Approximation Algorithms for Two-Machine Flow-Shop Scheduling with a Conflict Graph

Yinhui Cai[1], Guangting Chen[2], Yong Chen[3](✉), Randy Goebel[4], Guohui Lin[4](✉), Longcheng Liu[4,5], and An Zhang[3]

[1] School of Sciences, Hangzhou Dianzi University, Hangzhou, China
[2] Taizhou University, Taizhou, Zhejiang, China
gtchen@hdu.edu.cn
[3] Department of Mathematics, Hangzhou Dianzi University, Hangzhou, China
{chenyong,anzhang}@hdu.edu.cn
[4] Department of Computing Science, University of Alberta, Edmonton, AB, Canada
{rgoebel,guohui}@ualberta.ca
[5] School of Mathematical Sciences, Xiamen University, Xiamen, China
longchengliu@xmu.edu.cn

Abstract. Path cover is a well-known intractable problem whose goal is to find a minimum number of vertex disjoint paths in a given graph to cover all the vertices. We show that a variant, where the objective function is not the number of paths but the number of length-0 paths (that is, isolated vertices), turns out to be polynomial-time solvable. We further show that another variant, where the objective function is the total number of length-0 and length-1 paths, is also polynomial-time solvable. Both variants find applications in approximating the two-machine flow-shop scheduling problem in which job processing constraints are formulated as a conflict graph. For the unit jobs, we present a 4/3-approximation algorithm for the scheduling problem with an arbitrary conflict graph, based on the exact algorithm for the variants of the path cover problem. For arbitrary jobs where the conflict graph is the union of two disjoint cliques (i.e., all the jobs can be partitioned into two groups such that the jobs in a group are pairwise conflicting), we present a simple 3/2-approximation algorithm.

Keywords: Flow-shop scheduling · Conflict graph · b-matching
Path cover · Approximation algorithm

1 Introduction

Scheduling is a well established research area that finds numerous applications in modern manufacturing industries and in operations research at large. All

Y. Cai and L. Liu—Co-first authors.

scheduling problems modeling real-life applications have at least two components: the machines and the jobs. One intensively studied problem focuses on scheduling constraints are imposed between a machine and a job, such as a time interval during which the job is allowed to be processed *nonpreemptively* on the machine, while the machines are considered as independent from each other, so are the jobs. For example, the parallel machine scheduling (the *multiprocessor scheduling* in [15]) is one of the first studied problems, denoted as $Pm \mid\mid C_{\max}$ in the three-field notation [20], in which a set of jobs each needs to be processed by one of the m given identical machines, with the goal to minimize the maximum job completion time (called the *makespan*); the m-machine flow-shop scheduling (the *flow-shop scheduling* in [15]) is another early-studied problem, denoted as $Fm \mid\mid C_{\max}$, in which a set of jobs each needs to be processed by all the m given machines in the same sequential order, with the goal to minimize the makespan.

In another category of scheduling problems, additional but limited resources are required for the machines to process the jobs [13]. The resources are renewable but normally non-sharable in practice; the jobs competing for the same resource have to be processed at different time if their total demand for a certain resource exceeds the supply. Scheduling with resource constraints [13,14] or scheduling with conflicts (SwC) [11] also finds numerous applications [3,9,22] and has attracted as much attention as the non-constrained counterpart. In this paper, we use SwC to refer to nonpreemptive scheduling problems with additional constraints or conflicting relationships among the jobs to disallow them to be processed concurrently on different machines. We note that SwC is also presented as the scheduling with agreements (SwA), in which a subset of jobs can be processed concurrently on different machines if and only if they agree with each other [4,5]. While in the most general scenario a conflict could involve multiple jobs, in this paper we consider only those conflicts each involves two jobs and consequently all the conflicts under consideration can be presented as a *conflict graph* $G = (V, E)$, where V is the set of jobs and an edge $e = (J_{j_1}, J_{j_2}) \in E$ represents a conflicting pair such that the two jobs J_{j_1} and J_{j_2} cannot be processed concurrently on different machines in any feasible schedule.

Extending the three-field notation [20], the parallel machine SwC with a conflict graph $G = (V, E)$ (also abbreviated as SCI in the literature) [11] is denoted as $Pm \mid G = (V, E), p_j \mid C_{\max}$, where the first field Pm tells that there are m parallel identical machines, the second field describes the conflict graph $G = (V, E)$ over the set V of all the jobs, where the job J_j requires a non-preemptive processing time of p_j on any machine, and the last field specifies the objective function to minimize the makespan $C_{\max}$. One clearly sees when $E = \emptyset$, $Pm \mid G = (V, E), p_j \mid C_{\max}$ reduces to the classical multiprocessor scheduling $Pm \mid\mid C_{\max}$, which is already NP-hard for $m \geq 2$ [15]. Indeed, with m either a given constant or part of input, $Pm \mid G = (V, E), p_j \mid C_{\max}$ is more difficult to approximate than the classical multiprocessor scheduling, as $P3 \mid G = (V, E), p_j = 1 \mid C_{\max}$ and $P2 \mid G = (V, E), p_j \in \{1, 2, 3, 4\} \mid C_{\max}$ are APX-hard [11,26]. There is a rich line of research to consider the unit jobs (that is, $p_j = 1$) and/or to consider certain special classes of conflict graphs. The interested reader might see [11] and the references therein.

