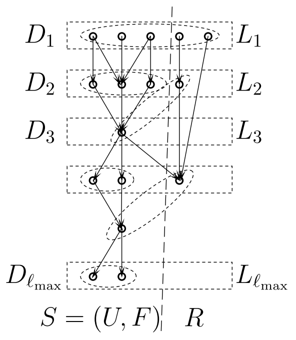

Fig. 4. An illustration on the acyclic partition $\mathcal{D} = \{D_{\ell_{\max}}, D_{\ell_{\max}-1}, \ldots, D_2, D_1\}$ achieved on the precedence graph $G = (V, E)$ shown in Fig. 2. The $\ell_{\max}$ layers $L_1, L_2, \ldots, L_{\ell_{\max}}$ are shown as dashed rectangles and each subset D_i is shown as a dashed oval.

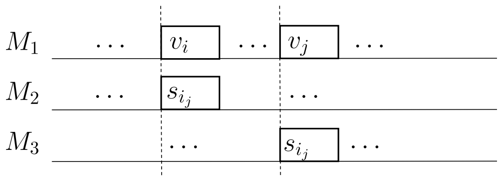

Fig. 5. In the optimal schedule π^*, at most two possible jobs, shown as v_i and v_j, of $V - U$ can be associated with a singleton job s_{i_j} of U.

3.2. The performance analysis

We will show that the schedule π constructed from the above partition $\mathcal{D} = \{D_1, D_2, \ldots, D_{\ell_{\max}}\}$ has a makespan $C^{\pi}_{\max} \le \frac{4}{3} C^*_{\max}$.

Lemma 3.4. *Given a DAG $G = (V, E)$, its layered representation $\{L_1, L_2, \ldots, L_{\max}\}$, and its spine $S = (U, F)$, the partition $\mathcal{D} = \{D_1, D_2, \ldots, D_{\ell_{\max}}\}$ achieved by the algorithm* Approx *is acyclic.*

Proof. We prove the acyclicity by contradiction.

Since $U_i \subseteq D_i$ for all $i = 1, 2, \ldots, \ell_{\max}$, D_i precedes D_{i+1} for all $i = 1, 2, \ldots, \ell_{\max} - 1$. Suppose to the contrary that $\mathcal{D}$ is cyclic, then there are levels i and i' such that $i > i'$ and D_i precedes $D_{i'}$. Note that D_i ($D_{i'}$, respectively) consists of a subset of jobs in L_i ($L_{i'}$, respectively), and possibly a job v_r with a smaller level $r \le i - 1$ if $i = i_j$ for some j (possibly a job $v_{r'}$ with a smaller level $r' \le \min\{r, i' - 1\}$ if $i' = i_{j'}$ for some j', respectively). It follows from Lemma 2.3 that D_i has to contain a job v_r with a smaller level $r \le i - 1$, $i = i_j$ for some j, and $D_{i_j} = \{s_{i_j}, v_r\}$. Moreover, the job v_r precedes a job in $D_{i'}$, denoted as v_t of level t. This leads to $r < t \le i'$. If v_t is agreeable with s_{i_j}, then by the algorithm description the job v_t, instead of v_r, should be picked into D_{i_j}, a contradiction. Hence v_t precedes s_{i_j}, which by transitivity implies that v_r precedes s_{i_j} too, and thus v_r shouldn't be picked into D_{i_j}, again a contradiction. These contradictions together prove that for any levels $i > i'$, D_i doesn't precede $D_{i'}$. □

Theorem 3.5. *The schedule π constructed from the partition $\mathcal{D} = \{D_1, D_2, \ldots, D_{\ell_{\max}}\}$ has a makespan $C^{\pi}_{\max} \le \frac{4}{3} C^*_{\max}$.*

Proof. Lemma 3.4 states that the partition $\mathcal{D}$ is acyclic. We use Corollary 2.2 to construct a schedule π with its makespan $C^{\pi}_{\max} = n + b + 2c$, where b and c denote the numbers of 2-subsets and singletons in $\mathcal{D}$.

