Combinations of Some Shop Scheduling Problems and the Shortest Path Problem: Complexity and Approximation Algorithms

Kameng Nip, Zhenbo Wang(✉), and Wenxun Xing

Department of Mathematical Sciences, Tsinghua University, Beijing, China
zwang@math.tsinghua.edu.cn

Abstract. We study several combinatorial optimization problems which combine the classic shop scheduling problems (open shop scheduling or job shop scheduling) and the shortest path problem. The objective of the considered problem is to select a subset of jobs that forms a feasible solution of the shortest path problem, and to execute the selected jobs on the open shop or job shop machines such that the makespan is minimized. We show that these problems are NP-hard even if the number of machines is two, and they cannot be approximated within a factor of less than 2 if the number of machines is an input unless P = NP. We present several approximation algorithms for these problems.

Keywords: Approximation algorithm · Combination of optimization problems · Job shop · Open shop · Scheduling · Shortest path

1 Introduction

Combinatorial optimization involves many active subfields, e.g. network flows, scheduling, bin packing. Usually these subfields are motivated by various applications or theoretical interests, and separately developed. The development of science and technology makes it possible to integrate manufacturing, service and management. At the same time, the decision-makers always need to deal with problems involving more than one combinatorial optimization problems. For instance, the network monitoring scenario described in [17] and the railway manufacturing scenario [12].

Wang and Cui [17] introduced a problem combining two classic combinatorial optimization problems, namely parallel machine scheduling and the vertex cover problem. The combination problem is to select a subset of jobs that forms a vertex cover, and to schedule it on some identical parallel machines such that the makespan is minimized. This work also inspired the study of the combination of different combinatorial optimization problems.

Flow shop, open shop and job shop are three basic models of multi-stage scheduling problems. Nip and Wang [12] studied a combination problem that combines two-machine flow shop scheduling and the shortest path problem.

D. Xu et al. (Eds.): COCOON 2015, LNCS 9198, pp. 97–108, 2015.
DOI: 10.1007/978-3-319-21398-9_8

They argued that this problem is NP-hard, and proposed two approximation algorithms with worst-case ratio 2 and $\frac{3}{2}$ respectively. Recently, Nip et al. [13] extended the results to the case that the number of flow shop machines is arbitrary. One motivation of this problem is manufacturing rail racks. We plan to build a railway between two cities. How should we choose a feasible path in a map, such that the corresponding rail tracks (jobs) can be manufactured on some shop machines as early as possible? Similar scenarios can be found in telecommunications and other transportation industries. It connects two classic combinatorial optimization problems, say shop scheduling and the shortest path problem. An intuitive question is what will happen if the shop environment is one of the other two well-known shop environments, i.e. open shop and job shop. This is the core motivation for this current work. In this paper, we mainly study two problems: the combination of open shop scheduling and the shortest path problem, and the combination of job shop scheduling and the shortest path problem.

The contributions of this paper are described as follows: (1) we argue that these combination problems are NP-hard even if the number of machines is two, and if the number of machines is an input, these problems cannot be approximated within a factor less than 2 unless P = NP; (2) we present several approximation algorithms with performance ratio summarized as follows in which $\epsilon > 0$ is any constant and μ is the maximum operations per job in job shop scheduling.

Table 1. Performance of our algorithms

Number of Machines	Open Shop	Job Shop
2	FPTAS	$\frac{3}{2} + \epsilon$*
m (fixed)	PTAS**	$O\left(\frac{\log^2(m\mu)}{\log\log(m\mu)}\right)$
m (input)	m	m

* Assume that each job has at most 2 operations.
** A $(2+\epsilon)$-approximation algorithm is also proposed.

The rest of the paper is organized as follows. In Section 2, we give a formal definition of the combination problems stated above, and briefly review some related problems and algorithms that will be used subsequently. In Section 3, we study the computational complexity of these combination problems and give an inapproximability result when the number of machines is an input. Section 4 provides several approximation algorithms for these problems. Some concluding remarks are provided in Section 5.

2 Preliminaries

2.1 Problem Description

We first recall the definitions of open shop and the job shop scheduling problems in the literatures.