In the general m-machine (also called m-stage) flow-shop [15] denoted as $Fm \mid\mid C_{\max}$, there are $m \geq 2$ machines $M_1, M_2, \ldots, M_m$, a set V of jobs each job J_j needs to be processed through $M_1, M_2, \ldots, M_m$ sequentially with processing times $p_{1j}, p_{2j}, \ldots, p_{mj}$ respectively. When $m = 2$, the two-machine flow-shop problem is polynomial time solvable, by Johnson's algorithm [23]; the m-machine flow-shop problem, when $m \geq 3$, is *strongly* NP-hard [16]. After several efforts [10,16,19,23], Hall presented a polynomial-time approximation scheme (PTAS) for the m-machine flow-shop problem, for any fixed integer $m \geq 3$ [21]. When m is part of input (*i.e.*, an arbitrary integer), there is no known constant ratio approximation algorithm, and the problem cannot be approximated within 1.25, unless P = NP [33].

The m-machine flow-shop SwC was first studied in 1980's. Blazewicz et al. [8] considered multiple resource characteristics including the number of resource types, resource availabilities and resource requirements; they expanded the middle field of the three-field notation to express these resource characteristics, for which the conflict relationships are modeled by complex structures such as hypergraphs. At the end, they proved complexity results for several variants in which either the conflict relationships are simple enough or only the unit jobs are considered. Further studies on more variants can be found in [6,7,28–30]. In this paper, we consider those conflicts each involves only two jobs such that all the conflicts under consideration can be presented as a conflict graph $G = (V, E)$. The m-machine flow-shop scheduling with a conflict graph $G = (V, E)$ is denoted as $Fm \mid G = (V, E), p_{ij} \mid C_{\max}$. We remark that our notation is slightly different from the one introduced by Blazewicz et al. [8], which uses a prefix "res" in the middle field for describing the resource characteristics.

Several applications of the m-machine flow-shop scheduling with a conflict graph have been mentioned in the literature. In a typical example of scheduling medical tests in an outpatient health care facility where each patient (the job) needs to do a sequence of m tests (the machines), a patient must be accompanied by their doctor during a test, so two patients under the care of the same doctor cannot go for tests simultaneously. That is, two patients of the same doctor are conflicting to each other, and all the conflicts can be effectively described as a graph $G = (V, E)$, where V is the set of all the patients and an edge represents a conflicting pair of patients.

In two recent papers [31,32], Tellache and Boudhar studied the problem $F2 \mid G = (V, E), p_{ij} \mid C_{\max}$, which they denote as FSC. In [32], the authors summarized and proved several complexity results; to name a few, $F2 \mid G = (V, E), p_{ij} \mid C_{\max}$ is strongly NP-hard when $G = (V, E)$ is the complement of a complete split graph [8,32] (that is, G is the union of a clique and an independent set), $F2 \mid G = (V, E), p_{ij} \mid C_{\max}$ is weakly NP-hard when $G = (V, E)$ is the complement of a complete bipartite graph [32] (that is, G is the union of two disjoint cliques), and for an arbitrary conflict graph $G = (V, E)$, $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$ is strongly NP-hard [32]. In [31], the authors proposed three mixed-integer linear programming models and a branch and bound algorithm to solve the last variant $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$ exactly; their empirical study shows that the branch and bound algorithm outperforms and can solve instances of up to $20,000$ jobs.

In this paper, we pursue approximation algorithms with provable performance for the NP-hard variants of the two-machine flow-shop scheduling with a conflict graph. In Sect. 2, we present a 4/3-approximation algorithm for the strongly NP-hard problem $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$ for the unit jobs with an arbitrary conflict graph. In Sect. 3, we present a simple 3/2-approximation algorithm for the weakly NP-hard problem $F2 \mid G = K_\ell \cup K_{n-\ell}, p_{ij} \mid C_{\max}$ for arbitrary jobs with a conflict graph that is the union of two disjoint cliques. Some concluding remarks are provided in Sect. 4.

