Combination of Two-Machine Flow Shop Scheduling and Shortest Path Problems

Kameng Nip and Zhenbo Wang*

Department of Mathematical Sciences, Tsinghua University, Beijing, 100084, China
zwang@math.tsinghua.edu.cn

Abstract. This paper studies a combinatorial optimization problem which is obtained by combining the two-machine flow shop scheduling problem and the shortest path problem. The objective of the obtained problem is to select a subset of jobs constitutes a feasible solution to the shortest path problem, and to execute the selected jobs on two-machine flow shop to minimize the makespan. We argue that this problem is NP-hard, and propose two approximation algorithms with constant factor guarantee.

Keywords: two-machine flow shop scheduling, shortest path, combination of optimization problems, approximation algorithm.

1 Introduction

With the rapid development of science and technology, manufacturing, service and management are often integrated, and decision-makers have to deal with systems involve several characteristics from more than one well-known combinatorial optimization problems. To the best of our knowledge, few research have been done about the combination of optimization problems in literature.

Wang and Cui [10] first studied a combination of the parallel machine scheduling problem and the vertex cover problem. The goal is to select a subset of jobs that forms a vertex cover of a given graph and to execute these jobs on m identical parallel machines. They proposed an $(3 - \frac{2}{m+1})$ - approximation algorithm for that problem. Wang et al. [11] have investigated a generalization of the above problem that combines the uniformly related parallel machine scheduling problem and a generalized covering problem. They proposed several approximation algorithms and mentioned as future research other combination of well-known combinatorial optimization problems. This is the core motivation for this work.

Let us consider the following scenario. We aim at building a railway between two specific cities. The railway needs to cross several adjacent cities, which is determined by a map (a graph). The processing time of manufacturing the rail track for each pair of cites is various. Manufacturing a rail track between two cities in the graph is associated with a job. The decision-maker needs to make two main decisions: (1) choosing a path to connect the two cities, and (2) deciding the

* Corresponding author.

D.-Z. Du and G. Zhang (Eds.): COCOON 2013, LNCS 7936, pp. 680–687, 2013.

schedule of manufacturing the rail tracks on this path in the factory. In addition, the manufacturing of rail tracks follows several working stages, each stage must start after the completion of the preceding stages, and we assume that there is only one machine for each stage. We wish to accomplish the manufacturing as early as possible, i.e. minimize the completion time of the schedule. It is a standard flow shop scheduling problem. In this paper, we assume that there are only two stages, leading to the two-machine flow shop scheduling problem. How can a decision maker choose a feasible path such that the corresponding jobs can be manufactured as early as possible? This problem combines the structure of the two-machine flow shop scheduling problem and the shortest path problem.

Following the framework introduced by Wang et al. [11], we can see our problem as a combination of two optimization problems, two-machine flow shop scheduling and some shortest path problem. The former problem can be solved in $O(n \log n)$ using Johnson's rule [6,8], and the classical shortest problem with non-negative edge weights can be solved in $O(|V|^2)$ using Dijkstra's algorithm [1,5].

The contributions of this paper include: (1) the argument that the considered problem is NP-hard, and (2) two constant factor approximation algorithms.

The rest of the paper is organized as follows. In section 2, we give a formal definition of our problem, and then briefly review the two-machine flow shop scheduling problem and some shortest path problems. In section 3, we study the computational complexity of our problem. Section 4 provides two approximation algorithms for this problem and we conclude in section 5.

2 Preliminaries

2.1 Problem Description

We first introduce the following generalized shortest path problem.

Definition 1. *Given a directed graph $G = (V, A, w^1, w^2)$ and two distinguished vertices $s, t \in V$ with $|A| = n$. Each arc $a_j \in A, j = 1, \cdots, n$ is associated with two weights w_j^1, w_j^2, and we define the vector $w^k = (w_1^k, w_2^k, \cdots, w_n^k)$ for $k = 1, 2$. The goal of our shortest path problem is to find a directed path P from s to t to minimize $f(w^1, w^2; x)$, in which f is certain specific objective function and $x \in \{0, 1\}^n$ is the decision variables such that $x_j = 1$ if and only if $a_j \in P$.*

We denote SP instead of $SP(G, s, t, f)$ to the problem described in definition 1. Notice that SP is a generalization of various shortest path problems. For instance, if we consider $w^2 = 0$ and $f = w^1 \cdot x$, where $\cdot$ is the dot product, this problem is the classical shortest path problem. If $f = \min\{w^1 \cdot x : w^2 \cdot x \leq K\}$, where K is a given number, this problem is the shortest weight-constrained path problem [7], and the decision version is known as ND30 in [7]. If $f = \max\{w^1 \cdot x, w^2 \cdot x\}$, the problem is the min-max shortest path problem [2,9,12] in literature.