Given a set of n jobs $J = \{J_1, \cdots, J_n\}$ and m machines $M = \{M_1, \cdots, M_m\}$, each job has several operations. At the same time, each machine can process at most one job and each job can be processed on one machine. In the open shop scheduling problem ($Om||C_{\max}$), each job must be processed on each machine exactly once, but the processing order can be arbitrary (in other words, the sequence of machines through which job passes can differ between jobs). In the job shop scheduling ($Jm||C_{\max}$), the processing order of each job is given in advance, and may differ between jobs. Furthermore, each job is allowed to be processed on the same machine more than once but consecutive operations of the same job must be processed on different machines, and is not necessary to go through all machines in the job shop. The goal of $Om||C_{\max}$ or $Jm||C_{\max}$ is to find a feasible schedule such that the makespan, that is, the completion time of the last stage among all the jobs is minimum.

Now we define the combination problems considered in this paper.

Definition 1 ($Om|$shortest path$|C_{\max}$)**.** *Given a directed graph $G = (V, A)$ with two distinguished vertices $s, t \in V$, and m machines. Each arc $a_j \in A$ corresponds to a job $J_j \in J$. The $Om|$*shortest path*$|C_{\max}$ problem is to find an $s-t$ directed path P of G, and to schedule the jobs of J_P on the open (job) shop machines such that the minimum makespan over all P, where J_P denotes the set of jobs corresponding to the arcs in P.*

Definition 2 ($Jm|$shortest path$|C_{\max}$)**.** *Given a directed graph $G = (V, A)$ with two distinguished vertices $s, t \in V$, and m machines. Each arc $a_j \in A$ corresponds to a job $J_j \in J$. The $Jm|$*shortest path*$|C_{\max}$ problem is to find an $s-t$ directed path P of G, and to schedule the jobs of J_P on the job shop machines such that the minimum makespan over all P, where J_P denotes the set of jobs corresponding to the arcs in P.*

Let the number of jobs (arcs) be n, i.e. $|A| = |J| = n$. Let p_{ij} be the processing times for J_j on machine M_i, and μ_{ij} be the frequency of J_j processed on M_i. Notice $\mu_{ij} = 1$ in the open shop.

It is not difficult to see that the shop scheduling problem (open shop or job shop) and the classic shortest path problem are special cases of our problems. For example, consider the following instances with $m = 2$ ([13]). If there is a unique path from s to t in G, as shown in the left of Fig. 1, our problem is the two-machine shop scheduling problem (open shop or job shop). If all the processing times on the second machine are zero, as shown in the right of Fig. 1, then our problem is equivalent to the classic shortest path with respect to the processing times on the first machine. Therefore we say the considered problems are the combinations of the shop scheduling problems and the shortest path problem.

In this paper, we will use the results of some optimization problems that have a similar structure to the classic shortest path problem. We introduce the generalized shortest path problem defined in [13].

Definition 3. *Given a weighted directed graph $G = (V, A, w^1, \cdots, w^K)$ and two distinguished vertices $s, t \in V$ with $|A| = n$, each arc $a_j \in A, j = 1, \cdots, n$ is*

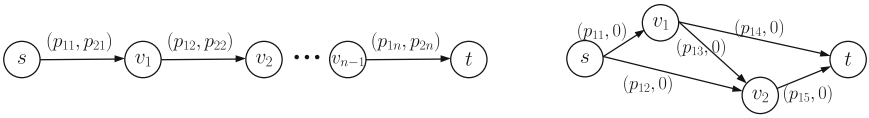

Fig. 1. Special cases of our problems

associated with K weights $w_j^1, \cdots, w_j^K$*, and we define vector* $w^k = (w_1^k, w_2^k, \cdots, w_n^k)$ *for* $k = 1, 2, \cdots, K$*. The goal of our shortest path problem* $SP(G, s, t, f)$ *is to find an* $s - t$ *directed path* P *that minimizes* $f(w^1, w^2, \cdots, w^K; x)$*, in which* f *is a given objective function and* $x \in \{0, 1\}^n$ *contains the decision variables such that* $x_j = 1$ *if and only if* $a_j \in P$*.*

For simplicity of notation, we denote SP instead of $SP(G, s, t, f)$ in the rest of the paper. Notice SP is a generalization of various shortest path problems. For example, if we set $K = 1$ and $f(w^1, x) = w^1 \cdot x$, where $\cdot$ is the dot product, it is the classic shortest path problem. If $f(w^1, w^2, \cdots, w^K; x) = \max\{w^1 \cdot x, w^2 \cdot x, \cdots, w^K \cdot x\}$, it is the min-max shortest path problem [1].