2 Approximating $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$

Tellache and Boudhar proved that $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$ is strongly NP-hard by a reduction from the well known Hamiltonian path problem, which is strongly NP-complete [15]. Furthermore, they remarked that $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$ has a feasible schedule of makespan $C_{\max} = n + k$ if and only if the complement $\overline{G}$ of the conflict graph G, called the agreement graph, has a path cover of size k (that is, a collection of k vertex-disjoint paths that covers all the vertices of the graph $\overline{G}$), where n is the number of jobs (or vertices) in the instance. This way, $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$ is polynomially equivalent to the Path Cover problem, which is NP-hard even on some special classes of graphs including planar graphs [17], bipartite graphs [18], chordal graphs [18], chordal bipartite graphs [24] and strongly chordal graphs [24]. In terms of approximability, to the best of our knowledge there is no $o(n)$-approximation algorithm for the Path Cover problem.

We begin with some terminologies. The conflict graphs considered in this paper are all simple graphs. All paths and cycles in a graph are also simple. The number of edges on a path/cycle defines the length of the path/cycle. A length-k path/cycle is also called a k-path/cycle for short. Note that a single vertex is regarded as a 0-path, while a cycle has length at least 3. For an integer $b \geq 1$, a b-matching of a graph is a spanning subgraph in which every vertex has degree no greater than b; a maximum b-matching is a b-matching that contains the maximum number of edges. A maximum b-matching of a graph can be computed in $O(m^2 \log n \log b)$-time, where n and m are the number of vertices and the number of edges in the graph, respectively [12]. Clearly, a graph could have multiple distinct maximum b-matchings.

Given a graph, a path cover is a collection of vertex-disjoint paths in the graph that covers all the vertices, and the size of the path cover is the number of paths therein. The Path Cover problem is to find a path cover of a given graph of the minimum size, and the well known Hamiltonian path problem is to decide whether a given graph has a path cover of size 1. Many variants of the Path Cover problem have been studied in the literature [1,2,25,27]. We mentioned earlier that Tellache and Boudhar proved that $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$ is polynomially equivalent to the Path Cover problem, but to the best of our knowledge there is no approximation algorithm designed for $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$. Nevertheless, one easily sees that, since $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$ has a feasible schedule of makespan $C_{\max} = n + k$ if and only if the complement $\overline{G}$ of the conflict graph G has a path cover of size k, a trivial algorithm simply processing the jobs one by one (each on the first machine M_1 and then on the second machine M_2) produces a schedule of makespan $C_{\max} = 2n$, and thus is a 2-approximation algorithm.

In this section, we will design two approximation algorithms with improved performance ratios for $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$. These two approximation algorithms are based on our polynomial time exact algorithms for two variants of the Path Cover problem, respectively. We start with the first variant called the *Path Cover with the minimum number of 0-paths*, in which we are given a graph and we aim to find a path cover that contains the minimum number of 0-paths. In the second variant called the *Path Cover with the minimum number of* $\{0, 1\}$*-paths*, we aim to find a path cover that contains the minimum total number of 0-paths and 1-paths. We remark that in both variants, we do not care about the size of the path cover.

2.1 Path Cover with the Minimum Number of 0-Paths

Recall that in this variant of the Path Cover problem, given a graph, we aim to find a path cover that contains the minimum number of 0-paths. The given graph is the complement $\overline{G} = (V, \overline{E})$ of the conflict graph $G = (V, E)$ in $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$. We next present a polynomial time algorithm that finds for $\overline{G}$ a path cover that contains the minimum number of 0-paths.

In the first step, we apply any polynomial time algorithm to find a maximum 2-matching in $\overline{G}$, denoted as M; recall that this can be done in $O(m^2 \log n)$-time, where $n = |V|$ and $m = |\overline{E}|$. M is a collection of vertex-disjoint paths and cycles; let $\mathcal{P}_0$ ($\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_{\geq 3}$, $\mathcal{C}$, respectively) denote the sub-collection of 0-paths (1-paths, 2-paths, paths of length at least 3, cycles, respectively) in M. That is, $M = \mathcal{P}_0 \cup \mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_{\geq 3} \cup \mathcal{C}$.

Clearly, if $\mathcal{P}_0 = \emptyset$, then we have a path cover containing no 0-paths after removing one edge per cycle in $\mathcal{C}$. In the following discussion we assume the existence of a 0-path, which is often called a *singleton*. We also call an ending vertex of a k-path with $k \geq 1$ as an endpoint for simplicity. The following lemma is trivial due to the edge maximality of M.

Lemma 1. *All the singletons and endpoints in the maximum* 2*-matching* M *are pairwise non-adjacent to each other in the underlying graph* $\overline{G}$.

Let v_0 be a singleton. If v_0 is adjacent to a vertex v_1 on a cycle of $\mathcal{C}$ in the underlying graph $\overline{G}$, then we may delete a cycle-edge incident at v_1 from M while add the edge (v_0, v_1) to M to achieve another maximum 2-matching with one less singleton. Similarly, if v_0 is adjacent to a vertex v_1 on a path of $\mathcal{P}_{\geq 3}$ (note that v_1 has to be an internal vertex on the path by Lemma 1) in the underlying graph $\overline{G}$, then we may delete a certain path-edge incident at v_1 from M while add the edge (v_0, v_1) to M to achieve another maximum 2-matching with one less singleton. In either of the above two cases, assume the edge deleted from M is (v_1, v_2); then we say the *alternating* path v_0-v_1-v_2 *saves* the singleton v_0.