We now give a formal definition of our problem, which is a combination of the two-machine flow shop problem and the shortest path problem.

Definition 2. *Given any instance I of the shortest path problem SP with $G = (V, A, w^1, w^2)$, and each arc $a_j \in A$ of I corresponds to a job $J_j \in J$ with processing times p_{1j}, p_{2j} on the two machines respectively. Define P_x be a set of jobs such that $J_j \in P_x$ if and only if $x_j = 1$. The $F2|$shortest path$|C_{max}$ problem is to find a feasible solution x of SP, that corresponds to a directed path connecting the vertices s and t in G, and assign the jobs of P_x on two-machine flow shop to minimize the makespan.*

2.2 Johnson's Rule for Two-Machine Flow Shop Scheduling

In flow shop scheduling, a schedule is called a permutation schedule if all jobs are processed in the same order on each machine [4]. Johnson [8] proposed a sequencing rule for $F2||C_{max}$, which is referred as Johnson's rule in literature.

Algorithm 1. Johnson's rule

1: Set $S_1 = \{J_j \in J | p_{1j} \leq p_{2j}\}$ and $S_2 = \{J_j \in J | p_{1j} > p_{2j}\}$.
2: Process the jobs in S_1 first with a non-decreasing order of p_{1j}, and then schedule the jobs in S_2 with a non-increasing order of p_{2j}, and ties may be broken arbitrarily.

In Johnson's rule, jobs are scheduled as early as possible. This rule produces a permutation schedule, and Johnson showed that it is an optimal schedule. Notice that this schedule is delivered in $O(n \log n)$ time.

We now introduce some well-known lower bounds for $F2||C_{max}$, that are used later to derive approximation algorithms to our problem. Let us denote by C_{max} the makespan in an arbitrary flow shop schedule with job set J, we have

$$C_{max} \geq \max \left\{ \sum_{J_j \in J} p_{1j}, \sum_{J_j \in J} p_{2j} \right\}, \tag{1}$$

and

$$C_{max} \leq \sum_{J_j \in J} (p_{1j} + p_{2j}). \tag{2}$$

For each job, we have

$$C_{max} \geq p_{1j} + p_{2j}, \qquad \forall J_j \in J. \tag{3}$$

Suppose J_v is the critical job of the flow shop, we have

$$C_{max} = \max_{J_u \in J} \left\{ \sum_{j=1}^{u} p_{1j} + \sum_{j=u}^{n} p_{2j} \right\} = \sum_{j=1}^{v} p_{1j} + \sum_{j=v}^{n} p_{2j}. \tag{4}$$

2.3 Algorithms for Shortest Path Problems

In this paper, we use two following results of the shortest path problem.

The first one is the well-known Dijkstra's algorithm, which solves the classical shortest path problem with nonnegative edge weights in $O(|V|^2)$ time [5].

The second one is an FPTAS result for min-max shortest path problem, which is presented by Aissi, Bazgan and Vanderpooten [2]. Their algorithm, denoted as the ABV algorithm, is based on dynamic programming and scaling technique, and we have the following result.

Theorem 1 ([2]). *Given an arbitrary positive value $\epsilon > 0$, in a given directed graph with two nonnegative weights associated with each arc, a direct path P between two specific vertices can be found by the ABV algorithm with the property*

$$\max\left\{\sum_{a_j\in P} w_j^1, \sum_{a_j\in P} w_j^2\right\} \le (1+\epsilon)\max\left\{\sum_{a_j\in P'} w_j^1, \sum_{a_j\in P'} w_j^2\right\}$$

for any other path P', and the running time is $O(|A||V|^3/\epsilon^2)$.

3 Computational Complexity of $F2|\text{shortest path}|C_{max}$

We argue that the decision version of our problem is NP-complete, by a reduction from a NP-complete problem PARTITION [7]. The proof is similar to the well-known NP-hardness proof of ND30 in [7], one could refer to the literature, such as the reduction presented in [3].

