Multi-Operation Multi-Machine Scheduling

Weizhen Mao

The College of William and Mary, Williamsburg VA 23185, USA

Abstract. In the multi-operation scheduling that arises in industrial engineering, each job contains multiple tasks (operations) that require execution in different shops. It is assumed that in each shop there is only one machine to perform the required operations. In this paper, a parallel model of multi-operation scheduling is proposed, in which multiple machines are available in each shop to perform the same type of operations. A multi-machine open-shop scheduling problem is studied in detail.

1 Introduction

Multi-operation scheduling as a research area is motivated by questions that arise in industrial manufacturing, production planning, and computer control. Consider a large automotive garage with specialized shops [**?**]. A car may require the following work: replace exhaust system, align wheels, and tune up. These three tasks may be carried out in any order. However, since the exhaust system, alignment, and tune-up shops are in different buildings, it is impossible to perform two tasks for a car simultaneously. When there are many cars requiring services at the three shops, it is desirable to construct a service schedule that takes the least amount of total time.

The example above is in fact a special case of the traditional *multi-operation* scheduling [**?**,**?**], where m shops $S_1,\ldots,S_m$ provide different types of services to n jobs $J_1,\ldots,J_n$. Each shop S_i contains *exactly one* machine M_i that does the actual work. Each job J_j contains m tasks $T_{j1},\ldots,T_{jm}$, with the required processing times $p_{j1},\ldots,p_{jm}$, respectively. Task T_{ji} must be executed by machine M_i in shop S_i, and no two tasks of the same job can be executed simultaneously. In addition, each machine can only execute one task at any time. The scheduling is called *open-shop* scheduling (O) if the order in which a job passes through the shops is immaterial, *flow-shop* scheduling (F) if each job has the same shop ordering, e.g. $(S_1,\ldots,S_m)$, and *job-shop* scheduling (J) if the jobs may have different shop orderings.

Multi-operation scheduling problems are optimization problems and include various parameters. To define such problems, we use the *three-field classification* $\alpha|\beta|\gamma$ [**?**], where α describes the shop and machine environment, β describes the job and task characteristics, and γ describes the optimality criteria. The automotive garage example can be denoted by $O3|\ |C_{max}$, which is a three-shop ($m=3$) open-shop scheduling problem of minimizing the *maximum completion time* (*makespan*).

The above multi-operation model assumes that there is only *one* machine in each shop to execute tasks. However, this is certainly *not* the case in the parallel processing environment, where more than one machine (processor) are available to perform a certain type of operations. This motivates us to define the following *multi-operation*

multi-machine model. Suppose that there are m shops $S_1,\ldots,S_m$ and n jobs $J_1,\ldots,J_n$. Shop S_i consists of k_i machines $M_{i1},\ldots,M_{ik_i}$ to perform the same type of operations. Job J_j consists of m tasks $T_{j1},\ldots,T_{jm}$ to be executed on any machine in $S_1,\ldots,S_m$, respectively. The machines in a shop work in parallel, and depending on the specific environment, they may be *identical* (P), having the same speed, or *uniform* (Q), having different but fixed speeds, or *unrelated* (R), having different speeds for different tasks. A multi-operation multi-machine scheduling problem can also be defined by the $\alpha|\beta|\gamma$ classification. For instance, if the first field is $OmPk$, then the problem is an open-shop scheduling problem, where there are m shops, each having k identical parallel machines. The automotive garage example can be modified to include the parallel processing concept by having more than one worker (machine) in each of the three specialized shops.

As in the traditional scheduling problems, a multi-operation multi-machine schedule can be nonpreemptive or preemptive. We say that a schedule is *nonpreemptive* if the execution of any task can not be interrupted, and that a schedule is *preemptive* if the execution of a task can be interrupted and be resumed later on. A preemptive schedule is desired when there is no penalty for an interruption. Since very few multi-operation scheduling problems can be solved in polynomial time, adding the multi-machine environment makes the resulting multi-operation multi-machine scheduling even more difficult.