In the general setting, in the underlying graph $\overline{G}$, v_0 is adjacent to the middle vertex v_1 of a 2-path P_1, one endpoint v_2 of P_1 is adjacent to the middle vertex v_3 of another 2-path P_2, one endpoint v_4 of P_2 is adjacent to the middle vertex v_5 of another 2-path P_3, and so on, one endpoint v_{2i-2} of P_{i-1} is adjacent to the middle vertex v_{2i-1} of another 2-path P_i, one endpoint v_{2i} of P_i is adjacent to a vertex v_{2i+1} of a cycle of $\mathcal{C}$ or a path of $\mathcal{P}_{\geq 3}$ (see an illustration in Fig. 1), on which the edge (v_{2i+1}, v_{2i+2}) is to be deleted. Then we may delete the edges $\{(v_{2j+1}, v_{2j+2}) \mid j = 0, 1, \ldots, i\}$ from M while add the edges $\{(v_{2j}, v_{2j+1}) \mid j = 0, 1, \ldots, i\}$ to M to achieve another maximum 2-matching with one less singleton; and we say the *alternating* path v_0-v_1-v_2-…-v_{2i}-v_{2i+1}-v_{2i+2} *saves* the singleton v_0.

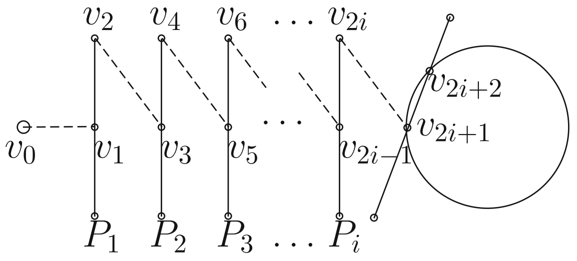

Fig. 1. An alternating path v_0-v_1-v_2-…-v_{2i}-v_{2i+1}-v_{2i+2} that saves the singleton v_0, where the last two vertices are on a path of $\mathcal{P}_{\geq 3}$ or a cycle. In the figure, solid edges are in and dashed edges are outside of the maximum 2-matching M.

```
ALGORITHM A($\overline{G} = (V, \overline{E})$):

Step 1. Compute a maximum 2-matching $M$;
Step 2. repeatedly find an alternating path to save a singleton in $M$,
    till either no singleton exists or no alternating path is found;
Step 3. break cycles in $M$ by removing one edge per cycle, and
    return the resulting path cover.
```

Fig. 2. A high-level description of ALGORITHM A for computing a path cover in the agreement graph $\overline{G} = (V, \overline{E})$.

The second step of the algorithm is to iteratively find a simple alternating path to save a singleton; it terminates when no alternating path is found. The resulting maximum 2-matching is still denoted as M.

In the last step, we break the cycles in M by deleting one edge per cycle to produce a path cover. Denote our algorithm as Algorithm A, of which a high-level description is provided in Fig. 2. We will prove in the next theorem that the path cover produced by Algorithm A contains the minimum number of 0-paths.

Theorem 1. Algorithm A *is an* $O(m^2 \log n)$*-time algorithm for computing a path cover with the minimum number of* 0*-paths in the agreement graph* $\overline{G}$*.*

2.2 Path Cover with the Minimum Number of $\{0, 1\}$-Paths

In this variant of the Path Cover problem, given a graph, we aim to find a path cover that contains the minimum total number of 0-paths and 1-paths. Again, the given graph is the complement $\overline{G} = (V, \overline{E})$ of the conflict graph $G = (V, E)$ in $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$. We next present a polynomial time algorithm called Algorithm B that finds for $\overline{G}$ such a path cover.

Recall that in Algorithm A for computing a path cover that contains the minimum number of 0-paths, an alternating path saving a singleton v_0 starts from the singleton v_0 and reaches a vertex v_{2i+1} on a path of $\mathcal{P}_{\geq 3}$ or on a cycle of $\mathcal{C}$ (see Fig. 1). If v_{2i+1} is on a cycle, then the last vertex v_{2i+2} can be any one of the two neighbors of v_{2i+1} on the cycle. If v_{2i+1} is on a k-path, then the last vertex v_{2i+2} is a *non-endpoint* neighbor of v_{2i+1} on the path (the existence is guaranteed by $k \geq 3$); and the reason why v_{2i+2} cannot be an endpoint is obvious since otherwise v_{2i+2} would be left as a new singleton after the edge swapping. In the current variant we want to minimize the total number of 0-paths and 1-paths; clearly v_{2i+2} cannot be an endpoint either and cannot even be the vertex adjacent to an endpoint, for the latter case because the edge swapping saves v_0 but leaves a new 1-path. To guarantee the existence of such vertex v_{2i+2}, the k-path must have $k \geq 4$, and if $k = 4$ then v_{2i+1} cannot be the middle vertex of the 4-path.

