运筹学学报
Operations Research Transactions

**优先序约束的排序问题:基于最大匹配的近似算法**

张安,陈永,陈光亭,陈占文,舒巧君,林国辉

引用本文:
张安,陈永,陈光亭,陈占文,舒巧君,林国辉. 优先序约束的排序问题:基于最大匹配的近似算法[J]. 运筹学学报, 2022, 26(3): 57-74.
ZHANG An,CHEN Yong,CHEN Guangting,CHEN Zhanwen,SHU Qiaojun,LIN Guohui. Maximum matching based approximation algorithms for precedence constrained scheduling problems[J]. *Operations Research Transactions*, 2022, 26(3): 57-74.

2022 年 9 月 运 筹 学 学 报 第 26 卷 第 3 期
Sep., 2022 Operations Research Transactions Vol.26 No.3

DOI: 10.15960/j.cnki.issn.1007-6093.2022.03.005

# Maximum matching based approximation algorithms for precedence constrained scheduling problems*

ZHANG An[1] CHEN Yong[1] CHEN Guangting[2]
CHEN Zhanwen[1] SHU Qiaojun[1] LIN Guohui[3,†]

**Abstract** We investigate the problem to schedule a set of precedence constrained jobs of unit size on an open-shop or on a flow-shop to minimize the makespan. The precedence constraints among the jobs are presented as a directed acyclic graph called the precedence graph. When the number of machines in the shop is part of the input, both problems are strongly NP-hard on general precedence graphs, but were proven polynomial-time solvable for some special precedence graphs such as intrees. In this paper, we characterize improved lower bounds on the minimum makespan in terms of the number of agreeable pairings among certain jobs and propose approximation algorithms based on a maximum matching among these jobs, so that every agreeable pair of jobs can be processed on different machines simultaneously. For open-shop with an arbitrary precedence graph and for flow-shop with a spine precedence graph, both achieved approximation ratios are $2-\frac{2}{m}$, where $m$ is the number of machines in the shop.

**Keywords** job precedence, open-shop, flow-shop, maximum matching, approximation algorithm

**Chinese Library Classification** O221.7

**2010 Mathematics Subject Classification** 90B35

# 优先序约束的排序问题：基于最大匹配的近似算法*

张 安[1] 陈 永[1] 陈光亭[2] 陈占文[1] 舒巧君[1] 林国辉[3,†]

**摘要** 本文研究具有加工次序约束的单位工件开放作业和流水作业排序问题，目标函数为极小化工件最大完工时间。工件之间的加工次序约束关系可以用一个被称为优先图的有向无圈图来刻画。当机器数作为输入时，两类问题在一般优先图上都是强 NP-困难的，而在入树的优先图上都是可解的。我们利用工件之间的许可对数获得了问题的新下界，并基于许可工件之间的最大匹配设计近似算法，其中匹配的许可工件对均能同时在不同机器上加工。对于一般优先图的开放作业问题和脊柱型优先图的流水作业问题，我们在理论上证明了算法的近似比为 $2-\frac{2}{m}$，其中 $m$ 是机器数目。

**关键词** 工件序，开放作业，流水作业，最大匹配，近似算法

Received: 2022-01-08

* Funding projects: Zhejiang Provincial Natural Science Foundation (No. LY21A010014), National Natural Science Foundation of China (No. 11971139)

1. Department of Mathematics, Hangzhou Dianzi University. Zhejiang, Hangzhou 310018, China; 杭州电子科技大学数学系, 浙江杭州 310018

2. Zhejiang University of Water Resources and Electric Power. Zhejiang, Hangzhou 310018, China; 浙江水利水电学院, 浙江杭州 310018

3. Department of Computing Science, University of Alberta. Edmonton, T6G 2E8 Alberta, Canada; 阿尔伯塔大学计算科学系, 阿尔伯塔埃德蒙顿 T6G 2E8

† Corresponding author E-mail: guohui@ualberta.ca

**中图分类号** O221.7
**2010 数学分类号** 90B35

Scheduling theory is an important sub-area in Operations Research, where the operations to be executed are generally referred to as jobs and the facilities execute the operations are referred to as machines. Besides the inter-relationships between the jobs and the machines that describe how the jobs should be processed by the machines, there are intra-relationships among the machines and intra-relationships among the jobs. One class of job intra-relationships is the precedence, which specifies the constraints that some jobs have to be finished before some other jobs can be started. Numerous industrial applications lead to various precedence constrained scheduling problems[1], which have received much algorithmic study since their emergence.

Graham[2] proposed the precedence constrained multiprocessor scheduling to minimize the makespan, or $P \mid prec \mid C_{\max}$ in the three-field notation[3], and showed that the list scheduling (LS) procedure has a worst-case performance ratio of $2 - \frac{1}{m}$, where $m$ is the number of parallel identical machines.

If the processing time of a job on every machine is one unit, that is, $p_{ij} = 1$ (where $i$ indexes the machine and $j$ indexes the job, and it simplifies to $p_j = 1$ on parallel identical machines), then the job is called a unit job. For the precedence constrained multiprocessor scheduling for unit jobs, denoted as $P \mid prec, p_j = 1 \mid C_{\max}$, Lam and Sethi[4] (and Braschi and Trystram[5]) refined Graham's analysis to achieve a slightly improved ratio of $2 - \frac{2}{m}$. Three decades later, Gangal and Ranade[6] revisited the problem and presented a $(2-\frac{7}{3m+1})$-approximation algorithm, assuming $m \geqslant 4$. Under the unique game conjecture[7], Svensson[8] claimed that for $P \mid prec, p_j = 1 \mid C_{\max}$, it is NP-hard to approximate within a constant factor strictly less than 2. When the number $m$ of parallel identical machines is a fixed constant greater than or equal to three, the problem is denoted as $Pm \mid prec, p_j = 1 \mid C_{\max}$; it is the "OPEN8" problem in the original list of Garey and Johnson[9], and it is still unknown to be NP-hard or not. Schuurman and Woeginger[10] also post an open question on whether $Pm \mid prec, p_j = 1 \mid C_{\max}$ admits a polynomial time approximation scheme (PTAS), for any fixed $m \geqslant 3$. Recently, a result by Levey and Rothvoss[11] implies that $Pm \mid prec, p_j = 1 \mid C_{\max}$ cannot be APX-hard, assuming $NP \nsubseteq DTIME(n^{\log(n)^{o(\log\log n)}})$.

The above results (the first six rows in Table 1) are for general precedence graphs. One can imagine and it is true that the computational complexity of the multiprocessor scheduling varies with different precedence graphs, as well as with different objective functions. We refer the readers to a survey[12] for many variants and known results.

The machines in the multiprocessor scheduling are identical and a job needs to be processed by only one of them. In an open-shop or a flow-shop of $m$ machines, a job needs to be processed by all the $m$ machines, in an arbitrary machine order or a fixed order, respectively. The existing research on the precedence constrained open-/flow-shop scheduling problems is limited, mostly complexity-oriented, and has not been updated for a long time (the last eight rows in Table 1). When the number $m$ of machines is part of the input, it was known that even $O \mid prec, p_{ij} = 1 \mid C_{\max}$ and $F \mid prec, p_{ij} = 1 \mid C_{\max}$

