Scheduling Unit-Length Jobs with Machine Eligibility Restrictions

Chung-Lun Li
Department of Logistics, The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong, China
Phone: +852-2766-7410; Fax: +852-2334-1765
Email: *lgtclli@polyu.edu.hk*

SHORT COMMUNICATION

Abstract

We consider uniform parallel machine scheduling problems with unit-length jobs where every job is only allowed to be processed on a specified subset of machines. We develop efficient methods to solve problems with various objectives, including minimizing a total tardiness function, a maximum tardiness function, total completion time, the number of tardy jobs, the makespan, etc.

Keywords: Scheduling; Machine eligibility restrictions; Unit-length jobs; Uniform machines

April 2004
Revised: February 2005

1. Introduction

In this paper, we consider the scheduling of unit-length jobs onto parallel machines of different speeds as well as different capabilities (or eligibility restrictions). The different capabilities of the machines cause them to be incompatible with some jobs. We consider two classes of performance measures for schedules: the min-sum objective and the min-max objective.

Let $\{J_1, J_2, \ldots, J_n\}$ be a given set of jobs and $\{M_1, M_2, \ldots, M_m\}$ be a given set of machines, where $m \le n$. For each $j = 1,2,\ldots,n$, let $\mathcal{M}_j \subseteq \{M_1, M_2, \ldots, M_m\}$ be the set of machines that is capable of processing job J_j. Let $v_i \in Z^+$ be the speed of machine M_i ($i = 1,2,\ldots,m$). Each job has a unit "length," meaning that its processing time is $1/v_i$ if it is processed by M_i. Corresponding to each job J_j is a due date d_j. We would like to schedule the jobs onto the machines so as to minimize some objective function, where job preemption is not allowed. Let C_j be the completion time of J_j, let $T_j = \max\{C_j - d_j, 0\}$ be the tardiness of J_j, let

$$U_j = \begin{cases} 1, & \text{if } C_j > d_j, \\ 0, & \text{if } C_j \le d_j, \end{cases}$$

and let f_j be a nondecreasing function ($j = 1,2,\ldots,n$). We consider the objective of minimizing $\sum_{j=1}^{n} f_j(T_j)$ as well as the objective of minimizing $\max_{j=1,\ldots,n}\{f_j(T_j)\}$. Note that these two objectives cover many commonly used performance measures in machine scheduling, including the minimization of makespan ($C_{\max}$), total completion time ($\sum C_j$), total weighted completion time ($\sum w_j C_j$), total tardiness ($\sum T_j$), total weighted tardiness ($\sum w_j T_j$), maximum tardiness ($T_{\max}$), maximum weighted tardiness ($\max\{w_i T_i\}$), number of tardy jobs ($\sum U_j$), and weighted number of tardy jobs ($\sum w_j U_j$). Using the well-known three-field representation of scheduling problems [5], our models are denoted by $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \Sigma f_j(T_j)$ and $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \max\{f_j(T_j)\}$.

Lin and Li [3] recently considered problems $Pm \mid p_j = 1, \mathcal{M}_j \mid C_{\max}$ and $Qm \mid p_j = 1, \mathcal{M}_j \mid C_{\max}$, and have developed polynomial time algorithms to solve them. Our work can be viewed as an extension of their work. We improve the computational complexities of their algorithms and extend their models to cover various other scheduling objectives.

2. Algorithms and Complexity

In this section, we develop efficient algorithms for a number of variants of our model.

Recall that function f_j is nondecreasing for $j = 1,2,\ldots,n$. Thus, for both problems $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \Sigma f_j(T_j)$ and $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \max\{f_j(T_j)\}$, an optimal solution must exist with no idle time between jobs or before the processing of the first job on each machine. Hence, it is sufficient to consider such an optimal solution. This implies that on each machine, there are n possible time slots to which the jobs may be assigned, where the starting time of the k^{th} time slot on machine M_i is $(k-1)/v_i$. Therefore, our objective is to assign n jobs to mn possible time slots in such a way that all machine eligibility constraints are satisfied and that the objective value is minimized.