Algorithm B is in spirit similar to but in practice slightly more complex than Algorithm A, mostly because the definition of an alternating path saving a singleton or a 1-path is different, and slightly more complex.

In the first step of Algorithm B, we apply any polynomial time algorithm to find a maximum 2-matching M in $\overline{G}$. Let $\mathcal{P}_0$ ($\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, $\mathcal{P}_4$, $\mathcal{P}_{\geq 5}$, $\mathcal{C}$, respectively) denote the sub-collection of 0-paths (1-paths, 2-paths, 3-paths, 4-paths, paths of length at least 5, cycles, respectively) in M. We also let $\mathcal{P}_{0,1} = \mathcal{P}_0 \cup \mathcal{P}_1$ denote the collection of all 0-paths (called *singletons*) and 1-paths in M.

Let $e_0 = (v_0, u_0)$ be an edge in M. In the sequel when we say e_0 is *adjacent* to a vertex v_1 in the graph $\overline{G}$, we mean v_1 is a different vertex (from v_0 and u_0) and at least one of v_0 and u_0 is adjacent to v_1; if both v_0 and u_0 are adjacent to v_1, then pick one (often arbitrarily) for the subsequent purposes. This way, we unify our treatment on singletons and 1-paths, for the reasons to be seen in the following. For ease of presentation, we use an *object* to refer to a vertex or an edge. Like in the last subsection, an ending vertex of a k-path with $k \geq 1$ or an ending edge of a k-path with $k \geq 2$ is called an *end-object* for simplicity.

Let v_0 be a singleton or $e_0 = (v_0, u_0)$ be a 1-path in M. In the underlying graph $\overline{G}$, if v_0 is adjacent to a vertex v_1 on a cycle of $\mathcal{C}$, or on a path of $\mathcal{P}_{\geq 5}$, or on a 4-path such that v_1 is not the middle vertex, then we may delete a certain edge incident at v_1 from M while add the edge (v_0, v_1) to M to achieve another maximum 2-matching with one less singleton if v_0 is a singleton or with one less 1-path. In either of the three cases, assume the edge deleted from M is (v_1, v_2); then we say the *alternating* path v_0-v_1-v_2 *saves* the singleton v_0 or the 1-path $e_0 = (v_0, u_0)$.

Analogously as in the last subsection, in the general setting, in the underlying graph $\overline{G}$, v_0 is adjacent to a vertex v_1 of a path $P_1 \in \mathcal{P}_{2,3,4}$ (if P_1 is a 4-path then v_1 has to be the middle vertex). Note that this vertex v_1 basically separates the two end-objects of the path P_1 — an analogue to the role of the middle vertex of a 2-path that separates the two endpoints of the 2-path. We say "an end-object of P_1 is adjacent to v_1 via v_2", to mean that if the end-object is a vertex then it is v_2, or if the end-object is an edge, then it is (v_2, u_2), with the edge (v_1, v_2) on the path P_1 either way (see an illustration in Fig. 3).

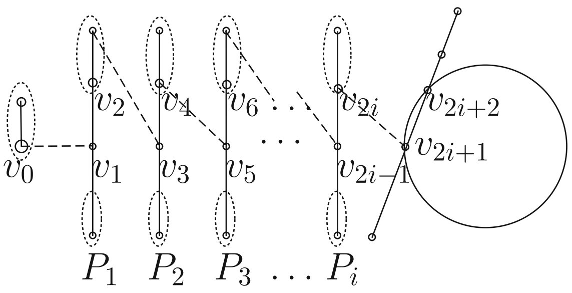

Fig. 3. An alternating path v_0-v_1-v_2-...-v_{2i}-v_{2i+1}-v_{2i+2} that saves the singleton v_0 or the 1-path $e_0 = (v_0, u_0)$, where the last two vertices are on a cycle of $\mathcal{C}$, or on a path of $\mathcal{P}_{\geq 5}$, or on a 4-path such that v_{2i+1} is not the middle vertex. In the figure, solid edges are in the maximum 2-matching M, dashed edges are outside of M, and a dotted circle contains an object which is either a vertex or an edge.