**Table 1** Complexity and approximability results on precedence constrained scheduling

| Problem | Complexity | Approximation |
|---|---|---|
| $P \mid prec \mid C_{\max}$ | NP-hard[2] | $2-\frac{1}{m}$ [2] |
| $P \mid prec, p_j=1 \mid C_{\max}$ | NP-hard to 2-approx[8] | $2-\frac{2}{m}$ [4, 5] |
| $P \mid prec \mid C_{\max}$, $m \geqslant 4$ | NP-hard | $2-\frac{7}{3m+1}$ [6] |
| $m \geqslant 3$ | NP-hard open[9] | |
| $Pm \mid prec, p_j=1 \mid C_{\max}$ $m \geqslant 4$ | PTAS open[10] | |
| $m \geqslant 4$ | Not APX-hard[11] | |
| $O \mid prec, p_{ij}=1 \mid C_{\max}$ | Strongly NP-hard[13, 14] | $2-\frac{2}{m}$ ([19] and this paper) |
| $F \mid prec, p_{ij}=1 \mid C_{\max}$ | Strongly NP-hard[13, 14] | |
| $F \mid spine, p_{ij}=1 \mid C_{\max}$ | NP-hard open | $2-\frac{2}{m}$ (this paper) |
| $O/F \mid intree/outtree, p_{ij}=1 \mid C_{\max}$ | P[17, 16] | |
| $O/F \mid intree, p_{ij}=1 \mid L_{\max}$ | P[15, 16] | |
| $O/F \mid outtree, p_{ij}=1 \mid L_{\max}$ | Strongly NP-hard[18, 14] | |
| $Om/Fm \mid prec, p_{ij}=1 \mid C_{\max}$, $m \geqslant 3$ | NP-hard open | |
| $O2/F2 \mid prec, p_{ij}=1 \mid L_{\max}$ | P[15, 16] | |

are already strongly NP-hard on general precedence graphs[13, 14]. When $m \geqslant 3$ is a fixed constant, the computational complexities of $Om \mid prec, p_{ij}=1 \mid C_{\max}$ and $Fm \mid prec, p_{ij} = 1 \mid C_{\max}$ are both open. Nevertheless, when $m=2$, for a more general objective to minimize the maximum lateness $L_{\max}$, both $O2 \mid prec, p_{ij}=1 \mid L_{\max}$ and $F2 \mid prec, p_{ij}=1 \mid L_{\max}$ are polynomial-time solvable, even when the jobs have different release times[15, 16].

Given a precedence graph, by noting that the precedence relation is transitive, we may remove the "redundant" precedence constraints from the graph, and thus we may assume without loss of generality that there are no redundant constraints in the given precedence graph. Then, a constraint in the precedence graph specifies a job is the immediate predecessor of the other job (or the other way around, the latter job is the immediate successor of the former). If each job has at most one immediate successor (predecessor, respectively), then the precedence graph is an intree (outtree, respectively). Bräsel et al.[17] proved that $O \mid intree/outtree, p_{ij}=1 \mid C_{\max}$ admits a polynomial-time exact algorithm, that is, the precedence graph is an intree or an outtree; the same conclusion holds for flow-shop counterpart[16]. Interestingly, both $O \mid intree, p_{ij}=1 \mid L_{\max}$ and $F \mid intree, p_{ij} = 1 \mid L_{\max}$ are polynomial-time solvable[15, 16], while both $O \mid outtree, p_{ij}=1 \mid L_{\max}$ and $F \mid outtree, p_{ij}=1 \mid L_{\max}$ are strongly NP-hard[18, 14].

In this paper, we study the two problems $O \mid prec, p_{ij}=1 \mid C_{\max}$ and $F \mid prec, p_{ij} = 1 \mid C_{\max}$ from the approximation algorithm perspective, and we assume the input $m \geqslant 3$ given that $O2 \mid prec, p_{ij}=1 \mid L_{\max}$ and $F2 \mid prec, p_{ij}=1 \mid L_{\max}$ are polynomial-time solvable[15, 16]. In the literature, few approximation algorithm exists except the most recently proposed $(2-\frac{2}{m})$-approximation algorithm for $O \mid prec, p_{ij}=1 \mid C_{\max}$ by Chen et al.[19]. We observe the special jobs on the spine of the precedence graph, characterize improved lower bounds on the minimum makespan in terms of the number of agreeable pairings among certain jobs, and propose approximation algorithms based on a maximum matching among these jobs, so that every agreeable pair of jobs can be processed on different machines simultaneously.

In the next section we introduce definitions and the preprocessing of the precedence graph to partition the jobs into layers, and construct the so-called spine of the graph[19]. In Section 3 we deal with open-shop scheduling, present a maximum matching scheme between the singletons, which are on the spine, and the jobs outside of the spine, and show that the resulting approximation algorithm has the same performance ratio of $(2-\frac{2}{m})$ as the one in [19] (the seventh row in Table 1). Flow-shop scheduling is dealt with in Section 4, where we present a maximum matching scheme between agreeable jobs in adjacent layers, and show that it leads to a $(2-\frac{2}{m})$-approximation algorithm when all the jobs are on the spine (the ninth row in Table 1). For both algorithms, the ratio $2-\frac{2}{m}$ is shown tight. We conclude the paper in Section 5.

## 1 Definitions and Preliminaries

We use $V=\{1,2,\cdots,n\}$ to denote the set of unit-jobs and $G=(V,E)$ to denote the directed precedence graph, in which a directed edge $(j_1,j_2)\in E$ states the constraint that the job $j_1$ precedes the other job $j_2$, that is, processing $j_2$ can be started only if $j_1$ has been processed by all the machines (i.e., completed). In the sequel, the word directed is often dropped. Note that the precedence relation is transitive, that is, $j_1$ precedes $j_2$ and $j_2$ precedes $j_3$ imply that $j_1$ precedes $j_3$, and in this case we call $(j_1,j_3)\in E$ a redundant precedence constraint in $E$. The precedence graph $G=(V,E)$ is a directed acyclic graph and $E$ is assumed containing no redundant constraints (or otherwise we may remove all of them from $E$, in $O(n^2)$ time).

If $j_1$ precedes $j_2$, then we call $j_1$ a predecessor of $j_2$ and $j_2$ a successor of $j_1$; additionally, if $(j_1,j_2)\in E$, then $j_1$ is an immediate predecessor of $j_2$ and $j_2$ is an immediate successor of $j_1$. In the sequel, a job might also be referred to as a vertex in the precedence graph, and we use vertex and job interchangeably.

If neither $j_1$ precedes $j_2$ nor $j_2$ precedes $j_1$, then they are called agreeable with each other and they can be processed by different machines simultaneously. A set $X$ of jobs is called agreeable if every pair of jobs in $X$ are agreeable; if $X\cup\{j\}$ is agreeable, then we say that the job $j$ is agreeable with $X$. For two agreeable sets $X_1$ and $X_2$, if there exists a job $j_i\in X_i, i=1,2$ such that $j_1$ precedes $j_2$, then we say that $X_1$ precedes $X_2$; additionally, if every job in $X_1$ precedes all the jobs in $X_2$, then $X_1$ fully precedes $X_2$. We remark that two agreeable sets might precede each other, but they cannot fully precede each other. Below we construct a sequence of agreeable sets so that one precedes the next set, but not its preceding set.

The following preprocessing of the precedence graph to partition the jobs into agreeable layers is presented in [19]. Given the precedence graph $G = (V, E)$, the first layer, denoted as $L_1$, of jobs are those without any immediate predecessors (that is, without any inedges), and they are subsequently removed from the precedence graph for further partitioning; iteratively, if the remaining precedence graph is non-empty, then the next layer of jobs are those without any immediate predecessors, and they are subsequently removed from further partitioning. The thus determined layers form into the layered representation $\mathcal{L} = \{L_1, L_2, \cdots, L_\ell\}$, assuming there are in total $\ell$ layers. Note that $\mathcal{L}$ can be constructed in $O(n^2)$ time, and each layer $L_i$ is non-empty and agreeable. For convenience, a job of $L_i$ is also called a level-$i$ job; by the construction process, we conclude that each level-$i$ job has an immediate predecessor in $L_{i-1}$, for $i \geqslant 2$, and that $L_i$ precedes $L_j$ but not the other way around for any pair $i < j$. We remark that $L_i$ does not necessarily fully precede $L_j$.

One sees that a longest (directed, omitted in the sequel) path in the precedence graph $G = (V, E)$ contains exactly $\ell$ vertices, among which a vertex precedes all the other vertices with larger level indices. This fact implies a lower bound of $m\ell$ time units on the makespan. Let $C^*_{OS}$ and $C^*_{FS}$ denote the minimum makespan for the problems $O \mid prec, p_{ij} = 1 \mid C_{\max}$ and $F \mid prec, p_{ij} = 1 \mid C_{\max}$, respectively, in which there are $m$ machines, $n$ jobs, and there are $\ell$ layers in the precedence graph $G = (V, E)$. Then,

$$C^*_{OS} \geqslant \max\{n, m\ell\}, \quad C^*_{FS} \geqslant \max\{n + m - 1, m\ell\}. \tag{1}$$

Using the layered representation $\mathcal{L}$, we can schedule in $O(n+m)$ time the jobs level by level where the jobs of each layer $L_i$ are processed in the same order on the $m$ machines and finished in $|L_i| + m - 1$ time units, in either of the open-shop and the flow-shop. It follows that the achieved makespan is $\sum_{i=1}^{\ell}(|L_i| + m - 1) = n + (m - 1)\ell$. Combining with the lower bounds in Eq. (1), we have the following theorem.