In this paper, we study an open-shop scheduling problem with two shops and k identical parallel machines in each shop to minimize the makespan. We organize the paper as follows. In Section 2, we give an NP-completeness proof of the nonpreemptive version. In Section 3, we present an efficient algorithm for the preemptive version. We conclude in Section 4.

2 Nonpreemptive Scheduling

The problem of interest in this section is the nonpreemptive open-shop scheduling of n jobs in two shops each with k identical machines to minimize the makespan. It is denoted by $O2Pk|\ |C_{max}$.

Theorem 1. *$O2Pk|\ |C_{max}$ is NP-complete even when $k = 2$.*

Proof Consider the corresponding decision problem, in which given a bound B, we are asked whether there is a multi-operation multi-machine schedule with makespan $C_{max} \leq B$. This decision problem of $O2P2|\ |C_{max}$ is certainly in NP. Now we show that it can be polynomially reduced from the NP-complete PARTITION [**?**]. Given any instance of PARTITION, $A = \{a_1, a_2, \ldots, a_n\}$ (positive integers), we construct an instance of the decision problem, in which there are n jobs $J_1,\ldots,J_n$. Each job J_j consists of two tasks T_{j1} and T_{j2} with processing times a_j and ε, respectively, where $0 \leq \varepsilon \leq \frac{1}{2}\min_j\{a_j\}$. Assume that there are two shops S_1 and S_2 and that each shop S_i has two identical machines M_{i1} and M_{i2}. Task T_{j1} can be executed by machines M_{11} and M_{12} only, while task T_{j2} by machines M_{21} and M_{22} only. Finally, let $B = \frac{1}{2}\sum_j a_j$.

We next show that there exists $A' \subseteq A$ such that $\sum_{a_j \in A'} a_j = \frac{1}{2}\sum_j a_j$ if and only if there is a nonpreemptive schedule with $C_{max} \leq B$ for the instance defined. If there is $A' \subseteq A$ with $\sum_{a_j \in A'} a_j = \frac{1}{2}\sum_j a_j$, without loss of generality assume that $A' = \{a_1,\ldots,a_h\}$.

A feasible schedule with $C_{max} \leq B$ of the instance can be constructed by scheduling tasks $T_{11}, \ldots, T_{h1}$ on machine M_{11} and the remaining tasks $T_{(h+1)1}, \ldots, T_{n1}$ on machine M_{12}. Let s_{j1} and f_{j1} be the starting time and finishing (completion) time of task T_{j1} in shop S_1. As for each task T_{j2} that must be executed in shop S_2, if $f_{j1} + \varepsilon \leq B$, then schedule T_{j2} in interval $[f_{j1}, f_{j1} + \varepsilon]$ on either M_{21} or M_{22}, otherwise schedule T_{j2} in interval $[s_{j1} - \varepsilon, s_{j1}]$ on either M_{21} or M_{22}. See Fig. 1 for an example.

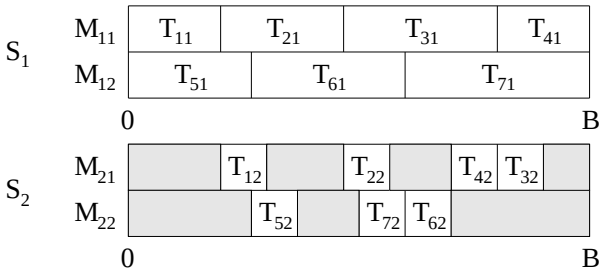

Fig. 1. A feasible schedule with $C_{max} \leq B$

If there is a feasible schedule with $C_{max} \leq B = \frac{1}{2}\sum_j a_j$ for the instance, the maximum completion time of tasks in S_1 must be $\frac{1}{2}\sum_j a_j$. Since the schedule is nonpreemptive, we define $A' = \{a_j | T_{j1}$ is executed by $\mathrm{M}_{11}\}$. Therefore, $\sum_{a_j \in A'} a_j = \frac{1}{2}\sum_j a_j$. Q.E.D.