Suppose one end-object of P_1, which is adjacent to v_1 via v_2, is adjacent to a vertex v_3 of another $P_2 \in \mathcal{P}_{2,3,4}$ (the same, if P_2 is a 4-path then v_3 has to be the middle vertex); one end-object of P_2, which is adjacent to v_3 via v_4, is adjacent to a vertex v_5 of another $P_3 \in \mathcal{P}_{2,3,4}$ (the same, if P_3 is a 4-path then v_5 has to be the middle vertex); and so on; one end-object of P_{i-1}, which is adjacent to v_{2i-3} via v_{2i-2}, is adjacent to a vertex v_{2i-1} of another $P_i \in \mathcal{P}_{2,3,4}$ (the same, if P_i is a 4-path then v_{2i-1} has to be the middle vertex); one end-object of P_i, which is adjacent to v_{2i-1} via v_{2i}, is adjacent to a vertex v_{2i+1} of a cycle of $\mathcal{C}$, or of a path of $\mathcal{P}_{\geq 5}$, or of a 4-path such that v_{2i+1} is not the middle vertex (see an illustration in Fig. 3), on which a certain edge (v_{2i+1}, v_{2i+2}) is to be deleted. Then we may delete the edges $\{(v_{2j+1}, v_{2j+2}) \mid j = 0, 1, \dots, i\}$ from M while add the edges $\{(v_{2j}, v_{2j+1}) \mid j = 0, 1, \dots, i\}$ to M to achieve another maximum

2-matching with one less singleton if v_0 is a singleton or with one less 1-path. We say the *alternating* path v_0-v_1-v_2-…-v_{2i}-v_{2i+1}-v_{2i+2} *saves* the singleton v_0 or the 1-path $e_0 = (v_0, u_0)$. It is important to note that in this alternating path, the vertex v_2 "*represents*" the end-object of P_1, meaning that when the end-object is an edge, it is treated very the same as the vertex v_2.

The second step of the algorithm is to iteratively find a simple alternating path to save an object of $\mathcal{P}_{0,1}$; it terminates when no alternating path is found. The resulting maximum 2-matching is still denoted as M.

In the last step, we break the cycles in M by deleting one edge per cycle to produce a path cover. A high-level description of ALGORITHM B is similar to the one for ALGORITHM A shown in Fig. 2, replacing a singleton by an object of $\mathcal{P}_{0,1}$. We will prove in Theorem 2 that the path cover produced by ALGORITHM B contains the minimum total number of 0-paths and 1-paths.

Theorem 2. ALGORITHM B *is an $O(m^2 \log n)$-time algorithm for computing a path cover with the minimum total number of 0-paths and 1-paths in the agreement graph $\overline{G} = (V, \overline{E})$.*

Remark 1. The path cover produced by ALGORITHM B has the minimum total number of 0-paths and 1-paths in the agreement graph $\overline{G} = (V, \overline{E})$. One may run ALGORITHM A at the end of the second step of ALGORITHM B to achieve a path cover with the minimum total number of 0-paths and 1-paths, and with the minimum number of 0-paths. During the execution of ALGORITHM A, a singleton trades for a 1-path.

2.3 Approximation Algorithms for $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$

Given an instance of the problem $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$, where there are n unit jobs $V = \{J_1, J_2, \dots, J_n\}$ to be processed on the two-machine flow-shop, with their conflict graph $G = (V, E)$, we want to find a schedule with a provable makespan.

For a k-path in the agreement graph $\overline{G} = (V, \overline{E})$, where $k \geq 0$, for example $P = J_1$-J_2-…-J_k-J_{k+1}, we compose a sub-schedule π_P in which the machine M_1 continuously processes the jobs $J_1, J_2, \dots, J_{k+1}$ in order, and the machine M_2 in one unit of time after M_1 continuously processes these jobs in the same order. The sub-makespan for the flow-shop to complete these $k+1$ jobs is thus $k+2$ (units of time). Let $M = \{P_1, P_2, \dots, P_\ell\}$ be a path cover of size ℓ in the agreement graph $\overline{G}$. For each path P_i we use $|P_i|$ to denote its length and construct the sub-schedule π_{P_i} as above that has a sub-makespan of $|P_i| + 2$. We then concatenating these ℓ sub-schedules (in an arbitrary order) into a full schedule π, which clearly has a makespan $C^{\pi}_{\max} = \sum_{i=1}^{\ell}(|P_i| + 2) = n + \ell$.

On the other hand, given a schedule π, if two jobs J_{j_1} and J_{j_2} are processed concurrently on the two machines, then they have to be agreeing to each other and thus adjacent in the agreement graph $\overline{G}$; we select this edge (J_{j_1}, J_{j_2}). Note that one job can be processed concurrently with at most two other jobs as there are only two machines. Therefore, all the selected edges form into a number of vertex-disjoint paths in $\overline{G}$ (due to the flow-shop, no cycle is formed); these paths together with the vertices outside of the paths, which are the 0-paths, form a path cover for $\overline{G}$. Assuming without loss of generality that two machines cannot both idle at any time point, the makespan of the schedule is exactly. the sum of the number of jobs and the number of paths.

We state this relationship between a feasible schedule and a path cover in the agreement graph $\overline{G}$ into the following lemma.