2.1 Problem $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \Sigma f_j(T_j)$

We first consider problem $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \Sigma f_j(T_j)$. In this problem, the cost of assigning job J_j to the k^{th} time slot of machine M_i is given by

$$c_{jki} = \begin{cases} f_j(\max\{(k/v_i) - d_j, 0\}), & \text{if } M_i \in \mathcal{M}_j, \\ +\infty, & \text{if } M_i \notin \mathcal{M}_j, \end{cases}$$

where $j = 1,2,\ldots,n$, $k = 1,2,\ldots,n$, and $i = 1,2,\ldots,m$. Let $n_1 = n$ and $n_2 = mn$. This $n_1 \times n_2$ *bipartite weighted matching problem* can be solved in $O(n_1 S(n_1 + n_2, n_1 n_2, C))$ time using the well-known Successive Shortest Path Algorithm, where $S(n_1 + n_2, n_1 n_2, C)$ is the time needed to solve a shortest path problem with $n_1 + n_2$ nodes, $n_1 n_2$ arcs, and maximum coefficient C (see [1] and [2] for details of this time-bound and the Successive Shortest Path Algorithm). Currently, the best-known strongly polynomial time-bound for $S(u, a, C)$ is $O(a + u \log u)$. Hence, the $n_1 \times n_2$ bipartite weighted matching problem can be solved in $O(n_1(n_1 n_2 + n_2 \log n_2))$ time. Therefore, problem $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \Sigma f_j(T_j)$ can be solved in $O(n(n^2 m + nm \log nm)) = O(n^3 m)$ time.

Note that this result implies that a number of special cases of problem $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \Sigma f_j(T_j)$ can be solved in $O(n^3 m)$ time as well. This includes problems $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \Sigma w_j T_j$, $Qm \mid p_j = 1, \mathcal{M}_j \mid \Sigma w_j C_j$, and $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \Sigma w_j U_j$, as well as their unweighted versions and identical-machine versions. Some of these special cases can actually be solved by algorithms with a lower computational complexity.

Consider the special case of $Pm \mid p_j = 1, \mathcal{M}_j \mid \Sigma C_j$. After the problem is formulated as an $n_1 \times n_2$ bipartite weighted matching problem, it can be solved in $O(\sqrt{n_1} \cdot n_1 n_2 \log(n_2 C))$ time using the Cost Scaling Algorithm, where C is the largest cost coefficient (see [1] and [4]). For this special case, $c_{jki} = k$ if $M_i \in \mathcal{M}_j$. Thus, all of the cost parameters of the

bipartite weighted matching problem are integers and are no greater than n, i.e., $C \le n$. Therefore, this special case is solvable in $O(\sqrt{n} \cdot n^2 m \log(n^2 m)) = O(n^{2.5} m \log n)$ time.

The special case of $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \Sigma U_j$ can also be solved more efficiently using the same approach. Here, c_{jki} equals either 0 or 1 if $M_i \in \mathcal{M}_j$. Thus, all of the cost parameters of the bipartite weighted matching problem are integers and are no greater than 1, i.e., $C \le 1$. Therefore, this special case is also solvable in $O(n^{2.5} m \log n)$ time.

2.2 Problem $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \max\{f_j(T_j)\}$

Next, we consider problem $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \max\{f_j(T_j)\}$. To solve this scheduling problem with a min-max objective, we consider the decision problem of whether there exists a feasible schedule with $\max\{f_j(T_j)\} \le \lambda$, where λ is a given nonnegative value. Using the same argument as in subsection 2.1, the objective of this decision problem is to assign n jobs to mn possible time slots in such a way that all machine eligibility constraints are satisfied and that the cost of each assigned job in no more than λ. Thus, this is an $n \times mn$ *bipartite cardinality matching problem* in which a job J_j is allowed to be assigned to the k^{th} time slot of machine M_i if and only if $M_i \in \mathcal{M}_j$ and $f_j(\frac{k}{v_i} - d_j) \le \lambda$. To solve problem $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \max\{f_j(T_j)\}$, we need to search for the smallest possible value of λ so that the outcome of the decision problem is positive. This can be done via binary search.