The NP-completeness of $O2Pk|\ |C_{max}$ indicates that no polynomial-time algorithm for the problem is likely to be found and that research should focus on designing fast and good approximation algorithms.

3 Preemptive Scheduling

The problem of interest in this section is the preemptive open-shop scheduling of n jobs in two shops each with k identical machines to minimize the makespan. It is denoted by $O2Pk|pmtn|C_{max}$. We give an $O(n^2)$ algorithm for the problem.

Let $i = 1, 2$, $j = 1, \ldots, n$, and $l = 1, \ldots, k$. Let p_{ji} be the processing time of task T_{ji} of job J_j. Note that task T_{ji} must be scheduled, with preemption allowed, on any machine M_{il} in shop S_i. Define $C = \max\{\frac{1}{k}\sum_j p_{j1}, \frac{1}{k}\sum_j p_{j2}, \max_j\{p_{j1} + p_{j2}\}\}$. C is obviously a lower bound on the makespan of any preemptive schedule. If we can construct a schedule with makespan C, the schedule must be optimal. Table 1 shows an instance and its corresponding C.

We first construct a schedule for tasks T_{j1} in S_1 using McNaughton's wrap-around rule [**?**]: just fill the machines successively, scheduling the tasks in any order and splitting a task whenever the bound C is met. The schedule in Fig. 2 is one for tasks T_{j1} in the example given in Table 1. Let $C^{(1)}_{max}$ be the makespan of the schedule constructed, then $C^{(1)}_{max} \leq C$ and $C^{(1)}_{max} < C$ only when $\sum_j p_{j1} < C$.

Table 1. An inatance of $O2P2|pmtn|C_{max}$

$k=2$	J_1	J_2	J_3	J_4	J_5	$\frac{1}{k}\sum_j p_{ji}$
S_1	2	3	4	1	2	6
S_2	2	1	1	2	4	5
$p_{j1}+p_{j2}$	4	4	5	3	6	$C=6$

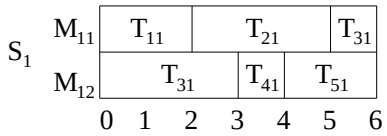

Fig. 2. A feasible schedule for S_1

We next construct a schedule for tasks T_{j2} in S_2 with makespan $C_{max}^{(2)} \leq C$. To do so, we need to know whether a task can be scheduled in a time interval. Let us consider the interval $(0,C)$ in the schedule for S_1 constructed in the previous step. We say that $t \in (0,C)$ is a *turning point* if at time t at least one machine in S_1 changes its status from executing to idling or from executing a task to executing another task. Assume that there are $h-1$ turning points $t_1 < \cdots < t_{h-1}$ in $(0,C)$. Clearly, $h-1 \leq n$. Let $t_0 = 0$ and $t_h = C$. We define an *interval* $I_r = (t_{r-1}, t_r]$ for $r = 1, \ldots, h$. Let D_r be the set of indices of the jobs whose tasks may be scheduled in interval I_r in S_2. Clearly, $n-k \leq |D_r| \leq n$. Let B_j be the set of indices of the intervals in which T_{j2} may be scheduled. Clearly, $|B_j| \leq h$ and $\sum_{r \in B_j}(t_r - t_{r-1}) = C - p_{j1}$. For the example given in Fig. 2, $I_1 = (0,2]$, $D_1 = \{2,4,5\}$, and $B_1 = \{2,3,4,5\}$.

Let x_{rj} be the length of time in I_r that T_{j2} of J_j is being executed by any machine in S_2. If we can determine x_{rj} for all r and j satisfying

$$x_{rj} = 0 \qquad \text{for } j \notin D_r, \tag{1}$$

$$\sum_{r \in B_j} x_{rj} = p_{j2} \qquad \text{for } j = 1, \ldots, n, \tag{2}$$

$$\sum_{j \in D_r} x_{rj} \leq k(t_r - t_{r-1}) \quad \text{for } r = 1, \ldots, h, \tag{3}$$

then we have a schedule for tasks T_{j2} in S_2 with $C_{max}^{(2)} \leq C$. Constraint (1) ensures that T_{j1} and T_{j2} do not overlap, constraint (2) ensures that the total amount of time for which T_{j2} is processed is the required processing time p_{j2}, and constraint (3) ensures that the total amount of time used in I_r on all k machines is no larger than that available. We use the following algorithm to determine x_{rj} for $j \in D_r$. For notational simplicity, let $\delta_r = \sum_{j \in D_r} x_{rj} - k(t_r - t_{r-1})$.