**Theorem 1** *The layered representation $\mathcal{L}$ of the precedence graph $G = (V, E)$ leads to an $O(n^2+m)$-time $(2-\frac{1}{m})$-approximation algorithm for the problems $O \mid prec, p_{ij} = 1 \mid C_{\max}$ and $F \mid prec, p_{ij} = 1 \mid C_{\max}$, respectively, where $m \geqslant 3$ is the number of machines in the shop and $n$ is the number of jobs.*

Let $U$ be the set of all the vertices that are on the longest paths in the precedence graph $G = (V, E)$. The set $U$ can be determined as follows: First, let $U_\ell = L_\ell$, that is, the set of level-$\ell$ jobs; then let $U_{\ell-1}$ be the set of immediate predecessors of the jobs of $U_\ell$ that are in $L_{\ell-1}$. We remark that $U_{\ell-1}$ is non-empty, and some immediate predecessors of the jobs of $U_\ell$ might not be in $L_{\ell-1}$. Iteratively, let $U_i$ be the set of immediate predecessors of the jobs of $U_{i+1}$ that are in $L_i$, for $i = \ell-2, \ell-3, \cdots, 1$; and lastly $U = U_1 \cup U_2 \cup \cdots \cup U_\ell$. The subgraph induced on $U$, $G[U] = (U, F)$, is defined as the spine[19] of the precedence graph $G = (V, E)$.

If $|U_i| = 1$ for some $i$, then the unique job of $U_i$ is called a singleton and $U_i$ is called a singleton subset. Let $\mathcal{U}^1$ denote the collection of all the singleton subsets. Note that $U_i \subseteq L_i$. Let $R_i = L_i \setminus U_i$ for each $i$, and $R = R_1 \cup R_2 \cup \cdots \cup R_{\ell-1}$; for convenience, a job in

$R$ ($R_i$, $U$, $U_i$, respectively) is called an $R$-job ($R_i$-, $U$-, $U_i$-job, respectively). If $R = \varnothing$, then the precedence graph $G = (V, E)$ is called a spine graph. For example, a single path, which is referred to as a chain[20] in the literature, is a spine graph. An intree or an outtree is not necessarily a spine graph, unless all its root-to-leaf or leaf-to-root paths, respectively, have the same length. Let $F \mid spine, p_{ij} = 1 \mid C_{\max}$ denote the problem when the precedence graph is a spine graph.

## 2 A matching-based approximation for $O \mid prec, p_{ij} = 1 \mid C_{\max}$

Consider an instance of $O \mid prec, p_{ij} = 1 \mid C_{\max}$, in which the precedence graph $G = (V, E)$ comes with its layered representation $\mathcal{L} = \{L_1, L_2, \cdots, L_\ell\}$, the spine $G[U] = (U, F)$, and the collection $\mathcal{U}^1$ of the singleton subsets. For each singleton subset $U_i \in \mathcal{U}^1$, let $u_i$ denote the unique $U_i$-job, and let $U^1$ denote the set of these singleton jobs.

In the sequel, the first number in the subscript of a job/vertex refers to the level of the job/vertex. For example, $r_{41}$ is a job/vertex of $R_4$.

In the first step of the approximation algorithm Approx 1 (of which a high-level description of the algorithm Approx 1 is depicted in Fig. 2), an auxiliary (undirected) bipartite graph $H = (U^1, R, E^1)$ is constructed. The vertices in one part are the singleton jobs and the vertices in the other part are the $R$-jobs. In the bipartite graph $H$, a singleton job $u_i \in U^1$ is adjacent to all agreeable jobs in $R_1 \cup R_2 \cup \cdots \cup R_i$, which include all the jobs of $R_i$ in particular (see Fig. 1 for an illustration, where $E^1$ consists of all the seven dashed edges).

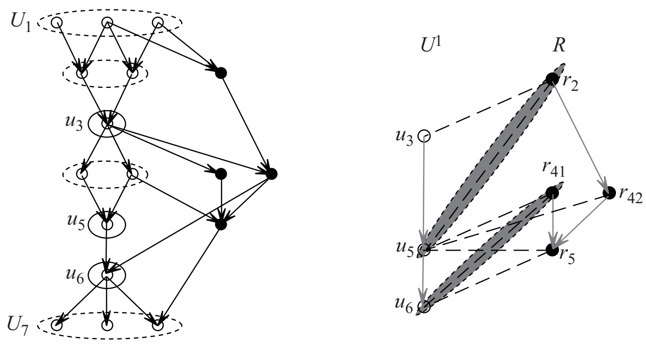


**Fig. 1** A precedence graph $G = (V, E)$ and its spine $G[U]$ (left), where the unfilled vertices form into $U$ and the four filled vertices form into $R$; $U^1 = \{u_3, u_5, u_6\}$ and the auxiliary bipartite graph $H = (U^1, R, E^1)$ (right), in which the gray directed edges are precedence and the dashed undirected edges are in $E^1$. A matching in $H$, $M = \{(u_5, r_2), (u_6, r_{41})\}$ with its two edges highlighted, becomes $\{(u_5, r_5), (u_6, r_{41})\}$ after the upgrading process, since $r_5$ is a successor of $r_2$; and further becomes $\{(u_5, r_{41}), (u_6, r_5)\}$ after the de-crossing process, which is non-upgradeable and non-crossing

The following lemma summarizes some structural properties of the bipartite graph $H$.

**Lemma 1** *In $H = (U^1, R, E^1)$, if a singleton job $u_i$ is adjacent to a job $r \in R_{i'}$ such that $i' < i$, then*

(1) *$u_i$ is adjacent to every job $r' \in R_{i''}$ such that $r$ precedes $r'$ and $i' < i'' < i$;*

(2) *every other singleton $u_{i''}$ with $i' < i'' < i$ is adjacent to $r$.*

**Proof** The proof is done by contradiction and the transitivity of the precedence relation.

Note that the singleton job $u_i$ is agreeable with the job $r$. If $u_i$ is not agreeable with the job $r'$, then we know from $i'' < i$ that $r'$ precedes $u_i$; it follows from the transitivity that $r$ precedes $u_i$, a contradiction. This proves the first item, and the second item can be proven in the same way. □

For the example illustrated in Figure 1, where $(u_6, r_{41}) \in E^1$, $r_5$ is a successor of $r_{41}$ and $u_5$ is a singleton, by Item (1) of Lemma 1 we have $(u_6, r_5) \in E^1$ and by Item (2) of Lemma 1 we have $(u_5, r_{41}) \in E^1$ too.

Given a matching in the bipartite graph $H = (U^1, R, E^1)$, which is a subset of edges that are non-adjacent to each other, and an edge $(u_i, r)$ in the matching, we say that the singleton job $u_i$ and the $R$-job $r$ are covered by (the edge in) the matching. If there is an $R$-successor $r'$ of $r$ which has a level $i' \leqslant i$ and is not covered by any other edge in the matching, then either since they are at the same level or by Lemma 1 we know that $(u_i, r')$ is in $E^1$ and thus we can use $(u_i, r')$ to replace the edge $(u_i, r)$. For example in Figure 1, a matching $M = \{(u_5, r_2), (u_6, r_{41})\}$ and $r_5$ is a successor of $r_2$, then the edge $(u_5, r_2)$ can be replaced by $(u_5, r_5)$. We refer this process of increasing the levels of the covered $R$-jobs until impossible to as upgrading. The resultant matching is said non-upgradeable. The following lemma shows that the upgrading process can be executed in $O(|M||R|)$ time.

**Lemma 2** *In $H = (U^1, R, E^1)$, every matching $M$ can be converted into another non-upgradeable matching of the same size in $O(|M||R|)$ time.*

**Proof** Let $(u_i, r)$ be an edge in the matching $M$ and $r'$ be an $R$-successor of $r$ which has a level $i' \leqslant i$ and is not covered by $M$. Moreover, the edge $(u_i, r)$ is selected so that the level index $i$ is the maximum, and subsequently so that level index $i'$ achieves the maximum. Lemma 1 states that $(u_i, r')$ is also an edge in $E^1$. Therefore, the edge $(u_i, r)$ of the matching $M$ can be replaced by $(u_i, r')$, while releasing $r$ of level strictly less than $i$ to be uncovered. The selection of $(u_i, r)$ and $r'$ guarantees that the new edge $(u_i, r')$ of the matching $M$ will not be upgraded afterwards.