2.2 Review of Open Shop and Job Shop Scheduling

Gonzalez and Sahni [5] first gave a linear time optimal algorithm for $O2||C_{\max}$. They also proved that $Om||C_{\max}$ is NP-hard for $m \geq 3$, however whether it is strongly NP-hard is still an outstanding open problem. A feasible shop schedule is called dense when any machine is idle if and only if there is no job that could be processed on it. Rácsmány (see Bárány and Fiala [2]) observed that for any dense schedule, the makespan is at most twice that of the optimal solution, which leads to a greedy algorithm. Sevastianov and Woeginger [15] presented a PTAS for fixed m, which is obtained by dividing jobs into large jobs and small jobs. Their algorithm first optimally schedules the large jobs, then fills the operations of the small jobs into the 'gaps'. In this paper, we will use these algorithms, and refer to them as the GS algorithm, Rácsmány algorithm and the SW algorithm respectively. We present the main results of these algorithms as follows.

Theorem 1 ([5]). *The GS algorithm returns an optimal schedule for* $O2||C_{\max}$ *in linear time such that* $C_{\max} = \max\left\{\max_{J_j \in J}(p_{1j} + p_{2j}), \sum_{J_j \in J} p_{1j}, \sum_{J_j \in J} p_{2j}\right\}$.

Theorem 2 ([2,16]). *Rácsmány algorithm returns a 2-approximation algorithm for* $Om||C_{\max}$ *such that* $C_{\max} \leq \sum_{J_j \in J} p_{lj} + \sum_{i=1}^{m} p_{ik} \leq 2C^*_{\max}$*, where* J_k *is the last completed job and it is processed on* M_l*, and* $C^*_{\max}$ *denotes the optimal makespan.*

Theorem 3 ([15]). *The SW algorithm is a PTAS for* $Om||C_{\max}$*.*

For job shop scheduling problems, few polynomially solvable cases are known. One is $J2|op \leq 2|C_{\max}$, which can be solved by Jackson's rule [6] that is an

extension of Johnson's rule for $F2||C_{\max}$ (flow shop scheduling problem with two machines [8]), where $op \leq 2$ means there are at most 2 operations per job.

In fact, a slight change may lead to NP-hard problems. For instance, $J2|op \leq 3|C_{\max}$ and $J3|op \leq 2|C_{\max}$ are NP-hard [9], $J2|p_{ij} \in \{1,2\}|C_{\max}$ and $J3|p_{ij} = 1|C_{\max}$ are strongly NP-hard [10]. For the general case $J||C_{\max}$, Shmoys, Stein and Wein [16] constructed a randomized approximation algorithm with worst-case ratio $O\left(\frac{\log^2(m\mu)}{\log\log(m\mu)}\right)$, where μ is the maximum number of operations per job. Schmidt, Siegel and Srinivasan [14] obtained a deterministic algorithm with the same bound by derandomizing. We refer to it as the SSW-SSS algorithm. Moreover, for fixed m, the best known approximation algorithm is also proposed in [16] with an approximation factor $2+\epsilon$, where $\epsilon > 0$ is an arbitrary constant. If μ is a constant, the problem is denoted as $Jm|op \leq \mu|C_{\max}$ and admits a PTAS [7]. We list the main results mentioned above as follows.

Theorem 4 ([6]). *Jackson's rule solves $J2|op \leq 2|C_{\max}$ in $O(n \log n)$ time.*

Theorem 5 ([14,16]). *The SSW-SSS algorithm solves $Jm||C_{\max}$ in polynomial time, and returns a schedule with makespan*

$$O\left(\frac{\log^2(m\mu)}{\log\log(m\mu)}\left(\max_{i\in\{1,\cdots,m\}}\sum_{J_j\in J}\mu_{ij}p_{ij} + \max_{J_j\in J}\sum_{i=1}^{m}\mu_{ij}p_{ij}\right)\right).$$

Furthermore, a well-known inapproximability result is that $O||C_{\max}$, $F||C_{\max}$ and $J||C_{\max}$ cannot be approximated within $\frac{5}{4}$ unless P = NP [18]. Recently, Mastrolilli and Svensson [11] showed that $J||C_{\max}$ cannot be approximated within $O(\log(m\mu)^{1-\epsilon})$ for $\epsilon > 0$ based on a stronger assumption than P $\neq$ NP.

