Viewpoint on "A note on unrelated parallel machine scheduling with time-dependent processing times"

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

March 2008
Revised April 2008

In a recent article, Kuo *et al* study an unrelated parallel machine scheduling problem in which the processing time of a job is a linear function of its start time and the objective is to minimize the sum of job completion times (Kuo *et al*, 2008). They show that problem is polynomial-time solvable when the number of machines is fixed. However, the computational complexity of their solution method is high. We provide here a modified solution method with an improved running time complexity.

We follow Kuo *et al*'s notation and consider the following two cases: Case 1: For every job J_j scheduled on machine M_i, the processing time of J_j is $p_{ij} = a_{ij} + b_i t_{ij}$. Case 2: For every J_j scheduled on M_i, the processing time of J_j is $p_{ij} = a_{ij} - b_i t_{ij}$. Here, $a_{ij} \geq 0$ and $b_i > 0$ are constants, and t_{ij} is the start time of processing of the job. Case 1 is denoted as $P/p_{ij} = a_{ij} + b_i t_{ij} / \Sigma C_j$, while Case 2 is denoted as $P/p_{ij} = a_{ij} - b_i t_{ij} / \Sigma C_j$. In Case 2, we assume that $0 < b_i < 1$ and $b_i(t_0 + \sum_{k=1}^{n} a_{ik} - a_{ij}) < a_{ij}$ for $i = 1,2,\ldots,m$ and $j = 1,2,\ldots,n$, where $t_0 \geq 0$ is the ready time of every job. The condition "$0 < b_i < 1$" ensures that the decrease of a job's processing time is less than one unit for every unit delay in its start time, while the condition "$b_i(t_0 + \sum_{k=1}^{n} a_{ik} - a_{ij}) < a_{ij}$" ensures that all job processing times are positive in any feasible schedule.

Note that in Kuo *et al*'s model, b_i is job independent. This assumption significantly reduces the complexity of $P/p_{ij} = a_{ij} \pm b_i t_{ij} / \Sigma C_j$. Let n_i denote the number of jobs assigned to M_i ($i = 1,2,\ldots,m$) and $P(n,m) = (n_1, n_2, \ldots, n_m)$ denote the "allocation vector", where $\sum_{i=1}^{m} n_i = n$. Kuo *et al*'s solution method enumerates all possible values of $P(n,m)$, and for each $P(n,m)$, it solves the following problem:

$$
\begin{aligned}
\textbf{AP}: \quad & \text{minimize} && \sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{r=1}^{n_i} W_{jir} y_{jir} \\
& \text{subject to} && \sum_{j=1}^{n} x_{jir} = 1 \quad (i = 1,2,\ldots,m;\ r = 1,2,\ldots,n_i) \\
& && \sum_{i=1}^{m} \sum_{r=1}^{n_i} x_{jir} = 1 \quad (j = 1,2,\ldots,n) \\
& && x_{jir} = 0 \text{ or } 1 \quad (j = 1,2,\ldots,n;\ i = 1,2,\ldots,m;\ r = 1,2,\ldots,n_i).
\end{aligned}
$$

In this formulation, $x_{jir} = 1$ if J_j is assigned to the rth position on machine M_i, and $x_{jir} = 0$ otherwise. If $t_0 = 0$, then $W_{jir} = a_{ij} \sum_{k=0}^{n_i - r} (1 + b_i)^k$ for problem $P/p_{ij} = a_{ij} + b_i t_{ij} / \Sigma C_j$, and $W_{jir} = a_{ij} \sum_{k=0}^{n_i - r} (1 - b_i)^k$ for problem $P/p_{ij} = a_{ij} - b_i t_{ij} / \Sigma C_j$. Here, W_{jir} represents the increase in objective function value of the problem when J_j is inserted into the rth position of M_i. (Note: If $t_0 > 0$, then W_{jir} should be set equal to $t_0 + (a_{ij} + b_i t_0) \sum_{k=0}^{n_i - r} (1 + b_i)^k$ and $t_0 + (a_{ij} - b_i t_0) \sum_{k=0}^{n_i - r} (1 - b_i)^k$ for problems $P/p_{ij} = a_{ij} + b_i t_{ij} / \Sigma C_j$ and $P/p_{ij} = a_{ij} - b_i t_{ij} / \Sigma C_j$, respectively.)