1. $x_{rj} \leftarrow \frac{p_{j2}}{C - p_{j1}}(t_r - t_{r-1})$ for $r = 1, \ldots, h$ and $j \in D_r$;
2. while there exists I_p with $\delta_p > 0$

3. locate I_q with $\delta_q < 0$;
4. define $0 \le \varepsilon_j \le x_{pj}$ for $j \in D_p \cap D_q$ such that $\sum_{j \in D_p \cap D_q} \varepsilon_j = \min\{\delta_p, -\delta_q\}$;
5. $x_{pj} \leftarrow x_{pj} - \varepsilon_j$ for $j \in D_p \cap D_q$;
6. $x_{qj} \leftarrow x_{qj} + \varepsilon_j$ for $j \in D_p \cap D_q$;

To prove the correctness of the algorithm, we wish to show that constraints (2) and (3) are satisfied by the final values of x_{rj} computed. Consider the initial values of x_{rj} in line 1. For each $j = 1, \ldots, n$, $\sum_{r \in B_j} x_{rj} = \frac{p_{j2}}{C - p_{j1}} \sum_{r \in B_j} (t_r - t_{r-1}) = p_{j2}$. In each iteration of the while loop, $\sum_{r \in B_j} x_{rj}$ remains to be p_{j2} since every time the value of some x_{pj} is decreased (line 5), the value of some x_{qj} is increased by the same amount (line 6). Therefore, constraint (2) is satisfied.

As for constraint (3), if the while loop terminates eventually, then we must have $\delta_r \le 0$, hence by definition $\sum_{j \in D_r} x_{rj} \le k(t_r - t_{r-1})$ for $r = 1, \ldots, h$, which is constraint (3). Before we show that the loop always terminates, we need Lemmas **??** and **??** to justify lines 3 and 4, respectively.

Lemma 1. *At the beginning of each iteration of the while loop, if there exists I_p with $\delta_p > 0$ (line 2), then there exists I_q with $\delta_q < 0$ (line 3).*

Proof Suppose not, we must have $\sum_r \delta_r > 0$. So $\sum_r \sum_{j \in D_r} x_{rj} - k \sum_r (t_r - t_{r-1}) = \sum_j p_{j2} - kC > 0$, and hence $C < \frac{1}{k} \sum_j p_{j2}$. This is impossible. Q.E.D.

Lemma 2. *In each iteration of the while loop, if there exist I_p and I_q with $\delta_p > 0$ and $\delta_q < 0$, then there exist $0 \le \varepsilon_j \le x_{pj}$ for $j \in D_p \cap D_q$ such that $\sum_{j \in D_p \cap D_q} \varepsilon_j = \min\{\delta_p, -\delta_q\}$ (line 4).*

Proof It is sufficient to show that $\sum_{j \in D_s \cap D_r} x_{sj} \ge \delta_s$ for any s with $\delta_s > 0$ and any $r \ne s$. Let w_{sr}^b and δ_s^b be the values of $\sum_{j \in D_s \cap D_r} x_{sj}$ and δ_s, respectively, at the *end* of the b^{th} iteration of the while loop. We wish to show that $w_{sr}^b \ge \delta_s^b$ for each b.