We next estimate this makespan. The key in the estimation is to show the following Eq. (1), that the number of singletons, c, in the acyclic partition $\mathcal{D}$ is no greater than a corresponding value, c^*, in the optimal schedule.

Consider an optimal schedule π^* that achieves the minimum makespan $C^*_{\max}$, and assume without loss of generality that the makespan is achieved at the first machine M_1. For a singleton job s_{i_j} of U, Lemma 3.2 states that it cannot be processed concurrently with any other job of U in π^*. Therefore, there are at most two distinct jobs in $V - U$, such that for each of them, when the machine M_1 is processing it, one of the other two machines M_2 and M_3 is processing s_{i_j}. See for an illustration in Fig. 5, where v_i and v_j are these two possible jobs. We say that these two jobs in $V - U$ are *associated* with the singleton job s_{i_j}. It is important to note that a job in $V - U$ associated with a singleton job cannot be associated with another singleton job, for otherwise the two singleton jobs were processed concurrently in π^* (contradicting Lemma 3.2). Also, for $j < j'$, since s_{i_j} precedes $s_{i_{j'}}$, no associated job of $s_{i_{j'}}$ can precede any associated job of s_{i_j}.

When there is one or two jobs in $V - U$ associated with the singleton job s_{i_j}, we pick one randomly. If the picked job has a level less than or equal to i_j, then we use t_{i_j} to denote it. If the picked job has a level greater than i_j, then we apply

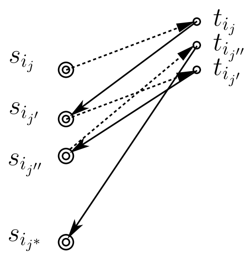

Fig. 6. Finding the unique target isolated job $s_{i_{j^*}}$ in the optimal schedule π^* for every source isolated job s_{i_j} in the schedule π. In the figure, t_{i_j} ($t_{i_{j'}}$, $t_{i_{j''}}$, respectively) is the picked job associated with s_{i_j} ($s_{i_{j'}}$, $s_{i_{j''}}$, respectively) in π^*, shown by dashed arrows, and is picked by the algorithm Approx to accompany $s_{i_{j'}}$ ($s_{i_{j''}}$, $s_{i_{j^*}}$, respectively) in π, shown by solid arrows.

Lemma 3.3 to locate one of its predecessor jobs with level exactly i_j and use t_{i_j} to denote this predecessor. Either way, the level of the associated job t_{i_j} is no greater than i_j. We claim that all these t_{i_j}'s, if exist, are distinct. The claim can be proven by induction on x that "the first x t_{i_j}'s are distinct", which holds trivially when $x = 1$. Without loss of generality assume that $t_{i_1}, t_{i_2}, \ldots, t_{i_x}$ are distinct. Recall that t_{i_j} is a predecessor of the picked job, or itself, associated with the singleton s_{i_j}, for all $j = 1, 2, \ldots, x$, and the picked job associated with the singleton $s_{i_{x+1}}$ does not precede any associated job of s_{i_j}, for all $j = 1, 2, \ldots, x$. Hence, if $t_{i_{x+1}}$ is the picked associated job itself and it collides into t_{i_j}, for some $1 \le j \le x$, then $t_{i_{x+1}}$ precedes the picked job associated with s_{i_j}, a contradiction. If $t_{i_{x+1}}$ is a predecessor of the picked associated job, then its level is exactly i_{x+1}, which is strictly less than the level of t_{i_j}, for all $1 \le j \le x$; that is, $t_{i_{x+1}}$ is distinct from t_{i_j}'s, for all $j = 1, 2, \ldots, x$. This proves the claim.

If there is no job in $V - U$ associated with the singleton job s_{i_j}, we say s_{i_j} is *isolated* in π^*.