Theorem 2. *The decision problem of $F2|\text{shortest path}|C_{max}$ is NP-complete.*

Nevertheless, we emphasize that ND30 is neither a special case nor simple application of our problem. Since idles may occur on machine 2 in the flow shop scheduling, it is not straightforward that the path found in our problem is relevant to a shortest weight-constrained path found in ND30, which has total weight and total length bounded by two given numbers.

4 Approximation Algorithms

4.1 A Natural Approximation Algorithm

The main idea of the first algorithm is, we first set $w_j^1 = p_{1j} + p_{2j}$ and $w_j^2 = 0$ for each arc and find the shortest path with respect to w^1 by Dijkstra's algorithm. Then we schedule the corresponding jobs in the flow shop by Johnson's rule.

It is straightforward that the total running time of the JD algorithm is $O(|V|^2)$. Then we study the performance. First, we introduce some notations. Let J^* be set of jobs in an optimal solution, and C^*_{max} be the corresponding makespan. J_x and C_{max} are those returned by the JD algorithm.

Algorithm 2. The JD algorithm

1: Find the shortest path in G with weight $(w_j^1, w_j^2) := (p_{1j} + p_{2j}, 0)$ by Dijkstra's algorithm. For the returned solution x, construct the job set P_x.
2: Schedule the jobs of P_x by Johnson's rule. Let σ be the returned job schedule and C_{max} the returned makespan, and denote the job set P_x by J_x.
3: **return** J_x, σ **and** C_{max}

Theorem 3. *The JD algorithm is 2-approximate.*

Proof. By the lower bound (1) introduced in section 2.2, we have

$$2C_{max}^* \geq \sum_{J_j \in J^*} p_{1j} + \sum_{J_j \in J^*} p_{2j} = \sum_{J_j \in J^*} (p_{1j} + p_{2j}). \tag{5}$$

Since the returned path is shortest with respect to w^1 , we have

$$C_{max} \leq \sum_{J_j \in J_x} (p_{1j} + p_{2j}) = \sum_{J_j \in J_x} w_j^1 \leq \sum_{J_j \in J^*} w_j^1 = \sum_{J_j \in J^*} (p_{1j} + p_{2j}), \tag{6}$$

Combining with (5) and (6), it follows that $C_{max} \leq 2C_{max}^*$.

Consider the following instance. A directed graph G includes three vertices, which are referred to v_1, v_2, v_3. There are three jobs (arcs): (v_1, v_2), (v_2, v_3), (v_1, v_3), with processing times $(1, 0), (0, 1), (2 - \epsilon, 0)$ respectively in which ϵ is small enough. We wish to find a path from vertex v_1 to v_3. The makespan of job schedule returned by the JD algorithm is $C_{max} = 2 - \epsilon$ with the arc (v_1, v_3), whereas the makespan of optimal job schedule is $C_{max}^* = 1$ with the arcs $(v_1, v_2), (v_2, v_3)$. The bound is tight as $\frac{C_{max}}{C_{max}^*} \to 2$ when $\epsilon \to 0$. □

4.2 An Improved Approximation Algorithm

Instead of finding a shortest path from s to t optimally with respect to certain weight, we could adopt the FPTAS result mentioned in section 2.3, that will return a $(1 + \epsilon)$-approximated solution for the min-max shortest path problem. Then we also implement Johnson's rule. In other words, by setting the objective function $f = \max\{w^1 \cdot x, w^2 \cdot x\}$ in SP.

We initially set $(w_j^1, w_j^2) := (p_{1j}, p_{2j})$. The algorithm iteratively runs the above executions for the min-max shortest path problem and $F2||C_{max}$ by adopting the following revision policy: in a current schedule, if there exists some job is big enough with respect to the current makespan, we will revise the weights of arcs corresponding to big jobs to (M, M), where M is a sufficient large number, and then mark these jobs. The algorithm terminates if no such a job exists. Another terminating condition is that when a marked job appears in a current schedule. We return the schedule with minimum makespan among all current schedules as the solution of the algorithm. We denote this algorithm as the JAR algorithm.

