UET flow shop scheduling with intree precedence constraints on two machines

N. Labehri
Département de mathématiques
Université M. Mammeri
Bp 531, 15000 Tizi-Ouzou

D. Rebaine
Département de mathématiques
Université M. Mammeri
Bp 531, 15000 Tizi-Ouzou

Abstract: In this paper, a linear algorithm is presented to schedule, in a two-machine flow shop environment n unit execution time (UET) jobs, related by an intree precedence constraints, so as to minimize simultaneously the overall completion time and the mean completion time.

Key words: scheduling, flow shop, Complexity.

Résumé: Dans cet article, un algorithme de complexité linéaire est présenté pour ordonnancer, dans un environnement de type flow shop, sur deux machines n jobs unitaires reliés par des contraintes de précédence de type arborescence entrante, pour minimiser simultanément les deux critères que sont la durée totale et le temps moyen d'achèvement d'un ordonnancement.

Mots clés: Ordonnancement, Flow, Makespan, Complexité.

1. Introduction

The usual flow shop problem can be described as follows.

Given are a set of n jobs and a set of m machines. Each machine can handle at most one job at a time and each job can be processed by at most one machine at a time. Each job consists of m tasks indexed by $1, \ldots, m$ and the i-th task of a job precedes its $(i+1)$-th task for $i = 1, \ldots, m-1$. Further, the i-th task of the j-th job has to be carried out on the i-th machine, during an uninterrupted period of time, l_{ij}. The purpose is to find a schedule of all the jobs which minimizes the overall completion time (also called the *Makespan*) and the mean finish time.

When there is no precedence constraint on the set of jobs, flow shop scheduling with respect to the *Makespan* is shown to be $\mathcal{NP}$-complete in the strong sense [3], even for the case $m = 3$. However, for the special case $m = 2$, there exists a polynomial time algorithm [2]. Concerning the mean finish time, the corresponding problem is $\mathcal{NP}$-hard even for the case $m = 2$ and preemption is allowed [5].

On the other hand, when precedence constraints on the jobs are considered, we usually distinguish between two types of dependency: a job i precedes job j means that the latter job can only start processing its first task just after the first task the former job has been completed; whereas, in the second type, the first task of job j can only start its first task after the last task of job i has been completed.

As far as the complexity aspects are concerned, in the first type of precedence constraints, the corresponding two-machine flow shop problem, with respect to the *Makespan*, is solvable in $O(n \log n)$ even for a series-parallel precedence graph [7], but becomes again $\mathcal{NP}$-hard for general precedence constraints [6]. In the second form of the precedence constraints, the corresponding two-machine flow shop problem becomes $\mathcal{NP}$-hard even for a tree like precedence graph [3].

In this paper, we consider the case in which the jobs are related by a tree (more precisely an intree, see the definition given below) precedence of the second type and the execution times are restricted to unity. Following the Graham's notation, these two problems are denoted respectively by $F|UET, intree|C_{\max}$ and $F|UET, intree|\sum_{i=1}^{n} C_i$

In [4], it was reported that Lageweg solved polynomially the $F2|UET, tree|C_{\max}$ problem and the $F2|UET, tree|\sum_{i=1}^{n} C_i$ problem[1]. But, to the best of our knowledge, Lageweg's algorithms and the proof of their optimality have not appeared yet in the literature. In this paper, we reconsider the above two problem and solve them simultaneously by a single algorithm[2] *i.e.*, we are solving the problem

[1] A tree is either an intree or an outtree.