Recall that in the partition $\mathcal{D}$, when $i \notin \{i_1, i_2, \ldots, i_k\}$, $|D_i| \ge 2$. If $|D_{i_j}| = 1$, that is, $D_{i_j} = \{s_{i_j}\}$, then we say s_{i_j} is *isolated* in the schedule π constructed from $\mathcal{D}$. We prove in the following the most important property that the number of isolated jobs in π is not greater than the number of isolated jobs in π^* (though the two meanings of "isolated" are different), by constructing an injective function.

Assume s_{i_j} is isolated in π, called a *source*. We find a path from s_{i_j} to a *target* isolated job in π^* as follows: If s_{i_j} is isolated in π^*, then s_{i_j} is the target and the path has length 0. If s_{i_j} is not isolated in π^*, that is, we have a job t_{i_j} associated with s_{i_j}, then t_{i_j} should have been picked by the algorithm Approx in an earlier iteration, since otherwise in the $(k+1-j)$-th iteration the singleton job s_{i_j} wouldn't be left alone in the set D_{i_j}. Therefore, we identify another singleton job $s_{i_{j'}}$, where $j' > j$, which is not isolated in π because in the $(k+1-j')$-th iteration the algorithm Approx picked up t_{i_j} to accompany the singleton job $s_{i_{j'}}$. Our path extends from s_{i_j} to $s_{i_{j'}}$. If $s_{i_{j'}}$ happens to be isolated in π^*, then we find the target $s_{i_{j'}}$ and our path ends; otherwise, we continue to use its associated job $t_{i_{j'}}$ to locate a third singleton job $s_{i_{j''}}$, where $j'' > j$ too, which is not isolated in π, and our path extends to $s_{i_{j''}}$. Due to the finitely many singleton jobs, our path ends at a singleton job $s_{i_{j^*}}$, which is the isolated target in π^*. See Fig. 6 for an illustration.

One sees that we have used the associated jobs t_{i_j}'s, which are distinct from each other, to locate a target isolated job in π^* for each source isolated job in π. Therefore, an isolated job in π^* wouldn't be discovered by multiple source isolated jobs in π; that is, the constructed mapping is an injection. In other words, the number of isolated jobs in π is not greater than the number of isolated jobs in π^*, the latter of which is denoted as c^*. There are b 2-subsets and c singletons in the partition $\mathcal{D}$; then there are c isolated jobs in π. We have

$$c \le c^*. \tag{1}$$

Recall the definition of an isolated job s_{i_j} in the optimal schedule π^*, that the machine M_1 processes nothing while any one of the machines M_2 and M_3 is processing s_{i_j}. That is, the machine M_1 idles for at least $2c^*$ units of time before the makespan. Since the load of M_1 is n, we have

$$C^*_{\max} \ge n + 2c^*. \tag{2}$$

On the other hand, we still have $\ell_{\max} \ge b + c$ and $C^*_{\max} \ge 3\ell_{\max}$; therefore,

$$C^*_{\max} \ge \max\{n + 2c^*, 3(b + c)\}, \tag{3}$$

which is a better lower bound than the one in Lemma 2.5. It follows that

$$C^{\pi}_{\max} = n + b + 2c = (n + 2c) + b \le C^*_{\max} + \frac{1}{3} C^*_{\max} = \frac{4}{3} C^*_{\max}.$$

This proves that the performance ratio for the algorithm Approx is 4/3.

For the running time, the algorithm Approx maintains the precedence relationships and updates the subsets L_i's for constructing the partition $\mathcal{D}$. The most time is spent for locating an agreeable job for accompanying a singleton job in U, which might take $O(n)$ time. Therefore, it is safe to conclude that the total running time of the algorithm Approx is $O(n^2)$. This finishes the proof of the theorem. $\square$

Corollary 3.6. *For any $m \geq 3$, the problem $Om \mid p_{ij} = 1, prec \mid C_{\max}$ admits an $O(n^2)$-time $(2 - \frac{2}{m})$-approximation algorithm, where n denotes the number of jobs.*