Algorithm 3. The JAR algorithm

1: Initially, $(w_j^1, w_j^2) := (p_{1j}, p_{2j})$, for each arc $a_j \in A$ corresponding to $J_j \in J$.
2: Given $\epsilon > 0$, implement the ABV algorithm to obtain a feasible solution x of SP, and construct the corresponding job set as P_x.
3: Schedule the jobs of P_x by Johnson's rule, denote the returned makespan as C'_{max}, and the job schedule as σ'.
4: $J_x := P_x$, $\sigma := \sigma'$, $C_{max} := C'_{max}$, $D := \emptyset$, $M := (1+\epsilon)\sum_{J_j \in J}(|p_{1j}| + |p_{2j}|) + 1$.
5: **while** $P_x \cap D = \emptyset$ **and** there exists a job J_j in P_x such that $p_{1j} + p_{2j} > \frac{2}{3}C'_{max}$ **do**
6: **for** all jobs with $p_{1j} + p_{2j} > \frac{2}{3}C'_{max}$ in $J \backslash D$ **do**
7: $(w_j^1, w_j^2) := (M, M)$, $D := D \cup \{J_j\}$.
8: **end for**
9: Implement the ABV algorithm to obtain a feasible solution x of SP, and construct the corresponding job set as P_x.
10: Schedule the jobs of P_x by Johnson's rule, denote the returned makespan as C'_{max}, and the job schedule as σ'.
11: **if** $C'_{max} < C_{max}$ **then**
12: $J_x := P_x$, $\sigma := \sigma'$, $C_{max} := C'_{max}$.
13: **end if**
14: **end while**
15: **return** J_x, σ **and** C_{max}.

Now, we discuss the computational complexity of the JAR algorithm. Let the total number of jobs be $|A| = n$. First, we need to revise the weights of at most n arcs, hence lines 6 - 8 execute at most $O(n)$ times in the whole execution of our algorithm. And at least one job is added to D in each iteration, the iterations in lines 5 - 14 execute at most n times. In each iteration, the running time of obtaining a path by the ABV algorithm and a job schedule by Johnson's rule is $O(n|V|^3/\epsilon^2)$ and $O(n \log n)$ respectively. And $O(n)$ time is enough to other operations. Hence, the total running time of the JAR algorithm is $O(n^2(|V|^3/\epsilon^2 + \log n))$.

The following theorem shows the performance of the JAR algorithm.

Theorem 4. *Given $\epsilon > 0$, the JAR algorithm is $\frac{3}{2}(1+\epsilon)$-approximate.*

Proof. Case 1. $J^* \cap D \neq \emptyset$

It implies that there is at least one job in the optimal solution, say J_j, such that $(p_{1j} + p_{2j}) > \frac{2}{3}C'_{max}$ holds for a current schedule with makespan C'_{max} during the execution. Notice that the schedule returned by the JAR algorithm is the schedule with minimum makespan among all current schedules, and we have $C_{max} \leq C'_{max}$. It follows from (3) that

$$C_{max} \leq C'_{max} < \frac{3}{2}(p_{1j} + p_{2j}) \leq \frac{3}{2}C^*_{max}. \tag{7}$$

Case 2. $J^* \cap D = \emptyset$

Consider the last current schedule during the execution of the algorithm. We denote the corresponding job set and the makespan as J' and C'_{max} respectively.

In this case, we first argue that $J' \cap D = \emptyset$. Suppose not, since $J^* \cap D = \emptyset$, the weights of arcs corresponding to the jobs in J^* have not been revised. Hence we have $(1+\epsilon)\max\left\{\sum_{J_j \in J^*} w_j^1, \sum_{J_j \in J^*} w_j^2\right\} < M$. Moreover, by the assumption $J' \cap D \neq \emptyset$, we have $\max\left\{\sum_{J_j \in J'} w_j^1, \sum_{J_j \in J'} w_j^2\right\} \geq M$. By Theorem 1, the solution returned by the ABV algorithm satisfies

$$M \leq \max\left\{\sum_{J_j \in J'} w_j^1, \sum_{J_j \in J'} w_j^2\right\} \leq (1+\epsilon)\max\left\{\sum_{J_j \in J^*} w_j^1, \sum_{J_j \in J^*} w_j^2\right\} < M,$$

that leads to a contradiction.