[2] In other words we are solving a bicriteria problem. In the multicriteria optimization terminology, the solution achieved by our algorithm is called *an ideal point.*

Theorem 7. *If opt denotes the value of an optimal schedule length and m the number of machines then*

$$opt \geq \max_{1 \leq i \leq h} \{m \times i + |V_i| - 1\}$$

Proof: From lemma 5, we can assume that an optimal schedule is a level schedule. Even if the precedence constraints are disregarded, it takes at least $|V_i|$ to complete the last processed vertex of V_i on the first machine for which $(m-1)$ units of times must be added to finish its processing on the $(m-1)$ remaining machines. It then takes at least $(i-1)m$ units of time to schedule the remaining vertices related by the precedence constraints containing at least one chain of height $(i-1)$ (see lemma 6). Consequently, we obtain

$$\begin{aligned} \text{opt} &\geq \max_{1 \leq i \leq h} \{(i-1)m + |V_i| + m - 1\} \\ &= \max_{1 \leq i \leq h} \{m \times i + |V_i| - 1\} \square \end{aligned}$$

Lemma 8. *The mean finish time on m machines of a chain of length i, starting at time t, is exactly $i(2t + m \times i + m)/2$, for any given schedule.*

Proof: To complete any vertex v in a chain, it takes exactly m units of times. Furthermore, a successor of v cannot start its processing on the first machine vertex until v has finished its processing on the m-th machine. Thus, summing all together the completion times of the i vertices establishes the statement of the lemma. □

Theorem 9. *If C_i denotes the value of the completion time of job i in an optimal mean finish time schedule and m the number of the machines then*

$$\sum_{i=1}^{n} C_i \geq \max_{1 \leq i \leq h} \left\{ \frac{1}{2}|V_i|\,(|V_i| + 2m - 1) + \frac{1}{2}(i-1)\,(2|V_i| + m(i+2) - 2) + \min(A_i, B_i) \right\}.$$

where

$$\begin{aligned} A_i &= \tfrac{1}{2}(i-2)(m-1)\,(2|V_i| + m(i-2))\,; \\ B_i &= \tfrac{1}{2}\lfloor R/(m-1)\rfloor(m-1)\,(2|V_i| + m\lfloor R/(m-1)\rfloor) \\ &\quad + \tfrac{1}{2}R \bmod (m-1)\,(2|V_i| + 2m\lfloor R/(m-1)\rfloor + R \bmod (m-1))\,; \\ R &= n - |V_i| - (i-1). \end{aligned}$$

Proof: Without loss of generality, let us assume the optimal schedule is a level schedule. Even if the precedence constraints are disregarded, the mean finish time of the set V_i is at least

$$\sum_{j=m}^{|V_i|+m-1} j = \frac{1}{2}|V_i|\,(|V_i| + 2m - 1)$$

The remaining $n - |V_i|$ vertices are related by a least one chain of length $i-1$ (including the root). The earliest time the first vertex of that chain (say C) can start processing is $|V_i| + m - 1$. Thus, from lemma 8, by letting t to be $|V_i| + m - 1$, the mean finish time of C is

$$\begin{aligned}\sum_{i \in C} C_i &= \frac{1}{2}(i-1)\left(2(|V_i| + m - 1) + m(i-1) + m\right) \\ &= \frac{1}{2}(i-1)(2|V_i| + m(i+2) - 2).\end{aligned}$$

Observe that on each machine, when scheduling a chain, there exist $(m-1)$ time slots between the execution of two successive vertices. Thus, for the chain C, there are $(i-1)(m-1)$ such time slots. Because of the root, only $(i-2)(m-1)$ are schedulable time slots.

Let us first assume that $R = n - |V_i| - i + 1 < (m-1)(i-2)$. From the above observation and from the fact that the earliest time a job of the R remaining vertices can start processing is $t = |V_i|$, it follows that

$$\begin{aligned}\sum_{j \in J - V_i - C} C_i \geq & \sum_{k=1}^{\lfloor R/(m-1) \rfloor} (m-1)\left(t + km\right) + \lfloor R/(m-1) \rfloor \times \sum_{k=1}^{m-1} k \\ & + \sum_{k=1}^{R \bmod (m-1)} \left(t + m\lfloor R/(m-1) \rfloor + k\right)\end{aligned}$$

Thus, doing all the computations yield the first bound of the lemma.