Lemma 2 [32]. *A feasible schedule π for the problem $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$ one-to-one corresponds to a path cover M in the agreement graph $\overline{G}$, and $C^{\pi}_{\max} = n + |M|$, where n is the number of jobs in the instance.*

Theorem 3. *The problem $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$ admits an $O(m^2 \log n)$-time 4/3-approximation algorithm, where $n = |V|$ and $m = |\overline{E}|$.*

Remark 2. If Algorithm A is used in the proof of Theorem 3 to compute a path cover with the minimum number of 0-paths and subsequently to construct a schedule π, then we have $C^{\pi}_{\max} \le \frac{3}{2} C^{*}_{\max}$. That is, we have an $O(m^2 \log n)$-time 3/2-approximation algorithm based on Algorithm A.

When the agreement graph $\overline{G}$ consists of k vertex-disjoint triangles such that a vertex of the i-th triangle is adjacent to a vertex of the $(i+1)$-st triangle, for $i = 1, 2, \ldots, k-1$, and the maximum degree is 3, Algorithm B could produce a path cover containing k 2-paths, while there is a Hamiltonian path in the graph. This suggests that the approximation ratio 4/3 is asymptotically tight.

3 Approximating $F2 \mid G = K_\ell \cup K_{n-\ell}, p_{ij} \mid C_{\max}$

In this section, we present a 3/2-approximation algorithm for the weakly NP-hard problem $F2 \mid G = K_\ell \cup K_{n-\ell}, p_{ij} \mid C_{\max}$ for arbitrary jobs with a conflict graph that is the union of two disjoint cliques. Note that the agreement graph $\overline{G} = K_{\ell, n-\ell}$ is a complete bipartite graph. Without loss of generality, let the job set of K_ℓ be $A = \{J_1, J_2, \ldots, J_\ell\}$ and the job set of $K_{n-\ell}$ be $B = \{J_{\ell+1}, J_{\ell+2}, \ldots, J_n\}$.

For the job set A, we merge all its jobs (in the sequential order with increasing indices) to become a single "*aggregated*" job denoted as J_A, with its processing time on the machine M_1 being $P^1_A = \sum_{j=1}^{\ell} p_{1j}$ and its processing time on the machine M_2 being $P^2_A = \sum_{j=1}^{\ell} p_{2j}$. Likewise, for the job set B, we merge all its jobs (in the sequential order with increasing indices) to become a single aggregated job denoted as J_B, with its two processing times being $P^1_B = \sum_{j=\ell+1}^{n} p_{1j}$ and $P^2_B = \sum_{j=\ell+1}^{n} p_{2j}$. We now have an instance of the classical two-machine flow-shop scheduling problem consisting of only two aggregated jobs J_A and J_B, and we may apply Johnson's algorithm [23] to obtain a schedule denoted as π. From π we obtain a schedule for the original instance of the problem $F2 \mid G = K_\ell \cup K_{n-\ell}, p_{ij} \mid C_{\max}$, which is also denoted as π as there is no major difference. We call this algorithm as Algorithm C.

Theorem 4. Algorithm C *is an $O(m)$-time 3/2-approximation algorithm for the problem $F2 \mid G = K_\ell \cup K_{n-\ell}, p_{ij} \mid C_{\max}$, where m is the number of edges in the conflict graph G.*

In the schedule produced by Algorithm C, one sees that when the jobs of A are processed on the machine M_1, the other machine M_2 is left idle. This is certainly disadvantageous. For instance, when the jobs are all unit jobs and $|A| = |B| = \frac{1}{2}n$, the makespan of the produced schedule is $\frac{3}{2}n$, while the agreement graph is Hamiltonian and thus the optimal makespan is only $n + 1$. This huge gap suggests that one could probably design a better approximation algorithm and we leave it as an open question.

4 Concluding Remarks

In this paper, we investigated approximation algorithms for the two-machine flow-shop scheduling problem with a conflict graph. In particular, we considered two special cases of all unit jobs and of a conflict graph that is the union of two disjoint cliques, that is, $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$ and $F2 \mid G = K_\ell \cup K_{n-\ell}, p_{ij} \mid C_{\max}$. For the first problem we studied the graph theoretical problem of finding a path cover with the minimum total number of 0-paths and 1-paths, and presented a polynomial time exact algorithm. This exact algorithm leads to a 4/3-approximation algorithm for the problem $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$. We also showed that the performance ratio 4/3 is asymptotically tight. For the second problem $F2 \mid G = K_\ell \cup K_{n-\ell}, p_{ij} \mid C_{\max}$, we presented a 3/2-approximation algorithm.

Designing approximation algorithms for $F2 \mid G = (V, E), p_{ij} = 1 \mid C_{\max}$ with a performance ratio better than 4/3 is challenging, since one way or the other, one has to deal with longer paths in a path cover or has to deal with the original Path Cover problem. Nevertheless, better approximation algorithms for $F2 \mid G = K_\ell \cup K_{n-\ell}, p_{ij} \mid C_{\max}$ can be expected.