Note that for each singleton job $u_i$ covered by the matching $M$, finding a corresponding $R$-job $r'$ for upgrading takes $O(|R|)$ time. It follows that the overall upgrading time is $O(|M||R|)$. □

Given a matching in the bipartite graph $H = (U^1, R, E^1)$ and two edges $(u_i, r)$ and $(u_{i'}, r')$ with $i' < i$ (implying $u_{i'}$ precedes $u_i$), if $r$ precedes $r'$, then the two edges are called crossing each other. Lemma 1 states that in this case both $(u_i, r')$ and $(u_{i'}, r)$ are also edges in $E^1$. Therefore, the two crossing edges can be replaced by the two non-crossing edges $(u_i, r')$ and $(u_{i'}, r)$, a process referred to as de-crossing. For example in Figure 1, a matching $M = \{(u_5, r_5), (u_6, r_{41})\}$ and its two edges are crossing, then the de-crossing process replaces these two edges by $(u_5, r_{41})$ and $(u_6, r_5)$. A matching not containing any crossing edges is said non-crossing.

**Lemma 3** *In $H = (U^1, R, E^1)$, every matching $M$ can be converted into another non-upgradeable and non-crossing matching of the same size in $O(|M||R|)$ time.*

**Proof** Given a matching $M$, by Lemma 2, we first execute the upgrading process in $O(|M||R|)$ time; the achieved non-upgradeable matching is still denoted as $M$.

When there are two crossing edges $(u_i, r)$ and $(u_{i'}, r')$ with $i' < i$ in the matching $M$, they are replaced by the corresponding two non-crossing edges $(u_i, r')$ and $(u_{i'}, r)$. A simple contradiction can be deployed to prove that the resultant matching is still not upgradeable. It follows that we can execute the de-crossing process to sequentially ensure that the edge of the matching $M$ incident at the singleton job $u_i$ with the current largest level index $i$ is not crossing with any other edges of $M$; and consequently the de-crossing process is executed in $O(|M|^2)$ time.

From $|M| \leqslant |R|$, the overall time for upgrading, followed by de-crossing, is thus $O(|M||R|)$. □

**Lemma 4** *A non-upgradeable and non-crossing matching $M$ in $H = (U^1, R, E^1)$ gives rise to a partition of the jobs into a sequence of $\ell$ agreeable subsets, among which there are exactly $|U^1| - |M|$ singleton subsets. Furthermore, processing the jobs subset-by-subset sequentially gives rise to a feasible schedule of makespan no greater than $n + (m-2)\ell + (|U^1| - |M|)$.*

**Proof** Let $R'$ denote the set of $R$-jobs not covered by $M$. For each $i$ such that $U_i$ is not a singleton subset or the singleton $u_i$ is not covered by $M$, let $L'_i = (R_i \cap R') \cup U_i$, which is the subset of all the un-covered jobs in $L_i$. For each singleton job $u_i$ covered by an edge $(u_i, r)$ in $M$, let $L'_i = (R_i \cap R') \cup \{u_i, r\}$ (that is, the subset of all the un-covered jobs in $L_i$ plus the two covered jobs $u_i$ and $r$).

One sees that each $L'_i$ is non-empty and agreeable, and $L'_i$ precedes $L'_{i+1}$ for every $i \leqslant \ell - 1$.

Next we want to prove that if $i > j$ then $L'_i$ does not precede $L'_j$. When $U_i$ is not a singleton subset or the singleton $u_i$ is not covered by $M$, $L'_i$ does not precede $L'_j$ since $L'_j \subseteq L_1 \cup L_2 \cup \cdots \cup L_j$ while $L'_i \subseteq L_i$.

When $U_i$ is a singleton subset and $u_i$ is covered by an edge $(u_i, r) \in M$, but $U_j$ is not a singleton subset or the singleton $u_j$ is not covered by $M$, $L'_i$ precedes $L'_j$ only if $r$ precedes a job of $L'_j \subseteq L_j$. However, if $r$ precedes a job of $U_j$ then $r$ precedes $u_i$, a contradiction; if $r$ precedes an $R$-job of $L'_j \setminus U_j$ then the matching $M$ can be upgraded, again a contradiction.

When both $U_i$ and $U_j$ are singleton subsets and, $u_i$ is covered by an edge $(u_i, r) \in M$ and $u_j$ is covered by an edge $(u_j, r') \in M$, $L'_i$ precedes $L'_j$ only if $r$ precedes $r'$. However, this suggests the two edges $(u_i, r)$ and $(u_j, r')$ are crossing, a contradiction. This finishes the proof that $L'_i$ does not precede $L'_j$ for any $i > j$.

Let $\mathcal{L}' = \{L'_1, L'_2, \cdots, L'_\ell\}$, which is a layered representation similar to $\mathcal{L}$, and the jobs can be processed layer-by-layer where the jobs of each layer $L'_i$ are processed in the permuted (i.e., cyclic) order on the $m$ machines in $\max\{|L'_i|, m\}$ time units. It follows that we achieve a feasible schedule in which for processing the jobs of $L'_i$, each machine idles exactly $\max\{0, m - |L'_i|\}$ time units. Since we have $|U^1|$ singleton jobs and $|M|$ of them are covered by the matching $M$, the number of layers of $\mathcal{L}'$ each contains only one job is at most $|U^1| - |M|$. Therefore, the makespan of the achieved schedule is

$$C_{\max} \leqslant n + (m-2)\ell + (|U^1| - |M|). \tag{2}$$

This finishes the proof of the lemma. We remark that from the given matching $M$, constructing the feasible schedule takes $O(n+m)$ time. □

The above analysis motivates the following second step of the algorithm Approx 1, in which a maximum (cardinality) matching $M^*$ in the bipartite graph $H = (U^1, R, E^1)$ is computed in $O(n^{1.5}\ell)$[21, 22] (or $\tilde{O}(|E^1|^{10/7})$[23]) time; and then from $M^*$ a feasible schedule $\pi$ is constructed using Lemmas 2, 3 and 4 in $O(n\ell + m)$ time. A high-level description of the complete algorithm Approx 1 is depicted in Fig. 2.

Algorithm Approx 1:

0. Initialization: Preprocess the precedence graph $G = (V, E)$ for
   - 0.1 the layered representation $\mathcal{L} = \{L_1, L_2, \cdots, L_\ell\}$,
   - 0.2 the spine $G[U] = (U, F)$, $R = V \setminus U$, and
   - 0.3 the set $U^1$ of singleton jobs in the spine;
1. Construct the auxiliary (undirected) bipartite graph $H = (U^1, R, E^1)$;
2. Compute a maximum matching $M^*$ in $H$, then
   - 2.1 upgrade $M^*$;
   - 2.2 de-crossing $M^*$;
   - 2.3 construct the layered representation $\mathcal{L}' = \{L'_1, L'_2, \cdots, L'_\ell\}$;
   - 2.4 construct a feasible schedule $\pi$.

**Fig. 2** A high-level description of the algorithm Approx 1

We have showed the makespan of the schedule achieved by the algorithm Approx 1 in Eq. (2). For the approximation ratio, we will prove next an improved lower bound on the minimum makespan using the number $|U^1|$ of singletons, in Eq. (3).

Consider an optimal schedule $\pi^*$ that achieves the minimum makespan $C^*_{OS}$, in which we assume without loss of generality that every job is processed at an integral time point for one time unit. The entire time span $[0, C^*_{OS}]$ for the first machine in the open-shop is partitioned into $C^*_{OS}$ unit time intervals. During each unit time interval, if one of the other $m-1$ machines processes a singleton job $u_i$, then this unit time interval is said associated to $u_i$. One sees that each singleton job is associated with exactly $m-1$ distinct unit time intervals, while a unit time interval can be associated to at most one singleton job (since singleton jobs cannot be processed simultaneously).