Now, if $R \geq (m-1)(i-2)$, then it is clear that the mean finish time of the R remaining vertices cannot be better than the mean finish time of $(m-1)(i-2)$ vertices. Thus, doing the same computations as above by replacing R by $(m-1)(i-2)$, and observing that in this case $R \bmod (m-1) = 0$, we obtain

$$\sum_{j \in J - V_i - C} C_i \geq \sum_{k=1}^{i-2} (m-1)\left(t + (k-1)m\right) + (i-2) \times \sum_{k=1}^{m-1} k$$

Thus, again doing all the necessary computations yield the other bound of the lemma.

The statement of the lemma is established by taking the minimum over the two bounds and the maximum over the h different V_i. □

Example 10. *Let us consider the tree of figure 1. The different V_i which can be obtained are:*

$$V_1 = \{1, 2, \ldots, 13\} \quad V_2 = \{1, 2, \ldots, 12\} \quad V_3 = \{1, 2, \ldots, 10\}$$
$$V_4 = \{1, 2, \ldots, 9\} \quad V_5 = \{1, 2, \ldots, 7\} \quad V_6 = \{1, 2, \ldots, 4\}.$$

With regard to the finish time, the lower bound obtained on two machines is

$$opt \geq \max\{2 + 13 - 1, 4 + 12 - 16 + 10 - 1, 10 + 7 - 1, 12 + 4 - 1\}$$
$$= 16$$

Similarly for the mean finish time on two machines, we obtain

$$\sum_{i=1}^{n} C_i \geq \max\{104, 90 + 15, 65 + 12 + 28, 54 + 42 + 11, 35 + 52 + 20, 14 + 55 + 36\}$$
$$= 107$$

3. An algorithm for the two-machine case

In this section, we present a polynomial algorithm to solve the above problem for the case of two machines.

The idea of our algorithm, which is almost the same as Hu's algorithm (see for example [1] for its description) which solves the same problem but on m identical machines, is to schedule the vertices level by level, starting from the highest level giving the priority to the vertices without predecessors in that level. This procedure is kept as long as long the number of the vertices in a level is strictly greater than one. When the level of a tree is reduced to a single vertex, the highest vertex without a predecessor is then scheduled before or after the vertex of the considered level, depending on wether the last processed vertex is the successor of the single vertex of the considered level. Formally, the algorithm is as below.

Algorithm 1

1. set $L := h$; lastsched:= 0;
2. Let S_{h-L+1} be the set of vertices of level L;
3. while $L \geq 1$ do
   begin
   a. if $|S_{h-L+1}| = 1$ (say $\{x_1\}$) then
      begin
      a.1. let y be the highest vertex such that pred(y)= $\emptyset$;
      a.2. if y exists then
         begin
         . if pred(x_1) = lastsched
            then $S_{h-L+1} := S_{h-L+1} \cup \{y\}$
            else $S_{h-L+1} := \{y\} \cup S_{h-L+1}$;
         . delete y from the tree;
         . lastsched:= last(S_{h-L+1});
         end;
      end;
   b. schedule the jobs in S_{h-L+1} from left to right on both machines;
   c. set $L := L - 1$;
   d. Set $S_{h-L+1} = W_{h-L+1} \cup succ(S_{h-L+2} - \{y\})$; where W_{h-L+1} denotes the set of vertices of level L without predecessors;
   end;

Example 11. *Let the following tree be the precedence constraints of a 13-job problem on two machines. At the successive iterations of the above algorithm, the lists S_i are generated as shown in figure 1.*

The final schedule generated by the above algorithm is as described by the Gantt diagram of figure 2.

The value of the *Makespan* of the above schedule is 16 whereas the value of its mean finish time is 107.

Theorem 12. *The running time of algorithm 1 is $O(n)$.*

Proof: The above algorithm consists, for each level L; $h \leq L \leq 1$, of constructing a list S_L, searching eventually for a vertex (without a predecessor), and then scheduling that list. It is clear that the construction and the scheduling of the list S_L takes a time of $O(|\text{level } h - L + 1|)$. It is also clear that searching for the highest vertex x such that pred(x) = $\emptyset$ takes a time proportional to the height of the tree. Let this

time be $O(x_k)$ which corresponds to the complexity of step (a). Thus, the complexity of the while-loop is

$$\begin{aligned} t(n) &= \sum_{L=1}^{h} O(|\text{level } h - L + 1|) + \sum_{L=1}^{h} O(x_k) \\ &= O\left(\sum_{L=1}^{h} |\text{level } h - L + 1|\right) + O\left(\sum_{L=1}^{h} x_k\right) \end{aligned}$$