To conclude this subsection, we list some trivial bounds for a dense shop schedule. Denote by $C_{\max}$ the makespan of an arbitrary dense shop schedule with job set J, and we have

$$C_{\max} \geq \max_{i\in\{1,\cdots,m\}}\left\{\sum_{J_j\in J}\mu_{ij}p_{ij}\right\}, \tag{1}$$

and

$$C_{\max} \leq \sum_{J_j\in J}\sum_{i=1}^{m}\mu_{ij}p_{ij}. \tag{2}$$

For each job, we have

$$C_{\max} \geq \sum_{i=1}^{m}\mu_{ij}p_{ij}, \qquad \forall J_j \in J. \tag{3}$$

2.3 Review of Shortest Path Problems

It is well-known that Dijkstra algorithm solves the classic shortest path problem with nonnegative edge weights in $O(|V|^2)$ time [3]. We have mentioned the min-max shortest path problem, that is NP-hard even for $K = 2$, and Aissi, Bazgan and Vanderpooten proposed an FPTAS if K is a fixed number [1]. We refer to their algorithm as the ABV algorithm, which has the following result.

Theorem 6 ([1]). *Given $\epsilon > 0$, in a directed graph with K nonnegative weights on each arc, where K is a fixed number, the ABV algorithm finds a path P between two specific vertices satisfying $\max_{i\in\{1,2,\cdots,K\}} \left\{\sum_{a_j\in P} w_j^i\right\} \leq (1+\epsilon) \max_{i\in\{1,2,\cdots,K\}} \left\{\sum_{a_j\in P'} w_j^i\right\}$ for any path P' between the two specified vertices, and the running time is $O(|A||V|^{K+1}/\epsilon^K)$.*

In this paper, sometimes we need to find the min-max shortest path among all the paths visiting some specified arcs if such a path exists. We propose a modified ABV algorithm for this problem, which will be involved in the complete version.

3 Computational Complexity

First, notice that $Om||C_{\max}$ and $Jm||C_{\max}$ are special cases of the corresponding combination problems, thus the combination problem is at least as hard as its component optimization problems. On the other hand, we know that $O2||C_{\max}$ and $J2|op \leq 2|C_{\max}$ are polynomially solvable. However, we can simply verify that the corresponding combination problems, say $O2|$shortest path$|C_{\max}$ and $J2|op \leq 2,$ shortest path$|C_{\max}$, are NP-hard by adopting the same reduction proposed in Theorem 2 of [12] for the NP-hardness of $F2|$shortest path$|C_{\max}$. We summarize the results as Theorem 7.

Theorem 7. *$J2|$*shortest path*$|C_{\max}$ is strongly* NP*-hard; $O2|$*shortest path*$|C_{\max}$ and $J2|op \leq 2,$* shortest path*$|C_{\max}$ are* NP*-hard.*

Now we consider the case where the number of machines m is part of the input. Williamson et al.[18] showed that it is NP-hard to approximate $O||C_{\max}$, $F||C_{\max}$ or $J||C_{\max}$ within a factor less than $\frac{5}{4}$ by a reduction from the restricted versions of 3-SAT. They also showed that deciding if there is a scheduling of length at most 3 is in P. We show that for these problems combining with shortest path problem, deciding if there is a scheduling of length at most 1 is still NP-hard. Our proof is established by constructing a reduction from 3-Dimensional Matching (3DM) that is NP-complete [4].

Theorem 8. *For $O|$*shortest path*$|C_{\max}$, deciding if there is a scheduling of length at most 1 is* NP*-hard.*

Notice that the reduction in Theorem 8 is also valid for $F|$shortest path$|C_{\max}$ and $J|$shortest path$|C_{\max}$, since each job in the reduction has only one nonzero processing time. Therefore we have the following result.

Corollary 1. *The problems* $O|\text{shortest path}|C_{\max}$, $F|\text{shortest path}|C_{\max}$ *and* $J|\text{shortest path}|C_{\max}$ *do not admit an approximation algorithm with worst-case ratio less than* 2, *unless* P = NP.

To our knowledge, the best known inapproximability results based on P $\neq$ NP for $F||C_{\max}$, $O||C_{\max}$ and $J||C_{\max}$ are still $\frac{5}{4}$. The corollary implies that the combination problems of the three shop scheduling problems and the shortest path problem may have stronger inapproximability results than the original problems.

4 Approximation Algorithms

4.1 An Intuitive Algorithm for Arbitrary m

An intuitive algorithm was proposed for $F2|\text{shortest path}|C_{\max}$ in [12]. The idea is to find the classic shortest path by setting the weight of an arc to be the sum of processing times of its corresponding job, and then schedule the returned jobs by Johnson's rule. This simple idea can be extended to the combination problems we considered, even if the number of machines is an input.