Since there are $n^2 m$ possible ways of assigning n jobs to mn time slots, there are at most $n^2 m$ possible values of $\max\{f_j(T_j)\}$. Thus, the binary search requires $O(\log(n^2 m)) = O(\log n)$ iterations. An $n \times mn$ bipartite cardinality matching problem can be solved in $O(\sqrt{n} \cdot n^2 m) = O(n^{2.5} m)$ time (see [1]). Therefore, the overall running time of the algorithm is $O(n^{2.5} m \log n)$.

Note that this result implies that a number of special cases of problem $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \max\{f_j(T_j)\}$ can be solved in $O(n^{2.5} m \log n)$ time as well. This includes problems $Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid T_{\max}$, $Qm \mid p_j = 1, \mathcal{M}_j \mid \max\{w_j C_j\}$, and $Qm \mid p_j = 1, \mathcal{M}_j \mid C_{\max}$. In fact, as will be shown in the next subsection, problem $Qm \mid p_j = 1, \mathcal{M}_j \mid C_{\max}$ can be solved more efficiently by using a different method.

2.3 Problems $Qm \mid p_j = 1, \mathcal{M}_j \mid C_{\max}$ and $Pm \mid p_j = 1, \mathcal{M}_j \mid C_{\max}$

In this subsection, we discuss a polynomial time algorithm for solving problem $Qm \mid p_j = 1, \mathcal{M}_j \mid C_{\max}$. The algorithm was proposed by Lin and Li [3], and it requires solving a *maximum flow problem* in every iteration. The underlying network, denoted as $N(c)$, is depicted in Figure 1. In this network, an arc $x_i \to y_j$ with unit capacity exists if and only if

$M_i \in \mathcal{M}_j$ $(i = 1,2,\ldots,m;\ \ j = 1,2,\ldots,n)$. A binary search is employed to search for the smallest value of c that will give us a flow value of n. Such a flow in network $N(c)$ corresponds to a feasible schedule of problem $Qm \mid p_j = 1, \mathcal{M}_j \mid C_{\max}$.

In fact, Lin and Li's algorithm can also generate an optimal solution to problem $Qm \mid p_j = 1, \mathcal{M}_j \mid C_{\max}$ if the arc capacities $v_1 c, v_2 c, \ldots, v_m c$ in network $N(c)$ are replaced by $\lfloor v_1 c \rfloor, \lfloor v_2 c \rfloor, \ldots, \lfloor v_m c \rfloor$, respectively. This is because if the makespan of the schedule is c, then there are at most $\lfloor v_i c \rfloor$ jobs assigned to machine M_i. Thus, we modify Lin and Li's algorithm by adjusting the capacity of $s \to x_i$ to $\lfloor v_i c \rfloor$ for $i = 1,2,\ldots,m$. Note that since there is no idle time in the schedule, the optimal makespan of problem $Qm \mid p_j = 1, \mathcal{M}_j \mid C_{\max}$ must be equal to k / v_i for some $i = 1,2,\ldots,m$ and $k = 1,2,\ldots,n$. Hence, there are only mn possible optimal makespan values. Thus, a binary search of c requires $O(\log nm) = O(\log n)$ iterations. In each iteration, solving the maximum flow problem requires $O(n \cdot nm + n^2 \log U)$ time if the Excess Scaling Algorithm is used, where U denotes the largest arc capacity [1]. In our case, $U \le v_{\max} c \le v_{\max} n$, where $v_{\max} = \max\{v_1, v_2, \ldots, v_m\}$. Thus, the running time of each iteration is $O(n \cdot nm + n^2 \log U) \le O(n^2 (m + \log n v_{\max}))$. The overall complexity of the algorithm is $O(n^2 (m + \log n v_{\max}) \log n)$, which is an improvement on Lin and Li's complexity of $O(n^3 \log n v_{lcm})$, where v_{lcm} is the least common multiple of $v_1, v_2, \ldots, v_m$.