It is clear that the sum of the cardinality of the levels is nothing else than the number,n, of the vertices. When it comes to search for a new vertex at step (a), this procedure starts from the level where the last search of this kind has finished. It follows then that

$$O\left(\sum_{L=1}^{h} x_k\right) = O(h)$$

Thus, the complexity of the while-loop which dominates the complexity of the above algorithm is $O(n)$. □

Observe that the successive lists $S_1, S_2, \ldots S_h$ generated by the above algorithm correspond to the set of vertices of the same level. In case a given level is reduced to a single vertex, the corresponding list contains another job of a lower level. Let S_f be the final list that selects at least two vertices from a single level $h - f + 1$. If no such stage exists then $S_f = S_0 = \emptyset$.

3.1. Minimizing the Makespan. To prove the optimality of the above algorithm, with respect to the overall completion time criterion, we will show that its

value on the schedule generated by the above algorithm is exactly the lower bound given in theorem 7.

Lemma 13. *All the lists $S_1, S_2 \ldots, S_f$ schedule only vertices from the set V_{h-f+1}.*

Proof: If, at each iteration of the algorithm, the corresponding list S_L; $1 \leq L \leq h-f+1$, contains more than two vertices of the same level then the result of the lemma follows immediately. Suppose now that there exists a list S_r that selects a vertex y such that its level $t < h-f+1$. From the algorithm, level $h-r+1$ is reduced to a single vertex. Furthermore, t is the highest level such that pred$(y)=\emptyset$ at that stage in the tree. Let us distinguish two cases:

case 1 There exists a vertex p in a list S_b, $r+1 \leq b \leq f$ such that pred$(p)=\emptyset$: This case cannot occur as the algorithm would select vertex p before vertex y. Consequently, $t \geq h-f+1$.

case 2 There does not exist a vertex without a predecessor up to list S_f: This means that all vertices associated to the lists $S_{r+1}, \ldots S_f$ have predecessors. But, as mentioned above, we know that level $h-r+1$ is reduced to a single vertex, say x. It then follows that vertex x has at least two successors which cannot be true in an intree. Consequently, $t \geq h-f+1$. □

Lemma 14. *If S_f exists then it takes exactly $|V_{h-f+1}|+1$ units of time to complete on two machines the vertices of the lists S_1 up to S_f.*

Proof: It is clear that if a vertex x is completed on the first machine at time t, then its successor can only start its processing on the first machine from time $t+1$. To ovoid an idle time on the first machine, it suffices to schedule at time t either a vertex whose predecessor has been scheduled before vertex x or a vertex without a predecessor. By definition, S_f is the last list which contains more than one vertex from a single level. This means that the cardinality of all the preceding lists S_i is greater than two. Let us distinguish two cases:

case 1 S_i contains only vertices from one single level: As the vertices of S_i are scheduled in a FIFO manner, relativyely to their predecessors giving the priority to the vertices without predecessors, clearly no idle time can occur when scheduling on the first machine the vertices of S_i.

case 2 S_i contains a vertices from different levels: This case only occurs when a given level is reduced to a single vertex, say x_1, in that stage of the tree. In that case, a vertex of a lower level and without a predecessor is scheduled either before

x_1 if pred(x_1) is the same as last(S_{i-1}) or after x_1 otherwise. Thus, no idle time can occur when scheduling on the first machine the vertices of S_i.

Consequently, in either case, when it comes to schedule lists S_i, $1 \leq i \leq f$ no idle time occurs on the first machine. The result of the lemma is thus established. □

Lemma 15. *The lists S_{f+1} up to S_h contain at most $(h - f - 2)$ vertices without predecessors.*

Proof: By definition of S_f, when it comes to schedule S_k; $f + 1 \leq k \leq h$; the corresponding level is reduced to a single vertex. From the algorithm, this single vertex is scheduled with a vertex without a predecessor. Furthermore, when it comes to schedule the root of the tree, contained in the list S_h, there is no vertex left to schedule. As the highest level left when scheduling list S_{f+1} is $(h - f)$, the result of the lemma follows immediately.