We now analyze the running time of Kuo *et al*'s method. Problem **AP** is an $n \times n$ assignment problem (i.e., assigning n jobs to a total of n positions spread across m machines). Solving **AP** requires $O(n^3)$ time. The number of possible allocation vectors is $\binom{n+m-1}{m-1}$ (see Mosheiov, 2001). Hence, the overall running time of Kuo *et al*'s method is $O(\binom{n+m-1}{m-1} n^3)$. When m is fixed, $O(\binom{n+m-1}{m-1}) = O(n^{m-1})$, and therefore, the running time of

Kuo *et al*'s method is $O(n^{m+2})$. This is consistent with Corollary 1 of Kuo *et al* (2008), which states that when $m=2$, $P/p_{ij}=a_{ij}\pm b_i t_{ij}/\Sigma C_j$ can be solved in $O(n^4)$ time using their method. Next, we consider the case in which m is not fixed. In particular, we consider the situation where $n>m$. Note that

$$\binom{n+m-1}{m-1}=\frac{n+m-1}{m-1}\cdot\frac{n+m-2}{m-2}\cdots\frac{n+1}{1}>2\cdot 2\cdots 2=2^{m-1}.$$

Therefore, in this case the running time of Kuo *et al*'s method is not polynomially bounded.

Although Kuo *et al* have shown that $P/p_{ij}=a_{ij}\pm b_i t_{ij}/\Sigma C_j$ are polynomially solvable when m is fixed. The computational complexity of their suggested solution method is quite high, particularly when m is large. We now propose an improved solution method. We describe our method for $P/p_{ij}=a_{ij}+b_i t_{ij}/\Sigma C_j$, while $P/p_{ij}=a_{ij}-b_i t_{ij}/\Sigma C_j$ can be solved similarly.

Let $(J_{\pi(i,n_i)},J_{\pi(i,n_i-1)},\ldots,J_{\pi(i,1)})$ denote the sequence of jobs processed by M_i ($i=1,2,\ldots,m$). Using the same argument as in the proof of Lemma 1 in Kuo *et al* (2008), it is easy to show that the total completion time of the jobs on M_i is

$$\sum_{s=1}^{n_i}C_{\pi(i,s)}=\sum_{s=1}^{n_i}\left[t_0+(a_{i,\pi(i,s)}+b_i t_0)\sum_{k=0}^{s-1}(1+b_i)^k\right]$$

for $i=1,2,\ldots,m$. Let $\overline{W}_{jis}=t_0+(a_{ij}+b_i t_0)\sum_{k=0}^{s-1}(1+b_i)^k$. (Note that unlike W_{jir}, the quantity $\overline{W}_{jis}$ is independent of n_i.) Define $y_{jis}=1$ if J_j is the sth last job processed by M_i, and $y_{jis}=0$ otherwise. Then, for $i=1,2,\ldots,m$,

$$\sum_{s=1}^{n_i}C_{\pi(i,s)}=\sum_{s=1}^{n_i}\sum_{j=1}^{n}\overline{W}_{jis}y_{jis}=\sum_{s=1}^{n}\sum_{j=1}^{n}\overline{W}_{jis}y_{jis},$$

where the second equality holds because $y_{jis}=0$ when $s>n_i$. Hence, the problem can be formulated as the following $n\times nm$ "constrained asymmetric assignment problem", where we would like to assign the n jobs to nm positions, with n positions on each machine, leaving a total of $nm-n$ positions unassigned:

$$\begin{aligned}
\mathbf{AP}':\quad & \text{minimize} && \textstyle\sum_{j=1}^{n}\sum_{i=1}^{m}\sum_{s=1}^{n}\overline{W}_{jis}y_{jis} && \\
& \text{subject to} && \textstyle\sum_{j=1}^{n}y_{jis}\le 1 \quad (i=1,2,\ldots,m;\ s=1,2,\ldots,n) && (1)\\
& && \textstyle\sum_{i=1}^{m}\sum_{s=1}^{n}y_{jis}=1 \quad (j=1,2,\ldots,n) && (2)\\
& && \textstyle\sum_{j=1}^{n}y_{ji1}\ge\sum_{j=1}^{n}y_{ji2}\ge\cdots\ge\sum_{j=1}^{n}y_{jin} \quad (i=1,2,\ldots,m) && (3)\\
& && y_{jis}=0 \text{ or } 1 \quad (j=1,2,\ldots,n;\ i=1,2,\ldots,m;\ s=1,2,\ldots,n). && (4)
\end{aligned}$$

In this formulation, constraints (4) ensure that on every machine, the unassigned positions must precede all assigned positions, so that $y_{jis}=1$ if and only if J_j is truly the sth last job on machine M_i.