We induct on b. When $b = 0$, w_{sr}^0 and δ_s^0 are defined by the initial values of x_{sj} (line 1). So $w_{sr}^0 = (t_s - t_{s-1}) \sum_{j \in D_s \cap D_r} \frac{p_{j2}}{C - p_{j1}}$ and $\delta_s^0 = (t_s - t_{s-1})(\sum_{j \in D_s} \frac{p_{j2}}{C - p_{j1}} - k)$. Since $|D_s - D_s \cap D_r| \le k$ and $\frac{p_{j2}}{C - p_{j1}} \le 1$, then $\sum_{j \in D_s \cap D_r} \frac{p_{j2}}{C - p_{j1}} \ge \sum_{j \in D_s} \frac{p_{j2}}{C - p_{j1}} - k$. Therefore, $w_{sr}^0 \ge \delta_s^0$.

Assume that $w_{sr}^{b-1} \ge \delta_s^{b-1}$ for any s with $\delta_s^{b-1} > 0$ and any $r \ne s$. Now consider b. Suppose that I_p is the interval with $\delta_p^{b-1} > 0$ chosen in line 2 in the b^{th} iteration and that I_q is the interval with $\delta_q^{b-1} < 0$ chosen in line 3 in the b^{th} iteration. After we move a total of $\min\{\delta_p^{b-1}, -\delta_q^{b-1}\}$ from I_p to I_q, we have

- $w_{pq}^b = w_{pq}^{b-1} - \min\{\delta_p^{b-1}, -\delta_q^{b-1}\}$ and $\delta_p^b = \delta_p^{b-1} - \min\{\delta_p^{b-1}, -\delta_q^{b-1}\}$.
- $w_{pr}^b = w_{pr}^{b-1} - \sum_{j \in D_p \cap D_q \cap D_r} \varepsilon_j \ge w_{pr}^{b-1} - \min\{\delta_p^{b-1}, -\delta_q^{b-1}\}$ for any $r \ne p, q$.
- $w_{sr}^b = w_{sr}^{b-1}$ and $\delta_s^b = \delta_s^{b-1}$ for any $s \ne p$ with $\delta_s^b > 0$ and any $r \ne s$.

So in summary, $w_{sr}^b \ge \delta_s^b$ for any s with $\delta_s > 0$ and any $r \ne s$. Q.E.D.

Let d be the number of intervals I_r with $\delta_r \ne 0$ before the while loop is executed. We then have $d \le h \le n + 1$. In each iteration, d is decreased by at least 1. So the loop must

terminate eventually. Constraint (3) is satisfied. It is clear that once we have all the x_{rj} for $r = 1, \ldots, h$ and $j = 1, \ldots, n$ satisfying constraints (1), (2), and (3), a schedule with $C_{max}^{(2)} \leq C$ for tasks T_{j2} in S_2 can be constructed easily. We have the following theorem.

Theorem 2. *$O2Pk|pmtn|C_{max}$ is solvable in $O(n^2)$ time.*

4 Conclusions

In this paper, we introduced a parallel model for multi-operation scheduling, and studied a two-shop open-shop scheduling problem with multiple identical machines available in each shop. We proved the NP-completeness for the nonpreemptive version and gave an efficient algorithm for the preemptive version. The multi-machine multi-operation scheduling is a better and more practical model than the traditional multi-operation scheduling since it captures the essence of parallel processing that is being employed in various aspects of industrial engineering. We wish that this paper will provide a starting point for future research in this area.

References

1. Coffman, E. G. Jr.: Computer and Job Shop Scheduling Theory. John Wiley and Sons, New York (1976)
2. Garey, M. R., Johnson, D. S.: Computers and Intractability: A guide to the theory of NP-completeness. Freeman, San Francisco (1979)
3. Gonzalez, T., Sahni, S.: Open shop scheduling to minimize finish time. J. ACM **23** (1976) 665–679
4. Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., Shmoys, D. B.: Sequencing and scheduling: Algorithms and complexity. Handbooks in Operations Research and Management Science, Volume 4: Logistics of Production and Inventory, S. C. Graves, A. H. G. Rinnooy Kan and P. Zipkin, ed., North-Holland (1990)
5. McNaughton, R.: Scheduling with deadlines and loss function. Management Sci. **6** (1959) 1–12