Assume the unit time interval $[j, j+1]$, for some $j$, is associated to the singleton job $u_i$. If the first machine processes a job $r$ during this time interval $[j, j+1]$, then $r$ is an $R$-job agreeable with $u_i$. In the bipartite graph $H = (U^1, R, E^1)$ we construct a subset $M \subseteq E^1$ of edges as follows: If the job $r$ has a level index no greater than $i$, then the edge $(u_i, r)$ is selected into $M$; otherwise, we conclude that $r$ has a level-$i$ predecessor $r' \in R_i$ which is agreeable with $u_i$, and then the edge $(u_i, r')$ is selected into $M$. Afterwards, no more edge incident at $u_i$ will be selected, even if the first machine processes another job during some other unit time interval associated to $u_i$.

**Lemma 5** *Given an optimal schedule $\pi^*$, the constructed edge subset $M$ is a matching in the bipartite graph $H = (U^1, R, E^1)$, and the makespan of $\pi^*$ is at least $n+(|U^1|-|M|)(m-1)$.*

**Proof** In the constructed edge subset $M$, every singleton job is covered by at most one edge; therefore, if $M$ is not a matching, then there exist two distinct singleton jobs denoted as $u_i$ and $u_{i'}$ with $i < i'$ such that both $(u_i, r)$ and $(u_{i'}, r)$ are in $M$ for some $R_{i''}$-job $r$ with $i'' \leqslant i$. By the edge selecting rule, we conclude that $i'' = i$ since otherwise $u_i$ and $u_{i'}$ would be processed by different machines simultaneously with the job $r$. Furthermore, in the optimal schedule $\pi^*$, $u_{i'}$ and $r$ are processed by different machines simultaneously during some unit time interval $[j', j'+1]$, while $u_i$ and a successor $r'$ of $r$ are processed by different machines simultaneously during another unit time interval $[j, j+1]$. However, the two constraints that $u_i$ precedes $u_{i'}$ and $r$ precedes $r'$ state clearly that $j < j'$ and $j' < j$, a contradiction. That is, every $R$-job is covered by at most one edge of $M$ too. This proves that $M$ is a matching in the bipartite graph $H = (U^1, R, E^1)$.

If a singleton job $u_i$ is not covered by any edge of $M$, then when $u_i$ is processed on any machine except the first machine, the first machine idles during that particular unit time interval. Therefore, the first machine idles in total $m-1$ time units associated with $u_i$, and consequently the first machine idles in total at least $(|U^1| - |M|)(m-1)$ time units. That is, the makespan

$$C^*_{OS} \geqslant n + (|U^1| - |M|)(m-1). \tag{3}$$

This finishes the proof of the lemma. □

**Theorem 2** *The matching based algorithm* Approx 1 *is an $O(\max\{n^2, \min\{n^{1.5}\ell, n^{10/7}\ell^{10/7}\}\} + m)$-time $(2-\frac{2}{m})$-approximation for the problem $O \mid prec, p_{ij} = 1 \mid C_{\max}$, where $m \geqslant 2$ is the number of machines in the open-shop, and the approximation ratio is tight.*

**Proof** Recall that in the second step of the algorithm Approx 1, a maximum matching $M^*$ in the bipartite graph $H = (U^1, R, E^1)$ is computed in $O(n^{1.5}\ell)$ (or $\tilde{O}(n^{10/7}\ell^{10/7})$) time, and from $M^*$ a feasible schedule $\pi$ is constructed in $O(n\ell + m)$ time which has a makespan $C_{\max} \leqslant n + (m-2)\ell + (|U^1| - |M^*|)$. Using the lower bounds of the minimum makespan in Eqs. (1) and (3), we have

$$C_{\max} \leqslant C^*_{OS} + (m-2)\ell \leqslant C^*_{OS} + \frac{m-2}{m} C^*_{OS} = \Big(2 - \frac{2}{m}\Big) C^*_{OS}.$$

That is, Approx 1 is a $(2-\frac{2}{m})$-approximation algorithm for the problem $O \mid prec, p_{ij} = 1 \mid C_{\max}$.

From the high-level description of the algorithm Approx 1 in Figure 2, we see that the initialization and the construction of the bipartite graph $H$ take $O(n^2)$ time. Therefore, the overall running time of Approx 1 is $O(\max\{n^2, \min\{n^{1.5}\ell, n^{10/7}\ell^{10/7}\}\} + m)$.

Consider an instance precedence graph $G = (V, E)$ for which there are $\ell$ levels of jobs, every job in the spine is a singleton job, the first level $L_1$ contains the job $u_1$ and $(m-1)\ell$ $R_1$-jobs, and $n = m\ell$. For this instance, the constructed bipartite graph $H = (U^1, R, E^1)$ is complete, a maximum matching $M^*$ contains exactly $\ell$ edges, from which the constructed feasible schedule $\pi$ has its makespan $C_{\max} = n + (m-2)(\ell-1)$. On the other hand, one can associate $m-1$ distinct $R_1$-jobs to each singleton job, resulting in a schedule of makespan $m\ell = n$ (and thus optimal). Therefore, the performance ratio of Approx 1 on this instance is

$$\frac{m\ell + (m-2)(\ell-1)}{m\ell} = 2 - \frac{2}{m} - \frac{m-2}{m\ell} \longrightarrow 2 - \frac{2}{m},$$

when $\ell$ (or equivalently, $n = m\ell$) approaches $+\infty$. This shows the tightness of the approximation ratio $2 - \frac{2}{m}$. □

## 3 A matching-based approximation for $F \mid spine, p_{ij} = 1 \mid C_{\max}$

The flow-shop scheduling to minimize the makespan is one of the classic scheduling models [9,SS15]. A schedule in which the job processing order is the same across all the machines is called a permutation schedule. It is known that when the number $m$ of machines is two or three, the flow-shop scheduling problem without precedence constraints, that is, $Fm \mid\mid C_{\max}$, admits an optimal schedule that is permutation, but not necessarily when $m \geqslant 4$[24] or when $m$ is part of the input. Nevertheless, for unit-jobs and an arbitrary precedence graph, one can prove by a simple induction on the number $n$ of jobs that the general problem $F \mid prec, p_{ij} = 1 \mid C_{\max}$ admits an optimal schedule that is permutation and no-wait, by "no-wait" every job is processed sequentially through the $m$ machines continuously (i.e., from one machine to another without waiting for the machines to become available) and completed in exactly $m$ time units. One thus sees that in such a permutation and no-wait schedule, how the jobs are processed on the $m$ machines are identical, except that the first machine starts at time 0, the second machine starts at time 1, and the $i$-th machine starts at time $i-1$ for $i = 3, 4, \cdots, m$. Also, if one job $j_1$ precedes the other job $j_2$, then on each machine $j_2$ starts processing at least $m-1$ time units after $j_1$ is finished by the machine. We summarize these into the following lemma.

**Lemma 6** *The problem $F \mid prec, p_{ij} = 1 \mid C_{\max}$ admits an optimal schedule that is permutation and no-wait, in which how the jobs are processed on the machines are identical, except that the $i$-th machine starts processing jobs at time $i-1$, and if one job precedes another, then on each machine their start processing times are at least $m$ time units apart.*

In the sequel we consider only permutation and no-wait schedules, and the precedence graph $G = (V, E)$ is a spine graph (that is, $R = \varnothing$). We note that the complexity of the problem $F \mid spine, p_{ij} = 1 \mid C_{\max}$ is unknown, but it seems to be NP-hard. Consider an instance of $F \mid spine, p_{ij} = 1 \mid C_{\max}$, in which the precedence graph $G = (V, E)$ comes with its layered representation $\mathcal{L} = \{L_1, L_2, \cdots, L_\ell\}$. Since $G$ is a spine graph, $U_i = L_i$

for each $i$. One important fact about a spine graph is that every job of $U_i$ has a successor in each $U_{i'}$ where $i' > i$, and the other way around every job of $U_{i'}$ has a predecessor in $U_i$. Consequently, if there is an index $i \leqslant \ell - 1$ such that $U_i$ fully precedes $U_{i+1}$, then all the jobs of $U_1 \cup U_2 \cup \cdots \cup U_i$ are predecessors of every job of $U_{i+1} \cup U_{i+2} \cup \cdots \cup U_\ell$. One thus sees that the instance can be decomposed into two independent sub-instances with the precedence graphs $G[U_1 \cup U_2 \cup \cdots \cup U_i]$ and $G[U_{i+1} \cup U_{i+2} \cup \cdots \cup U_\ell]$, respectively, followed by concatenating their solution schedules into a schedule for the original instance. For this reason, we may assume without loss of generality that no $U_i$ fully precedes $U_{i+1}$, which implies no singleton subset among the $U_i$'s.