Note that $\overline{W}_{ji1}\le\overline{W}_{ji2}\le\cdots\le\overline{W}_{jin}$ for all $i=1,2,\ldots,m$ and $j=1,2,\ldots,n$. Thus, constraints (3) can be removed from the formulation without affecting the optimal solution value of the

problem. In other words, solving the following problem will provide us with an optimal solution to $\mathbf{AP}'$:

$$\mathbf{AP}'': \quad \text{minimize} \quad \sum_{j=1}^{n}\sum_{i=1}^{m}\sum_{s=1}^{n}\overline{W}_{jis}y_{jis}$$
$$\text{subject to} \quad \text{constraints (1), (2) and (4).}$$

Note: A rigorous proof of this statement can be developed easily by considering any optimal solution to $\mathbf{AP}''$ and showing that an alternative feasible solution can be created through adjusting the y_{jis} values (say, by repeatedly swapping the values of y_{jis} and $y_{j,i,s+1}$ whenever $\sum_{j=1}^{n} y_{jis} < \sum_{j=1}^{n} y_{j,i,s+1}$) so that the new solution satisfies (3) and has an objective function value no worse than the original solution.

Problem $\mathbf{AP}''$ is an $n \times nm$ (unconstrained) "asymmetric assignment problem" with n jobs and nm positions. One simple way to solve this asymmetric assignment problem is to convert it to an $nm \times nm$ assignment problem by introducing $nm - n$ dummy jobs. Solving the converted assignment problem requires $O(m^3 n^3)$ time. Thus, when m is fixed, the running time of this solution method becomes $O(n^3)$, regardless what the value of m is. Therefore, this method is significantly more efficient than Kuo *et al*'s method for large-sized problems.

When m is not fixed, this solution method has a polynomial running time of $O(m^3 n^3)$. This is again a significant improvement over Kuo *et al*'s method. In fact, in this case, $\mathbf{AP}''$ can be solved with a lower computational complexity using a more sophisticated method. Note that an $n_1 \times n_2$ asymmetric assignment problem is the same as a minimum weighted bipartite matching problem with an underlying bipartite graph $G = (N_1 \cup N_2, A)$, where $n_1 = |N_1|$ and $n_2 = |N_2| > n_1$, and all the nodes in N_1 are required to be matched. This minimum weighted bipartite matching problem can be solved in $O(n_1(|A| + n_2 \log n_2))$ time using the Successive Shortest Path Algorithm (see Cheng *et al*, 1996 and Ahuja *et al*, 1993, ch. 12). Therefore, $\mathbf{AP}''$ can be solved in $O(n^3 m + n^2 m \log(nm))$ time. This implies that when m is not fixed, problem $P/p_{ij} = a_{ij} + b_i t_{ij} / \Sigma C_j$ can be solved in $O(n^3 m + n^2 m \log(nm))$ time.

Problem $P/p_{ij} = a_{ij} - b_i t_{ij} / \Sigma C_j$ can also be solved with the same efficiency using the above method, except that $\overline{W}_{jis}$ is redefined as $t_0 + (a_{ij} - b_i t_0)\sum_{k=0}^{s-1}(1-b_i)^k$.

References

Ahuja RK, Magnanti TL and Orlin JB (1993). *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall: Englewood Cliffs, NJ.

Cheng TCE, Chen ZL and Li C-L (1996). Parallel-machine scheduling with controllable processing times. *IIE Transactions* **28:** 177-180.

Kuo W-H, Hsu C-J and Yang D-L (2008). A note on unrelated parallel machine scheduling with time-dependent processing times. *Journal of the Operational Research Society,* forthcoming, doi:10.1057/palgrave.jors.2602576

Mosheiov G (1992). Parallel machine scheduling with a learning effect. *Journal of the Operational Research Society* **52:** 1165-1169.