Lemma 16. *It takes exactly $2(h - f)$ units of times to schedule on both machines the lists S_{f+1} up to S_h.*

Proof: It is clear that the highest vertex left after scheduling S_f is at level $(h - f)$. Thus, from lemma 6, it takes exactly $2(h - f)$ units of time to schedule the longest chain left in the tree. Because of the precedence constraints, there exists an idle period of time generated by the schedule of the last vertex scheduled in V_{h-f+1} and the first vertex of the longest left chain if S_f exists, followed by $(h - f - 1)$ other idle periods generated by the longest chain of length $(h - f)$. From lemma 15, at most $(h - f - 2)$ vertices without predecessors are left in the tree after the schedule of S_f. Thus, there are enough time slots to plot those vertices as they are scheduled before each vertex of the chain of length $(h - f)$, if S_f exists, otherwise they are scheduled after each vertex of the chain of length h. Consequently, the lemma is established. □

Theorem 17. *The above algorithm generates an optimum schedule*

Proof: We know from lemma 14 that it takes $|V_{h-f+1}| + 1$ units of time to schedule the lists S_1 up to S_f, if list S_f exists. From lemma 16, it takes $2(h - f)$ units of time to schedule the vertices of lists S_{f+1} up to S_h. Thus, to schedule the whole tree, it takes either $2h$ units of times if S_f does not exists, or $|V_{h-f+1}| + 1 + 2(h - f)$ units of time otherwise which is nothing else than the lower bound given in theorem 7 for $m = 2$ and $i = h$ for the first value or $i = h - f + 1$ for the second value. □

3.2. Minimizing the mean finish time. In this section, we prove that the mean finish time of the schedule generated by algorithm 1 is optimal by showing that the lower bound given in theorem 9 is achieved. Before proceeding further, let us rewrite the lower bound of theorem 9 for $m = 2$. Observe, in this case, we have $x \bmod (m-1) = 0$ for any integral x. Thus, we obtain the following simplified formulae

$$\sum_{i=1}^{n} C_i \geq \max_{1 \leq i \leq h} \frac{1}{2}|V_i|\,(|V_i|+3) + (i-1)\,(|V_i|+i+1) + \min(A_i, B_i) \qquad (1)$$

where

$$\begin{aligned} A_i &= (i-2)\,(|V_i|+i-2) \\ B_i &= (n-i+1)(n-|V_i|-i+1) \end{aligned}$$

Lemma 18. *The mean finish time of the vertices of the lists S_1 up to S_f is exactly*

$$\frac{1}{2}|V_{h-f+1}|\,(|V_{h-f+1}|+3)$$

.

Proof: We know from lemma 13 that lists S_1 up S_f schedule only the vertices of V_{h-f+1}. Further those lists are scheduled without an idle time on the first machine (lemma 14). Thus, the mean finish time of the vertices of S_1 up to S_f is $\frac{1}{2}|V_{h-f+1}|\,(|V_{h-f+1}|+3)$. □

Lemma 19. *if list S_f exists then the mean finish time of the vertices of the lists S_{f+1} up to S_h is exactly*

$$\sum_{i \in \bigcup_{k=f+1}^{h} S_k} C_i = (h-f)\,(|V_{h-f+1}|+h-f+2) + (n-h+f+1)(n-|V_{h-f+1}|-h+f).$$

Proof: It follows from the definition of S_f that the lists S_k, $f+1 \leq k \leq h$, contain at most two vertices $\{x, y\}$: x is at level $h-k+1$ and y is at a lower level than x. Thus, the vertices of type x form a chain (say C) of length $(h-f)$. Because $\mathrm{pred}(x)$ is scheduled at the last position in S_f, then first vertex x starts at time $|V_{h-f+1}|+1$. Thus, the mean finish time of C is

$$\begin{aligned} \sum_{i \in C} C_i &= \frac{1}{2}(h-f)\,(2(|V_{h-f+1}|+1)+2(h-f)+2) \\ &= (h-f)(V_{h-f+1}+h-f+2). \qquad (2) \end{aligned}$$