Algorithm 1. The SD algorithm for $O|\text{shortest path}|C_{\max}$ ($J|\text{shortest path}|C_{\max}$)

1: Find the shortest path in G with weights $w_j^1 := \sum_{i=1}^{m} \mu_{ij} p_{ij}$ by Dijkstra algorithm. For the returned path P, construct the job set J_P.
2: Obtain a dense schedule for the jobs of J_P by an arbitrary open (job) shop scheduling algorithm. Let σ be the returned job schedule and $C_{\max}$ the returned makespan, and denote the job set J_P by S.
3: **return** S, σ **and** $C_{\max}$.

Theorem 9. *For* $O|\text{shortest path}|C_{\max}$ *and* $J|\text{shortest path}|C_{\max}$, *the SD algorithm is* m*-approximated, and this bound is tight.*

4.2 A Unified Algorithms for Fixed m

In [12], a $\frac{3}{2}$-approximation algorithm was proposed for $F2|\text{shortest path}|C_{\max}$. The idea is to iteratively find a feasible path by the ABV algorithm [1] with two weights for each arc, and schedule the corresponding jobs by Johnson's rule, then adaptively modify the weights of arcs and repeat the procedures until we obtain a feasible schedule with good guarantee. We generalize this idea to solve the combination problems considered in this paper. We first propose a unified framework which is denoted as UAR(Alg, ρ, m), where Alg is a polynomial time algorithm used for shop scheduling, ρ is a control parameter to decide the termination rule of the iterations and the jobs to be modified, and m is the number of machines. The pseudocode of the UAR(Alg, ρ, m)algorithm is described by Algorithm 2.

By setting the appropriate scheduling algorithms and control parameters, we can derive algorithms for different combination problems. Notice that at most n jobs are modified in the UAR(Alg, ρ, m) algorithm, therefore the iterations

Algorithm 2. Algorithm UAR(Alg, ρ, m)

1: Initially,$(w_j^1, w_j^2, \cdots, w_j^m) := (\mu_{1j}p_{1j}, \mu_{2j}p_{2j}, \cdots, \mu_{mj}p_{mj})$, for $a_j \in A$ corresponding to J_j.
2: Given $\epsilon > 0$, use the ABV algorithm [1] to obtain a feasible path P to SP, and construct the corresponding job set as J_P.
3: Schedule the jobs of J_P by the algorithm Alg, denote the returned makespan as $C'_{\max}$, and the job schedule as σ'.
4: $S := J_P$, $\sigma := \sigma'$, $C_{\max} := C'_{\max}$, $D := \emptyset$, $M := (1+\epsilon)\sum\limits_{J_j \in J}\sum\limits_{i=1}^{m}\mu_{ij}p_{ij} + 1$.
5: **while** $J_P \cap D = \emptyset$ **and** there exists J_j in J_P satisfying $\sum\limits_{i=1}^{m}\mu_{ij}p_{ij} \geq \rho C'_{\max}$ **do**
6: **for** all jobs satisfy $\sum\limits_{i=1}^{m}\mu_{ij}p_{ij} \geq \rho C'_{\max}$ in $J \backslash D$ **do**
7: $(w_j^1, w_j^2, \cdots, w_j^m) := (M, M, \cdots, M)$, $D := D \cup \{J_j\}$.
8: **end for**
9: Use the ABV algorithm [1] to obtain a feasible path P to SP, and construct the corresponding job set as J_P.
10: Schedule the jobs of J_P by the algorithm Alg, denote the returned makespan as $C'_{\max}$, and the job schedule as σ'.
11: **if** $C'_{\max} < C_{\max}$ **then**
12: $S := J_P$, $\sigma := \sigma'$, $C_{\max} := C'_{\max}$.
13: **end if**
14: **end while**
15: **return** S, σ **and** $C_{\max}$.

execute at most n times. Since the scheduling algorithms for shop scheduling and the ABV algorithm [1] are all polynomial time algorithms (for fixed m and ϵ), we claim that the following algorithms based on UAR(Alg, ρ, m) are polynomial-time algorithms. We present the algorithms and prove their performance as follows.

We first apply the UAR(Alg, ρ, m) algorithm to $O2|$shortest path$|C_{\max}$ by setting Alg be the GS algorithm [5] and $\rho = 1$. We refer to this algorithm as the GAR algorithm.