In the first step of the approximation algorithm Approx 2 (of which a high-level description is depicted in Fig. 4), an auxiliary undirected graph $\overline{G} = (V, \overline{E})$ is constructed. We use the notation $\overline{G}$ for the reason that this auxiliary graph is a part of the complement of $G$: For every $i = 1, 2, \cdots, \ell - 1$, between $U_i$ and $U_{i+1}$, if a job $j_1 \in U_i$ does not precede a job $j_2 \in U_{i+1}$, then $j_1$ and $j_2$ are adjacent in $\overline{G}$, that is, the undirected edge $(j_1, j_2) \in \overline{E}$ indicates that $j_1$ and $j_2$ are agreeable so that they can be processed simultaneously. We use $\overline{E}(U_i, U_{i+1})$ to denote the subset of edges between $U_i$ and $U_{i+1}$, which is non-complete and non-empty by our assumption (see Figure 3 for an illustration).

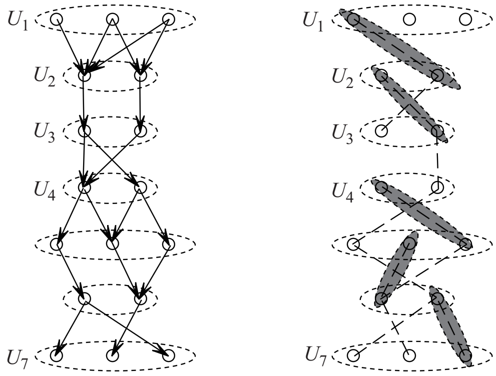


**Fig. 3** A spine precedence graph $G = (V, E)$ (left), in which no $U_i$ fully precedes $U_{i+1}$ for any $i$; the auxiliary graph $\overline{G} = (V, \overline{E})$ (right), in which the edges are dashed. A lexicographically largest matching in $\overline{G}$ is highlighted with its binary vector $(1, 1, 0, 1, 1, 1)$, which contains no edge from $\overline{E}(U_3, U_4)$

A matching in $\overline{G}$ is called an agreement matching if it contains at most one edge from each $\overline{E}(U_i, U_{i+1})$. In the sequel we consider only agreement matchings in $\overline{G}$ and drop the word "agreement". Given a matching $M$, let $\delta_i^M = |M \cap \overline{E}(U_i, U_{i+1})|$, which is binary indicating whether or not the matching contains an edge from $\overline{E}(U_i, U_{i+1})$. This way, by ignoring the detailed edges, $M$ can be represented as an $(\ell - 1)$-dimensional binary vector $v^M = (\delta_1^M, \delta_2^M, \cdots, \delta_{\ell-1}^M)$; furthermore, induced by the lexicographical precedence $0 \prec 1$, $M$ is said lexicographically larger than another matching $M'$ if the vector $v^M$ associated with $M$ is lexicographically larger than the vector $v^{M'}$ associated with $M'$ (*e.g.*, $(1, 1, 0, 1, 1, 1)$ is lexicographically larger than $(1, 0, 1, 1, 1, 1)$). Let $M^*$ be a lexicographically largest matching. By our assumption that $\overline{E}(U_i, U_{i+1}) \neq \varnothing$ for every $i \leqslant \ell - 1$, one sees that there are no adjacent 0's in the vector $v^{M^*}$.

**Lemma 7** *A lexicographically largest matching is a maximum matching in the graph $\overline{G} = (V, \overline{E})$.*

**Proof** By contradiction, we assume $M^*$ is a lexicographically largest matching, $M$ is a maximum matching in the graph $\overline{G} = (V, \overline{E})$ such that it is the lexicographically largest among all maximum matchings, and that $v^{M^*} > v^M$. It follows that there is a level index $k \leqslant \ell - 1$ such that $\delta_i^{M^*} = \delta_i^M$ for all $i = 1, 2, \cdots, k-1$ while $1 = \delta_k^{M^*} > \delta_k^M = 0$. One then sees that the edges determined by the new vector $(\delta_1^{M^*}, \delta_2^{M^*}, \cdots, \delta_k^{M^*}, 0, \delta_{k+2}^M, \delta_{k+3}^M, \cdots, \delta_{\ell-1}^M)$ form into a matching of size at least $|M|$ and lexicographically larger than $M$, a contradiction to the selection of $M$. □

**Lemma 8** *A lexicographically largest matching in the graph $\overline{G} = (V, \overline{E})$ can be computed in $O(n^2)$ time.*

**Proof** Given an edge $(j_1, j_2)$ of $\overline{E}(U_1, U_2)$, an edge of $\overline{E}(U_2, U_3)$ that can co-exist with $(j_1, j_2)$ in a matching has the form $(j_2', j_3)$ with $j_2' \neq j_2$; and we may interpret this discovery process as graph directed traversal from the edge $(j_1, j_2)$, along the artificial edge $(j_2, j_2')$, to the edge $(j_2', j_3)$. It follows that discovering the longest prefix of all 1's for the binary vector $v^{M^*}$ associated with a lexicographically largest matching $M^*$ is equivalent to finding the longest path starting with an edge of $\overline{E}(U_1, U_2)$ by the graph directed traversal, which takes $O(n^2)$ time. Assuming the length of this prefix is $k$, then $\delta_{k+1}^{M^*} = 0$ and we continue to discover the second longest chunk of all 1's for the binary vector $v^{M^*}$, by finding the longest path starting with an edge of $\overline{E}(U_{k+2}, U_{k+3})$ through the graph directed traversal. Since the edges explored during the second graph directed traversal do not overlap with those explored during the first traversal, we conclude that the total time for computing the lexicographically largest matching $M^*$ is $O(n^2)$, where $n = |V|$. □

**Lemma 9** *A lexicographically largest matching $M^*$ in the graph $\overline{G} = (V, \overline{E})$ gives a job processing order for each layer of $\mathcal{L} = \{U_1, U_2, \cdots, U_\ell\}$, which together, in $O(n+m)$ time, form a feasible schedule of makespan no greater than $n + (m-1)\ell - |M^*|$.*

**Proof** Recall that we are constructing a permutation and no-wait schedule. Using the layer representation $\mathcal{L} = \{U_1, U_2, \cdots, U_\ell\}$, we decide a processing sequence for the jobs of each layer according to the lexicographically largest matching $M^*$. The key idea is: For an edge $(j_1, j_2)$ of $M^* \cap \overline{E}(U_i, U_{i+1})$, we put the job $j_1$ as the last in the processing sequence for $U_i$ and put the job $j_2$ as the first in the processing sequence for $U_{i+1}$. The other jobs of each layer, if any, are arbitrarily ordered in between the decided the first and the last jobs. For the edge $(j_1, j_2)$ of $M^* \cap \overline{E}(U_i, U_{i+1})$, since $j_1$ and $j_2$ are agreeable, and we process $j_2$ on the first machine during the same time processing $j_1$ on the last machine. That is, the sub-schedules for $U_i$ and $U_{i+1}$ overlap exactly one time unit. On the other hand, if $M^* \cap \overline{E}(U_i, U_{i+1}) = \varnothing$, then the jobs of $U_{i+1}$ are started processing after all the jobs of $U_i$ are finished, that is, the two sub-schedules do not overlap. Therefore, an edge of $M^*$ essentially "saves" one time unit for the last machine in the flow-shop, and the schedule is constructed in $O(n+m)$ time.

Since the time-span for processing the jobs of the layer $U_i$ is $|U_i|+m-1$, the makespan of the constructed schedule $\pi$ using $M^*$ is thus

$$C_{\max} = \sum_{i=1}^{\ell}(|U_i| + m - 1) - |M^*| = n + (m-1)\ell - |M^*|. \tag{4}$$

This finishes the proof of the lemma. □

The above analysis motivates the second step of the algorithm Approx 2, which is to compute a lexicographically largest matching $M^*$ in the graph $\overline{G} = (V, \overline{E})$, and then to construct a feasible permutation and no-wait schedule. From Lemmas 8 and 9, this second step takes $O(n^2+m)$ time. A high-level description of the algorithm Approx 2 is depicted in Fig. 4.