The total number of vertices of type x_{k2} is $R = n - |V_{h-f+1}| - (h-f)$. We know from lemma 15 that $R \leq h - f - 2$. As each y is scheduled before each x, it follows that the mean finish time of these vertices is

$$\begin{aligned} \sum_{i \in J - V_i - C} C_i &= \sum_{i=1}^{R} (|V_{h-f+1}| + (2k-1)) \\ &= (n-h+f)(n-|V_{h-f+1}|-h+f). \end{aligned} \tag{3}$$

Adding all together (2) and (3) yield the result of the lemma. □

Lemma 20. *if list S_f does not exist then the mean finish time of the vertices of the lists S_{f+1} up to S_h is exactly*

$$\sum_{i \in \bigcup_{k=f+1}^{h} S_k} C_i = h(h+1) + (n-h)(n-h+2).$$

Proof: Following the same argument as above and by taking into account that, in this case,

1. each vertex of type x is scheduled after each vertex of type y, and
2. the first vertex x of the chain of length h is scheduled at time 0,

the result of the lemma follows immediately.

Theorem 21. *Algorithm 1 generates an optimum schedule*

Proof: As $R \leq h - f + 2$ (see lemma 15), we have that $\min(A_{h-f+1}, B_{h-f+1}) = B_{h-f+1}$. We know from lemma 18 and lemma 19 that the mean finish time of the lists S_1 up to S_h is

$$\begin{aligned} \sum_{i \in S_i} C_i &= \frac{1}{2}|V_{h-f+1}|\,(|V_{h-f+1}|+3) + (h-f)(V_{h-f+1}+h-f+2) + \\ &\quad + (n-h+f)(n-|V_{h-f+1}|-h-f). \end{aligned}$$

which nothing else than the lower bound given by in (1) for $i = h - f + 1$. □

4. Conclusion

In this paper, we have presented a linear algorithm to solve the two-machine flow shop problem in which the n unit execution time (UET) jobs are related by an intree precedence constraints, so as to minimize simultaneously the overall completion time and the mean completion time.

For further research, an immediate question to look at would be first the generalization of the above algorithm so as to handle the m-machine case with respect to the Makespan and the mean finish time. The ideal solution would be the one which minimizes both criteria simultaneously. Then, secondly, one should look at the same problem in which the jobs are related by a outtree precedence graph. These two problems are under study and we believe strongly that they are solvable in polynomial time, at least for the mono-criterion case.

As far as the complexity aspect is concerned, it would be interesting to know the minimal precedence graph for which the above two problems become $\mathcal{NP}$-hard.

References

[1] Blazewicz, J. *et al.* (1994): *Scheduling algorithms in computer and manufacturing systems*, Springer Verlag.

[2] Johnson, S.M. (1954): Optimal two–and–three–stage production schedules with setup times included, *Naval Res.Logist.Quart.*, **1**, pp. 61–68.

[3] Lenstra, J.K. *et al.* (1977): Complexity of machine scheduling problems, *Annals of Discrete Mathematics*, **1**, pp. 343-362.

[4] Lawler, E.L. *et al.* (1993): Sequencing and scheduling: Algorithms and complexity, in Grave *et al.* (eds): *Handbook in Operation Research and Management science, Vol 4: Logistics of Production and Inventory*, North-Holland, Amsterdam.

[5] Du, J. & Leung, J.Y.T. (1993): Minimizing mean flow time in two machine open shops and flow shops, *J.Algorithms*, **14**, pp. 24-44.

[6] Rinnooy kan, A.H.G. (1976): *Machine scheduling problems: classification, complexity and computations*, Martinus Nijhoff, The Hague.

[7] Sidney, J.B. (1979): The two machines maximum flow time with series-parallel precedence constraints, *Ops. Res.*,**27**, pp. 782-791.