Algorithm 3. The GAR algorithm for $O2|$shortest path$|C_{\max}$

1: Let $m = 2$, Alg be the GS algorithm [5] for $O2||C_{\max}$ and $\rho = 1$.
2: Solve the problem by using UAR(Alg, ρ, m).

Theorem 10. *The GAR algorithm is an FPTAS for* $O2|$shortest path$|C_{\max}$.

We point out that the proofs of the worst-case performance of algorithms based on UAR(Alg, ρ, m) are quite similar. In the following proofs of this subsection, we will only describe the key ideas and main steps since the results can be obtained by analogous arguments. We will adopt the same notations as in the proof of Theorem 10, and also analyze the same two cases.

For $Om|$shortest path$|C_{\max}$ where m is fixed, we obtain the following RAR algorithm based on UAR(Alg, ρ, m) and Rácsmány algorithm [2,16].

Algorithm 4. The RAR algorithm for $Om|\text{shortest path}|C_{\max}$

1: Let Alg be Rácsmány algorithm [2,16] for $Om||C_{\max}$ and $\rho = \frac{1}{2}$.
2: Solve the problem by using UAR(Alg, ρ, m).

Theorem 11. *Given $\epsilon > 0$, the RAR algorithm is a $(2+\epsilon)$-approximation algorithm for $Om|\text{shortest path}|C_{\max}$.*

The framework can also be applied to the combination problem of job shop scheduling and the shortest path problem. For the combination of $J2|op \leq 2|C_{\max}$ and the shortest path problem, we obtain a $(\frac{3}{2}+\epsilon)$-approximation algorithm by using Jackson's rule and setting $\rho = \frac{2}{3}$ in the UAR(Alg, ρ, m) algorithm. We refer to this algorithm as the JJAR algorithm, and describe it in Algorithm 5. Recall that all $\mu_{ij} = 1$ in $J2|op \leq 2|C_{\max}$.

Algorithm 5. The JJAR algorithm for $J2|op \leq 2, \text{shortest path}|C_{\max}$

1: Let $m = 2$, Alg be Jackson's rule for $J2|op \leq 2|C_{\max}$ and $\rho = \frac{2}{3}$.
2: Solve the problem by using UAR(Alg, ρ, m).

Before studying the worst-case performance of the JJAR algorithm, we establish the following lemma. Let $(1 \to 2)$ $((2 \to 1))$ indicate the order that a job needs to be processed on M_1 (M_2) first and then on M_2 (M_1).

Lemma 1. *For $J2|op \leq 2|C_{\max}$, let $C^J_{\max}$ be the makespan returned by Jackson's rule. Suppose we change the processing order of all jobs to be $(1 \to 2)$ $((2 \to 1))$, and the processing times keep unchanged. Then schedule the jobs by Johnson's rule for $F2||C_{\max}$, and denote the makespan as $C^1_{\max}$ $(C^2_{\max})$. We have $C^J_{\max} \leq \max\{C^1_{\max}, C^2_{\max}\}$.*

Now we can study the performance of the JJAR algorithm for $J2|op \leq 2, \text{shortest path}|C_{\max}$.

Theorem 12. *Given $\epsilon > 0$, the JJAR algorithm is a $(\frac{3}{2}+\epsilon)$-approximation algorithm for $J2|op \leq 2, \text{shortest path}|C_{\max}$.*

Finally, we study the general case $Jm|\text{shortest path}|C_{\max}$, where m is fixed. By Theorem 5, we know that there exists $\alpha > 0$, such that the SSW-SSS algorithm [14,16] returns a schedule satisfying

$$C'_{\max} \leq \alpha \frac{\log^2(m\mu)}{\log\log(m\mu)} \left(\max_{i\in\{1,\cdots,m\}} \sum_{J_j \in J'} \mu_{ij} p_{ij} + \max_{j \in J'} \sum_{i=1}^{m} \mu_{ij} p_{ij} \right). \tag{4}$$

The factor α is decided by choosing the probability of the randomized steps and the subsequent operations in the SSW-SSS algorithm [14,16] [14,16], and its

value can be obtained by complicated calculation. Assume we determine such value of α. We can design an approximation algorithm with worst-case ratio $O\left(\frac{\log^2(m\mu)}{\log\log(m\mu)}\right)$ for $Jm|\text{shortest path}|C_{\max}$. We refer to this algorithm as the SAR algorithm, and describe it in Algorithm 6.