Algorithm Approx 2:

0. Initialization: Preprocess the spine precedence graph $G = (V, E)$ for
   - 0.1 the layered representation $\mathcal{L} = \{U_1, U_2, \cdots, U_\ell\}$;
   - 0.2 assume no $U_i$ fully precedes $U_{i+1}$, or otherwise decompose the instance;
1. Construct the auxiliary graph $\overline{G} = (V, \overline{E})$;
2. Compute a lexicographically largest matching $M^*$ in $\overline{G}$, then
   - 2.1 construct a permutation and no-wait feasible schedule $\pi$.

**Fig. 4** A high-level description of the algorithm Approx 2

We have showed the makespan of the schedule achieved by the algorithm Approx 2 in Eq. (4). For the approximation ratio, we will prove next an improved lower bound on the minimum makespan using the matching $M^*$ in Eq. (5).

Recall that there are no adjacent 0's in the binary vector $v^{M^*}$ associated with the lexicographically largest matching $M^*$, by the assumption that no $U_i$ fully precedes $U_{i+1}$ for any $i \leqslant \ell - 1$. We next show that a 0-entry in the vector $v^{M^*}$ also implies some interesting local structure in the graph $\overline{G} = (V, \overline{E})$, similar to the fully precedence.

**Lemma 10** *Let $M^*$ be a lexicographically largest matching in the graph $\overline{G} = (V, \overline{E})$ and $\delta_i^{M^*} = 0$. Then all the edges of $\overline{E}(U_i, U_{i+1})$ are incident at a common vertex $c_i \in U_i$ (that is, $U_i \setminus \{c_i\}$ fully precedes $U_{i+1}$), and $c_i$ is covered by an edge of $M^*$.*

**Proof** From $M^*$ being a lexicographically largest matching and $\delta_i^{M^*} = 0$, we conclude that $i > 1$. The proof is then done by contradiction, where one sees that if there are edges of $\overline{E}(U_i, U_{i+1})$ incident at two distinct vertices of $U_i$, or if $c_i$ is not covered by an edge of $M^*$, then one edge of $\overline{E}(U_i, U_{i+1})$ can be added to $M^*$ (by possibly removing the edge of $M^* \cap \overline{E}(U_{i+1}, U_{i+2})$ from $M^*$), suggesting $M^*$ wouldn't be the lexicographically largest. □

Let $M^*$ be a lexicographically largest matching in the graph $\overline{G} = (V, \overline{E})$ and $\delta_i^{M^*} = 0$. Lemma 10 states the existence of a vertex $c_i \in U_i$ such that $U_i \setminus \{c_i\}$ fully precedes $U_{i+1}$. We next consider an optimal permutation and no-wait schedule $\pi^*$, and let $b_i$ be the last processed job among those of $U_i \setminus \{c_i\}$ and $a_{i+1}$ be the first processed job among those of $U_{i+1}$ in $\pi^*$. Since $b_i$ precedes $a_{i+1}$, their start processing times on the last machine are at least $m$ time units apart (Lemma 6), and we denote the time interval on the last machine from $b_i$ being finished to $a_{i+1}$ being started as $I_i$. By transitivity of the precedence relation, $U_i \setminus \{c_i\}$ fully precedes each of $U_{i+1}, U_{i+2}, \cdots, U_\ell$, and thus no job of $U_{i+1} \cup U_{i+2} \cup \cdots \cup U_\ell$ will be processed inside $I_i$, suggesting the defined intervals $I_i$'s for 0-entries in the vector $v^{M^*}$ are non-overlapping.

**Lemma 11** *In the optimal schedule $\pi^*$, the last machine idles for at least $m-2$ time units inside the time interval $I_i$, where $m \geqslant 3$ is the number of machines in the flow-shop.*

**Proof** We assume to the contrary that the last machine idles for $z$ time units inside the time interval $I_i$, with $z \leqslant m-3$. Recall that $\pi^*$ is permutation and no-wait. We first claim that the $|I_i| - z$ jobs processed inside the time interval $I_i$ all precede $c_i$ and agree with $U_i \setminus \{c_i\}$. Note that $|I_i| - z \geqslant (m-1) - (m-3) = 2$.

To prove the claim, first from $b_i$ being the last processed job among those of $U_i \setminus \{c_i\}$, no predecessor of any job of $U_i \setminus \{c_i\}$ can be processed inside $I_i$. That is, these $|I_i| - z$ jobs are agreeable with $U_i \setminus \{c_i\}$, and thus can be either $c_i$ or predecessors of $c_i$. If $c_i$ is among them, then $c_i$ would be agreeable with at least $m-2-z$ other jobs of them, a contradiction to their precedence relationship. Therefore, all these $|I_i| - z$ jobs precede $c_i$ (and, each is agreeable with at least $m-2-z$ others).

Recall the important fact about a spine graph is that every job of $U_i$ has a predecessor in $U_{i'}$, for any $i' < i$, and the other way around that every job of $U_{i'}$ has a successor in $U_i$. Given that $|I_i| - z \geqslant 2$, we conclude there is an index $i' < i$ such that $U_{i'}$ contains two jobs that are agreeable with $U_i \setminus \{c_i\}$. Denote these two jobs as $c_{i'}^1$ and $c_{i'}^2$. We further assume $i'$ is the largest such index, that is, for every $i'' > i'$, $U_{i'}$ contains only one job, denoted as $c_{i''}$, that is agreeable with $U_i \setminus \{c_i\}$. One sees that $c_{i''}$ precedes $c_{i''+1}$ for $i'' = i'+1, i'+2, \cdots, i-1$, and both $c_{i'}^1$ and $c_{i'}^2$ precede $c_{i'+1}$.

On the other hand, let $b_{i''}$ be a job of $U_{i''}$ preceding $b_{i''+1}$, for $i'' = i'+1, i'+2, \cdots, i-1$. By the transitivity of the precedence relation, $c_{i''}$ is agreeable with $b_{i''+1}$, for $i'' = i' + 1, i' + 2, \cdots, i-1$, and both $c_{i'}^1$ and $c_{i'}^2$ are agreeable with $b_{i'+1}$. That is, $(c_{i''}, b_{i''+1}) \in \overline{E}(U_{i''}, U_{i''+1})$, for $i'' = i'+1, i'+2, \cdots, i-1$, and $(c_{i'}^1, b_{i'+1}), (c_{i'}^2, b_{i'+1}) \in \overline{E}(U_{i'}, U_{i'+1})$.

Let $k$ be the largest index among $\{i', i'+1, \cdots, i-1\}$ such that $c_k$ is not covered by an edge of $M^*$, where $c_{i'}$ refers to either $c_{i'}^1$ or $c_{i'}^2$. Due to the two jobs $c_{i'}^1$ and $c_{i'}^2$, $k$ is well defined. We then for every $i'' = k, k+1, \cdots, i-1$, replace the edge of $M^* \cap \overline{E}(U_{i''}, U_{i''+1})$ by the edge $(c_{i''}, b_{i''+1})$, followed by adding an edge of $\overline{E}(\{c_i\}, U_{i+1})$ to $M^*$. This process makes $M^*$ lexicographically larger, contradicting to the assumption that the original $M^*$ is the lexicographically largest.

This proves the lemma that the last machine idles for at least $m-2$ time units inside the interval $I_i$. $\square$

**Theorem 3** *The lexicographically largest matching based algorithm* Approx 2 *is an $O(n^2+m)$-time $(2-\frac{2}{m})$-approximation for the problem $F \mid spine, p_{ij}=1 \mid C_{\max}$, where $m \geqslant 3$ is the number of machines in the flow-shop, and the approximation ratio is tight.*

**Proof** Recall that in the second step of the algorithm Approx 2, a lexicographically largest matching $M^*$ in the graph $\overline{G} = (V, \overline{E})$ is computed in $O(n^2)$ time, and from $M^*$ a feasible schedule $\pi$ is constructed in $O(n+m)$ time which has a makespan $C_{\max} \leqslant n + (m-1)\ell - |M^*|$. The total running time is thus $O(n^2 + m)$.