Let J_v be a critical job in the last current schedule, and suppose that $p_{1v} \geq p_{2v}$. It follows from $p_{1j} \geq p_{2j}$ for $j = v+1, \cdots, n$ in the schedule returned by Johnson's rule and (4) that $C'_{max} \leq \sum_{J_j \in J'} p_{1j} + p_{2v}$. Since $J' \cap D = \emptyset$, we have $p_{1j} + p_{2j} \leq \frac{2}{3}C'_{max}$ for all jobs $J_j \in J'$, as otherwise the algorithm will continue. Thus, it follows from (1), (3), Theorem 1 and the fact that the schedule returned by the JAR algorithm is the schedule with minimum makespan among all current schedules, we have

$$\begin{aligned} C_{max} \leq C'_{max} &\leq \sum_{J_j \in J'} p_{1j} + p_{2v} \leq \sum_{J_j \in J'} w_j^1 + \frac{1}{2}(p_{1v} + p_{2v}) \\ &\leq (1+\epsilon)\max\left\{\sum_{J_j \in J^*} w_j^1, \sum_{J_j \in J^*} w_j^2\right\} + \frac{1}{3}C'_{max} \\ &= (1+\epsilon)\max\left\{\sum_{J_j \in J^*} p_{1j}, \sum_{J_j \in J^*} p_{2j}\right\} + \frac{1}{3}C'_{max} \\ &\leq (1+\epsilon)C^*_{max} + \frac{1}{3}C'_{max}. \end{aligned}$$

It suffices to show that $C_{max} \leq C'_{max} \leq \frac{3}{2}(1+\epsilon)C^*_{max}$.

For the case that the last current schedule with critical job $p_{1v} < p_{2v}$, an analogous argument will yield the same result. Therefore, the JAR algorithm is $\frac{3}{2}(1+\epsilon)$-approximate for $F2|\text{shortest path}|C_{max}$.

The following instance shows that the worst case ratio of the JAR algorithm can not less than $\frac{3}{2}$. A directed graph G has four vertices, which are referred to v_1, v_2, v_3, v_4. Given $\epsilon > 0$, there are four jobs (arcs): (v_1, v_2), (v_1, v_3), (v_3, v_2), and (v_2, v_4), with processing times $(1,1), ((1+4\epsilon), 0), (0, (1+4\epsilon))$ and $(1,1)$ respectively. We wish to find a path from v_1 to v_4. Notice that the ABV algorithm returns the path with the arcs (v_1, v_2) and (v_2, v_4), and the corresponding makespan C'_{max} by Johnson's rule is 3. All the corresponding jobs satisfy $p_{1j} + p_{2j} = 2 \leq \frac{2}{3}C'_{max}$, and thus the algorithm terminates. Therefore, the makespan of the returned job schedule by the JAR algorithm is $C_{max} = 3$. On the other hand, the optimal makespan is $C^*_{max} = 2 + 4\epsilon$, with the corresponding

arcs (v_1, v_3), (v_3, v_2), and (v_2, v_4). The worst case ratio of the JAR algorithm can not less than $\frac{3}{2}$ as $\frac{C_{max}}{C^*_{max}} \to 3/2$ when $\epsilon \to 0$ for this instance. □

5 Conclusions

This paper studies a combination problem of two-machine flow shop scheduling and shortest path problems. It is interesting to find an approximation algorithm with a better performance ratio for this problem. On the other hand, one can consider the combination problem of more generalized forms, such as combining with m-machine flow shop scheduling problem, or covering problem presented in [11]. All these questions motivate us to further investigate.

Acknowledgments. This work has been supported by Bilateral Scientific Cooperation Project between Tsinghua University and K.U. Leuven. We would like to thank Fabrice Talla Nobibon for helpful comments and suggestions.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, New Jersey (1993)
2. 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)
3. Batagelj, V., Brandenburg, F.J., Mendez, P., Sen, A.: The generalized shortest path problem. CiteSeer Archives (2000)
4. Chen, B., Potts, C.N., Woeginger, G.J.: A review of machine scheduling: Complexity, algorithms and approximability. In: Du, D.Z., Pardalos, P.M. (eds.) Handbook of Combinatorial Optimization, vol. 3, pp. 21–169. Kluwer (1998)
5. Dijkstra, E.W.: A note on two problems in connexion with graph. Numerische Mathematik 1, 269–271 (1959)
6. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1, 117–129 (1976)
7. Gary, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman, San Francisco (1979)
8. Johnson, S.M.: Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1, 61–68 (1954)
9. Kouvelis, P., Yu, G.: Robust discrete optimization and its applications. Kluwer Academic Publishers, Boston (1997)
10. Wang, Z., Cui, Z.: Combination of parallel machine scheduling and vertex cover. Theoretical Computer Science 460, 10–15 (2012)
11. Wang, Z., Hong, W., He, D.: Combination of parallel machine scheduling and covering problem. Working paper. Tsinghua University (2012)
12. Yu, G.: Min-max optimization of several classical discrete optimization problems. Journal of Optimization Theory and Applications 98, 221–242 (1998)