Algorithm 6. The SAR algorithm for $Jm|\text{shortest path}|C_{\max}$

1: Let *Alg* be the SSW-SSS algorithm [14,16] for $Jm||C_{\max}$ and $\rho = \frac{\log\log(m\mu)}{2\alpha\log^2(m\mu)}$.
2: Solve the problem by using UAR(Alg, ρ, m).

Theorem 13. *The SAR algorithm is an* $O\left(\frac{\log^2(m\mu)}{\log\log(m\mu)}\right)$*-approximation algorithm for* $Jm|\text{shortest path}|C_{\max}$.

Remind that the SAR algorithm relies on the assumption, that we can determine the constant α for the SSW-SSS algorithm [14,16]. We can calculate it by following the details of the SSW-SSS algorithm [14,16], and in fact we can choose α large enough to guarantee the performance ratio of our algorithm.

4.3 A PTAS for $Om|\text{shortest path}|C_{\max}$

In the previous subsection, we introduced a $(2+\epsilon)$-approximation algorithm for $Om|\text{shortest path}|C_{\max}$ based on the UAR(Alg, ρ, m) algorithm. By a different approach, we propose a $(1+\epsilon)$-approximation algorithm for any $\epsilon > 0$, i.e. a PTAS. We also iteratively find feasible solutions, but guarantee that one of the returned solutions has the same first N-th largest jobs with an optimal solution where N is a given constant. Precisely speaking, we say job J_j is larger than job J_k if $\max\limits_{i\in\{1,\cdots,m\}} p_{ij} > \max\limits_{i\in\{1,\cdots,m\}} p_{ik}$. To do this, we enumerate all size N subsets J^N of J, and then iteratively modify the weights of the graph such that the jobs larger than any job in J^N will not be chosen. Then find a feasible solution which contains all the jobs in J^N corresponding to the modified graph, i.e., the corresponding path is constrained to visit all the arcs corresponding to J^N if such a path exists.

To find a feasible solution in each iteration, we adopt the modified ABV algorithm to obtain a near optimal min-max shortest path among all the paths visiting the arcs corresponding to J^N if such a path exists. Then we schedule the selected jobs by [15] which is denoted as the SW algorithm [15] for $Om||C_{\max}$. We refer to our algorithm as the SAE algorithm, and describe it in Algorithm 7.

There are $\binom{n}{N}$ distinct subsets J^N, thus the iterations between line 4 - line 12 run at most $O(n^N)$ times, that is a polynomial of n since N is a constant when m and ϵ are fixed. Since the modified ABV algorithm is an FPTAS and the SW algorithm [15] is a PTAS, the running time of each iteration is also bounded by the polynomial of n if m and ϵ are fixed. It suffices to show that the SAE algorithm terminates in polynomial time. The following theorem indicates the SAE algorithm is a PTAS.

Algorithm 7. The SAE algorithm for $Om|\text{shortest path}|C_{\max}$

Given $0 < \epsilon < 1$, set $N = m\left(\frac{m(3+\epsilon)}{\epsilon}\right)^{2^{\frac{m(3+\epsilon)}{\epsilon}}}$.
Let $D := \emptyset$, $M := (1 + \frac{\epsilon}{3}) \sum_{J_j \in J} \sum_{i=1}^{m} p_{ij} + 1$, $C_{\max} := \sum_{J_j \in J} \sum_{i=1}^{m} p_{ij}$.
Initially, $(w_j^1, w_j^2, \cdots, w_j^m) := (p_{1j}, p_{2j}, \cdots, p_{mj})$, for $a_j \in A$ corresponding to J_j.
for all $J^N \subset J$, with $|J^N| = N$ **do**
  $(w_j^1, w_j^2, \cdots, w_j^m) := (p_{1j}, p_{2j}, \cdots, p_{mj})$, $D := \emptyset$.
  For jobs $J_k \in J \setminus J^N$ with $\max_{i \in \{1,\cdots,m\}} p_{ik} > \min_{J_j \in J^N} \max_{i \in \{1,\cdots,m\}} p_{ij}$, set $(w_k^1, w_k^2, \cdots, w_k^m) := (M, M, \cdots, M)$, $D := D \cup \{J_k\}$.
  Use the modified ABV algorithm to obtain a feasible path P of SP such that the returned path visits all the arcs corresponding to J^N if such a path exists. Construct the corresponding job set as J_P.
  Schedule the jobs of J_P by the SW algorithm [15], denote the returned makespan as $C'_{\max}$, and the job schedule as σ'.
  if $C'_{\max} < C_{\max}$ **then**
    $S := J_P$, $\sigma := \sigma'$, $C_{\max} := C'_{\max}$.
  end if
end for
return S, σ, $C_{\max}$.