By Lemma 11 and the fact that the defined time intervals $I_i$'s do not overlap, we have another lower bound of the minimum makespan as

$$C^*_{FS} \geqslant n + m - 1 + (m-2)(\ell - 1 - |M^*|). \tag{5}$$

Using these lower bounds in Eqs. (1) and (5), we have

$$C_{\max} \leqslant C^*_{FS} + (m-3)|M^*| + (\ell-1) \leqslant C^*_{FS} + (m-2)(\ell-1) < C^*_{FS} + \frac{m-2}{m}C^*_{FS} = (2 - \frac{2}{m})C^*_{FS}.$$

That is, Approx 2 is a $(2-\frac{2}{m})$-approximation algorithm for the problem $F \mid spine, p_{ij} = 1 \mid C_{\max}$.

To prove the tightness of the ratio $2-\frac{2}{m}$, consider the instance shown in Fig. 5, in which there are $m$ machines in the flow-shop, $n = (2k+2)m$ jobs, and the precedence graph is a spine graph containing $2k+1$ levels. In the constructed auxiliary graph a lexicographically largest matching contains $2k$ edges (for example, $M^* = \{(im+1, (i+1)m), i = 1, 2, \cdots, 2k\}$) resulting in a schedule with makespan $n + (m-1)(2k+1) - 2k = (4m-4)k + (3m-1)$. On the other hand, one sees that the identity permutation $(1, 2, \cdots, n)$ gives rise to a feasible no-wait schedule to process all the $n$ jobs consecutively (and thus optimal), for which the makespan is $n + (m-1) = 2mk + (3m-1)$. It follows that the performance ratio of the

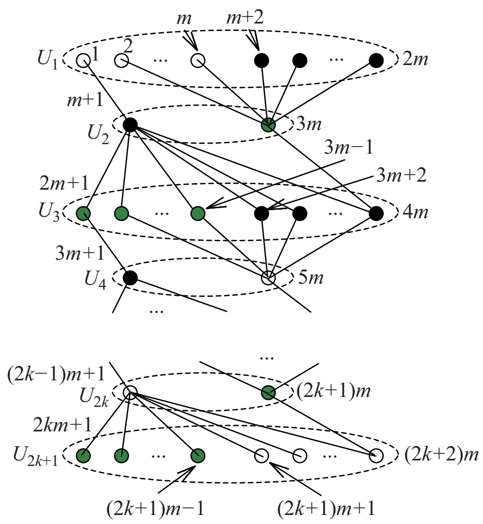


**Fig. 5** The spine precedence graph $G = (V, E)$ of the instance showing the tightness of the ratio: Each layer $U_i$ is indicated by a dashed oval and the directions of all the edges are downwards

algorithm Approx 2 on this instance is

$$\frac{(4m-4)k+(3m-1)}{2mk+(3m-1)} \longrightarrow 2-\frac{2}{m},$$

when $k$ (or equivalently, $n=(2k+2)m$) approaches $+\infty$. □

## 4 Concluding remarks

We studied the precedence constrained scheduling of unit jobs on an open-shop and a flow-shop, in which the number $m$ of machines is part of the input. Both problems are strongly NP-hard[14, 13]. We observed the jobs on the spine of the precedence graph and characterized improved lower bounds on the minimum makespan in terms of the number of agreeable pairings among certain jobs. We then presented a maximum matching based $(2-\frac{2}{m})$-approximation algorithm for $O \mid prec, p_{ij}=1 \mid C_{\max}$ and for $F \mid spine, p_{ij}=1 \mid C_{\max}$, respectively. The performance ratios are shown tight. The complexity of the problem $F \mid spine, p_{ij}=1 \mid C_{\max}$ is unknown yet, but it seems to be NP-hard as we see from the algorithm design and the tight instance that an exact $m$-set cover seems to be involved. We nevertheless leave it as an open question. One thus might wonder whether a similar conclusion to the multiprocessor scheduling[8] holds, that it is NP-hard to approximate either problem within a constant factor strictly less than 2.

## References


[1] Pinedo M. *Scheduling: Theory, Algorithm and Systems* (*5th Ed.*) [M]. New York: Springer, 2016.

[2] Graham R L. Bounds for certain multiprocessing anomalies [J]. *Bell Labs Technical Journal*, 1966, **45**: 1563-1581.

[3] Graham R L, Lawler E L, Lenstra J K, et al. Optimization and approximation in deterministic sequencing and scheduling: A survey [J]. *Annuals of Discrete Mathematics*, 1979, **5**: 287-326.

[4] Lam S, Sethi R. Worst case analysis of two scheduling algorithms [J]. *SIAM Journal on Computing*, 1977, **6**: 518-536.

[5] Braschi B, Trystram D. A new insight into the Coffman-Graham algorithm [J]. *SIAM Journal on Computing*, 1994, **23**: 662-669.

[6] Gangal D, Ranade A. Precedence constrained scheduling in $(2-\frac{7}{3p+1})$·optimal [J]. *Journal of Computer and System Sciences*, 2008, **74**: 1139-1146.

[7] Bansal N, Khot S. Optimal long code test with one free bit [C]//*Proceedings of FOCS 2009*, 2009: 453-462.

[8] Svensson O. Conditional hardness of precedence constrained scheduling on identical machines [C]//*Proceedings of STOC 2010*, 2010: 745-754.

[9] Garey M R, Johnson D S. *Computers and Intractability: A Guide to the Theory of NP-Completeness* [M]. San Francisco: W. H. Freeman & Co., 1979.

[10] Schuurman P, Woeginger G J. Polynomial time approximation algorithms for machine scheduling: Ten open problems [J]. *Journal of Scheduling*, 1999, **2**: 103-213.

[11] Levey E, Rothvoss T. A $(1+\varepsilon)$-approximation for makespan scheduling with precedence constraints using LP hierarchies [C]//*Proceedings of SODA 2016*, 2016: 168-177.

[12] Prot D, Bellenguez-Morinea O. A survey on how the structure of precedence constraints may change the complexity class of scheduling problems [J]. *Journal of Scheduling*, 2018, **21**: 3-16.

[13] Leung J Y T, Vornberger O, Witthoff J D. On some variants of the bandwidth minimization problem [J]. *SIAM Journal on Computing*, 1984, **13**: 650-667.

[14] Timkovsky V G. Identical parallel machines vs. unit-time shops and preemptions vs. chains in scheduling complexity [J]. *European Journal of Operational Research*, 2003, **149**: 355-376.

[15] Brucker P, Jurisch B, Jurisch M Z. Open shop problems with unit time operations [J]. *Operations Research*, 1993, **37**: 59-73.

[16] Bruno J, Jones III J W, So K. Deterministic scheduling with pipelined processors [J]. *IEEE Transactions on Computers*, 1980, **29**: 308-316.

[17] Bräsel H, Kluge D, Werner F. A polynomial algorithm for the $[n/m/0, t_{ij} = 1, \text{tree}/C_{\max}]$ open shop problem [J]. *European Journal of Operational Research*, 1994, **72**: 125-134.

[18] Brucker P, Knust S. Complexity results for single-machine problems with positive finish-start time-lags [J]. *Computing*, 1999, **63**: 299-316.

[19] Chen Y, Goebel R, Lin G, et al. Open-shop scheduling for unit jobs under precedence constraints. *Theoretical Computer Science*, 2020, **803**: 144-151.

[20] Tanaev V S, Sotskov Y N, Strusevich V A. *Scheduling Theory: Multi-Stage Systems* [M]. Netherlands: Springer, 1994.

[21] Hopcroft J E, Karp R M. An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs [J]. *SIAM Journal on Computing*, 1973, **2**: 225-231.

[22] Karzanov A V. O nakhozhdenii maksimal'nogo potoka v setyakh spetsial'nogo vida i nekotorykh prilozheniyakh (in Russian; title translation: On finding maximum flows in networks with special structure and some applications) [J]. *Matematicheskie Voprosy Upravleniya Proizvodstvom*, 1973, **5**: 81-94.

[23] Madry A. Navigating central path with electrical flows: From flows to matchings, and back [C]//*Proceedings of FOCS 2013*, 2013: 253-262.

[24] Potts C N, Shmoys D B, Williamson D P. Permutation vs. non-permutation flow shop schedules [J]. *Operations Research Letters*, 1991, **10**: 281-284.