Acknowledgements. This research is partially supported by the NSFC Grants 11571252, 11771114 and 61672323, the China Scholarship Council Grants 201508330054 and 201706315073, NSERC Canada and the Fundamental Research Funds for the Central Universities (Grant No. 20720160035).

References

1. Asdre, K., Nikolopoulos, S.D.: A linear-time algorithm for the k-fixed-endpoint path cover problem on cographs. Networks **50**, 231–240 (2007)
2. Asdre, K., Nikolopoulos, S.D.: A polynomial solution to the k-fixed-endpoint path cover problem on proper interval graphs. Theoret. Comput. Sci. **411**, 967–975 (2010)
3. Baker, B.S., Coffman, E.G.: Mutual exclusion scheduling. Theoret. Comput. Sci. **162**, 225–243 (1996)

4. Bendraouche, M., Boudhar, M.: Scheduling jobs on identical machines with agreement graph. Comput. Oper. Res. **39**, 382–390 (2012)
5. Bendraouche, M., Boudhar, M.: Scheduling with agreements: new results. Int. J. Prod. Res. **54**, 3508–3522 (2016)
6. Błażewicz, J., et al.: Scheduling Under Resource Constraints - Deterministic Models (1986)
7. Błażewicz, J., et al.: Scheduling unit - time tasks on flow - shops under resource constraints. Ann. Oper. Res. **16**, 255–266 (1988)
8. Blazewicz, J., et al.: Scheduling subject to resource constraints: classification and complexity. Discrete Appl. Math. **5**, 11–24 (1983)
9. Bodlaender, H.L., Jansen, K.: Restrictions of graph partition problems. Part I. Theoret. Comput. Sci. **148**, 93–109 (1995)
10. Chen, B., et al.: A new heuristic for three-machine flow shop scheduling. Oper. Res. **44**, 891–898 (1996)
11. Even, G., et al.: Scheduling with conflicts: online and offline algorithms. J. Sched. **12**, 199–224 (2009)
12. Gabow, H.N.: An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In: STOC 1983, pp. 448–456 (1983)
13. Garey, M.R., Graham, R.L.: Bounds for multiprocessor scheduling with resource constraints. SIAM J. Comput. **4**, 187–200 (1975)
14. Garey, M.R., Johnson, D.S.: Complexity results for multiprocessor scheduling under resource constraints. SIAM J. Comput. **4**, 397–411 (1975)
15. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, San Francisco (1979)
16. Garey, M.R., et al.: The complexity of flowshop and jobshop scheduling. Math. Oper. Res. **1**, 117–129 (1976)
17. Garey, M.R., et al.: The planar hamiltonian circuit problem is NP-complete. SIAM J. Comput. **5**, 704–714 (1976)
18. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Elsevier, New York (2004)
19. Gonzalez, T., Sahni, S.: Flowshop and jobshop schedules: complexity and approximation. Oper. Res. **26**, 36–52 (1978)
20. Graham, R.L., et al.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discrete Math. **5**, 287–326 (1979)
21. Hall, L.A.: Approximability of flow shop scheduling. Math. Program. **82**, 175–190 (1998)
22. Halldórsson, M.M., et al.: Multicoloring trees. Inf. Comput. **180**, 113–129 (2003)
23. Johnson, S.M.: Optimal two- and three-machine production schedules with setup times included. Naval Res. Logist. **1**, 61–68 (1954)
24. Müller, H.: Hamiltonian circuits in chordal bipartite graphs. Discrete Math. **156**, 291–298 (1996)
25. Pao, L.L., Hong, C.H.: The two-equal-disjoint path cover problem of matching composition network. Inf. Process. Lett. **107**, 18–23 (2008)
26. Petrank, E.: The hardness of approximation: gap location. Comput. Complex. **4**, 133–157 (1994)
27. Rizzi, R., et al.: On the complexity of minimum path cover with subpath constraints for multi-assembly. BMC Bioinform. **15**, S5 (2014)
28. Röck, H.: Scheduling unit task shops with resource constraints and excess usage costs. Technical report, Fachbereich Informatik, Technical University of Berlin, Berlin (1983)

29. Röck, H.: Some new results in flow shop scheduling. Zeitschrift für Oper. Res. **28**, 1–16 (1984)
30. Süral, H., et al.: Scheduling unit-time tasks in renewable resource constrained flowshops. Zeitschrift für Oper. Res. **36**, 497–516 (1992)
31. Tellache, N.E.H., Boudhar, M.: Two-machine flow shop problem with unit-time operations and conflict graph. Int. J. Prod. Res. **55**, 1664–1679 (2017)
32. Tellache, N.E.H., Boudhar, M.: Flow shop scheduling problem with conflict graphs. Ann. Oper. Res. **261**, 339–363 (2018)
33. Williamson, D.P., et al.: Short shop schedules. Oper. Res. **45**, 288–294 (1997)