Theorem 14. *The SAE algorithm is a PTAS for* $Jm|\text{shortest path}|C_{\max}$.

5 Conclusions

This paper studies several problems combining two well-known combinatorial optimization problems. We show the hardness of the problems, and present some approximation algorithms. It is interesting to find approximation algorithms with better worst-case ratios for $J2|op \leq 2, \text{shortest path}|C_{\max}$ and $Jm|\text{shortest path}|C_{\max}$. Moreover, it needs further study to close the gap between the 2-inapproximability results and the m-approximation algorithms for $O|\text{shortest path}|C_{\max}$ and $J|\text{shortest path}|C_{\max}$. We can also consider other interesting combinations of combinatorial optimization problems.

Acknowledgments. Wang's research has been supported by NSFC No. 11371216 and Bilateral Scientific Cooperation Project between Tsinghua University and KU Leuven. Xing's research has been supported by NSFC No. 11171177.

References

1. Aissi, H., Bazgan, C., Vanderpooten, D.: Approximating min-max (regret) versions of some polynomial problems. In: Chen, D.Z., Lee, D.T. (eds.) COCOON 2006. LNCS, vol. 4112, pp. 428–438. Springer, Heidelberg (2006)
2. Bárány, I., Fiala, T.: Többgépes ütemezési problémák közel optimális megoldása (in Hungarian). Szigma - Matematikai - Közgazdasági Folyóirat **15**, 177–191 (1982)

3. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1**, 269–271 (1959)
4. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman, San Francisco (1979)
5. Gonzalez, T., Sahni, S.: Open shop scheduling to minimize finish time. Journal of the Association for Computing Machinery **23**, 665–679 (1976)
6. Jackson, J.R.: An extension of Johnson's results on job-lot scheduling. Naval Research Logistics Quarterly **3**, 201–203 (1956)
7. Jansen, K., Solis-Oba, R., Sviridenko, M.: Makespan minimization in job shops: A linear time approximation scheme. SIAM Journal on Discrete Mathematics **16**, 288–300 (2003)
8. Johnson, S.M.: Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly **1**, 61–68 (1954)
9. Lenstra, J.K., Kan, A.R., Brucker, P.: Complexity of machine scheduling problems. Annals of Operations Research **1**, 343–362 (1977)
10. Lenstra, J.K., Rinnooy Kan, A.: Computational complexity of discrete optimization. Annals of Operations Research **4**, 121–140 (1979)
11. Mastrolilli, M., Svensson, O.: Hardness of approximating flow and job shop scheduling problems. Journal of the Association for Computing Machinery **58**, 20:1–20:32 (2011)
12. Nip, K., Wang, Z.: Combination of two-machine flow shop scheduling and shortest path problems. In: Du, D.-Z., Zhang, G. (eds.) COCOON 2013. LNCS, vol. 7936, pp. 680–687. Springer, Heidelberg (2013)
13. Nip, K., Wang, Z., Talla Nobibon, F., Leus, R.: A Combination of Flow Shop Scheduling and the Shortest Path Problem. Journal of Combinatorial Optimization **29**, 36–52 (2015)
14. Schmidt, J.P., Siegel, A., Srinivasan, A.: Chernoff-hoeffding bounds for applications with limited independence. SIAM Journal on Discrete Mathematics **8**, 223–250 (1995)
15. Sevastianov, S.V., Woeginger, G.J.: Makespan minimization in open shops: A polynomial time approximation scheme. Mathematical Programming **82**, 191–198 (1998)
16. Shmoys, D.B., Stein, C., Wein, J.: Improved approximation algorithms for shop scheduling problems. SIAM Journal on Computing **23**, 617–632 (1994)
17. Wang, Z., Cui, Z.: Combination of parallel machine scheduling and vertex cover. Theoretical Computer Science **460**, 10–15 (2012)
18. Williamson, D.P., Hall, L.A., Hoogeveen, J.A., Hurkens, C.A.J., Lenstra, J.K., Sevast'janov, S.V., Shmoys, D.B.: Short shop schedules. Operations Research **45**, 288–294 (1997)