Proof. Basically we can construct from the acyclic partition $\mathcal{D}$ a schedule with makespan $C_{\max} \leq n + \sum_{i=1}^{m}(m-i)b_i$, where b_i denotes the number of i-subsets in the partition $\mathcal{D}$, for $1 \leq i \leq m-1$. (In the above case where $m = 3$, $c = b_1$ and $b = b_2$.) While the lower bounds in Eq. (3) are updated accordingly as $C^*_{\max} \geq \max\{n + (m-1)b^*_1, m\left(\sum_{i=1}^{m} b_i\right)\}$. Since we still have $b_1 \leq b^*_1$, these two inequalities imply that $C_{\max} \leq C^*_{\max} + \frac{m-2}{m}C^*_{\max} = (2 - \frac{2}{m})C^*_{\max}$. $\square$

4. Concluding remarks

We studied the open-shop scheduling problem for unit jobs under precedence constraints. The problem has been shown to be strongly NP-hard when the number of machines is part of the input [14], but left as an open problem when the number m of machines is a fixed constant greater than 2, since 1978 [8]. We approached this problem by proposing a $(2 - \frac{2}{m})$-approximation algorithm, for $m \geq 3$. Addressing the complexity and designing better approximation algorithms are both challenging and exciting.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

YC and AZ are supported by the NSFC Grants 11571252, 11771114 and 11971139; they are also supported by the CSC Grants 201508330054 and 201908330090, respectively. RG and GL are supported by the NSERC Canada.

References

[1] H. Bräsel, D. Kluge, F. Werner, A polynomial algorithm for the $[n/m/0, t_{ij} = 1, tree/C_{\max}]$ open shop problem, Eur. J. Oper. Res. 72 (1994) 125–134.
[2] P. Brucker, Scheduling Algorithms, Springer, New York, 2007.
[3] P. Brucker, B. Jurisch, M.Z. Jurisch, Open shop problems with unit time operations, Oper. Res. 37 (1993) 59–73.
[4] P. Brucker, S. Knust, Complexity results for scheduling problems, http://www2.informatik.uni-osnabrueck.de/knust/class/, 2009.
[5] C. Dürr, The scheduling zoo, http://schedulingzoo.lip6.fr, 2016.
[6] T. Gonzalez, S. Sahni, Open shop scheduling to minimize finish time, J. ACM 23 (1976) 665–679.
[7] R.L. Graham, Combinatorial scheduling theory, in: L.A. Steen (Ed.), Mathematics Today Twelve Informal Essays, Springer, New York, 1978.
[8] J.K. Lenstra, A.H.G. Rinnooy Kan, Complexity of scheduling under precedence constraints, Oper. Res. 26 (1978) 22–35.
[9] M. Pinedo, Scheduling: Theory, Algorithm and Systems, Springer, New York, 2016.
[10] D. Prot, O. Bellenguez-Morinea, A survey on how the structure of precedence constraints may change the complexity class of scheduling problems, J. Sched. 21 (2018) 3–16.
[11] S.V. Sevastianov, G.J. Woeginger, Makespan minimization in open shops: a polynomial time approximation scheme, Math. Program. 82 (1998) 191–198.
[12] S.V. Sevastianov, G.J. Woeginger, Linear time approximation scheme for the multiprocessor open shop problem, Discrete Appl. Math. 114 (2001) 273–288.
[13] V.S. Tanaev, Y.N. Sotskov, V.A. Strusevich, Scheduling Theory: Multi-Stage Systems, Springer, 1994.
[14] V.G. Timkovsky, Identical parallel machines vs. unit-time shops and preemptions vs. chains in scheduling complexity, Eur. J. Oper. Res. 149 (2003) 355–376.
[15] D.P. Williamson, L.A. Hall, J.A. Hoogeveen, C.A.J. Hurkens, J.K. Lenstra, S.V. Sevastianov, D.B. Shmoys, Short shop schedules, Oper. Res. 45 (1997) 288–294.