For the case where all machines are identical (i.e., $Pm \mid p_j = 1, \mathcal{M}_j \mid C_{\max}$), Lin and Li [3] proposed a maximum flow algorithm and reported a running time of $O(n^3 \log n)$. Their algorithm can be implemented by converting network $N(c)$ into a simple network. The resulting complexity becomes $O(n^{2.5} m^{1.5} \log n)$. In this special case, $v_{\max} = 1$, and our computational complexity becomes $O(n^2 (m + \log n) \log n)$, which again is an improvement on Lin and Li's complexity.

3. Conclusion

We have developed and analyzed several algorithms for solving a number of variants of the uniform machine scheduling problem with unit-length jobs and machine eligibility restrictions. The results are summarized in Table 1. These algorithms make use of the bipartite weighted matching, bipartite cardinality matching, and maximum flow models as well as binary searches to achieve low computational complexity.

Table 1. Summary of results.

Problem	Computational complexity
$Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \Sigma f_j(T_j)$	$O(n^3 m)$
$Pm \mid p_j = 1, \mathcal{M}_j \mid \Sigma C_j$	$O(n^{2.5} m \log n)$
$Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \Sigma U_j$	$O(n^{2.5} m \log n)$
$Qm \mid p_j = 1, d_j, \mathcal{M}_j \mid \max\{f_j(T_j)\}$	$O(n^{2.5} m \log n)$
$Qm \mid p_j = 1, \mathcal{M}_j \mid C_{\max}$	$O(n^2 (m + \log n v_{\max}) \log n)$
$Pm \mid p_j = 1, \mathcal{M}_j \mid C_{\max}$	$O(n^2 (m + \log n) \log n)$

Acknowledgment:

The author thanks Mr. Jinwen Ou for his assistance in verifying some technical details in this paper. This research was supported in part by the Areas of Strategic Development Fund of The Hong Kong Polytechnic University.

References:

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] T.C.E. Cheng, Z.L. Chen, and C.-L. Li, Parallel-machine scheduling with controllable processing times. *IIE Transactions*, 28 (1996) 177-180.

[3] Y. Lin and W. Li, Parallel machine scheduling of machine-dependent jobs with unit-length. *European Journal of Operational Research*, 156 (2004) 261-266.

[4] J.B. Orlin and R.K. Ahuja, New scaling algorithms for the assignment and minimum mean cycle problems. *Mathematical Programming*, 54 (1992) 41-56.

[5] M. Pinedo, *Scheduling: Theory, Algorithms, and System*s, 2nd Edition, Prentice Hall, Upper Saddle River, NJ, 2002.

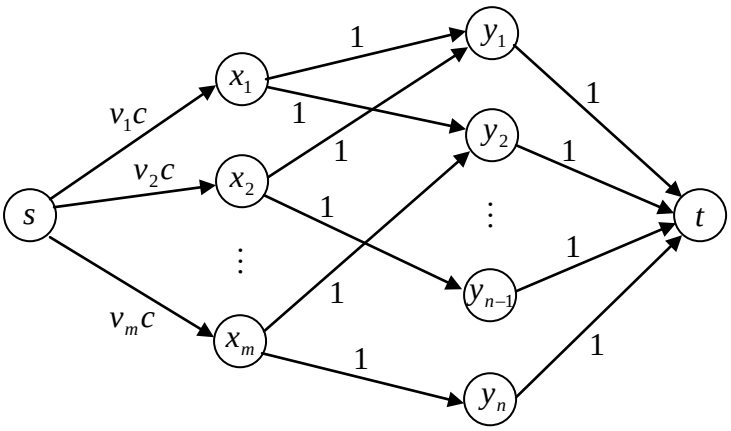

Figure 1. Network $N(c)$.