



ELSEVIER

Discrete Applied Mathematics 59 (1995) 11–21

**DISCRETE
APPLIED
MATHEMATICS**

A polynomial algorithm for an open shop problem with unit processing times and tree constraints

H. Bräsel, D. Kluge, F. Werner*

Fakultät für Mathematik, Technische Universität "Otto von Guericke", PSF 4120, 39016 Magdeburg, Germany

Received 19 June 1991; revised 10 June 1993

Abstract

In this paper we consider the open shop problem with unit processing times and tree constraints (outtree) between the jobs. The complexity of this problem was open. We present a polynomial algorithm which decomposes the problem into subproblems by means of the occurrence of unavoidable idle time. Each subproblem can be solved on the base of an algorithm for the corresponding open shop problem without tree constraints.

Keywords: Scheduling; Open shop problems; Polynomial algorithm

1. Introduction

In this paper we consider the $[O/p_{ij} = 1, \text{outtree}/\sum C_i]$ open shop problem (problem P^T): n jobs J_i with $i \in I = \{1, \dots, n\}$ have to be processed on m machines M_j with $j \in J = \{1, \dots, m\}$. We have the condition $p_{ij} = 1$ for the processing times of each operation (i, j) which represents the processing of J_i on M_j . The machine and job orders can be chosen arbitrarily (open shop) but we have to consider precedence constraints between the jobs. If job J_i is a predecessor of J_k , then the last operation of J_i must be performed before the processing of the first operation of J_k starts. Here the outtree problem is considered, i.e. each job has at most one direct predecessor in the corresponding graph of precedence constraints. Now we look for a feasible combination of the machine and job orders which minimizes the function $f = \sum C_i$, where C_i denotes the completion time of J_i .

In the case of arbitrary processing times, the problem $[O2/\text{tree}/\sum C_i]$ is already NP-hard [6]. For unit time open shop problems, there exist several polynomial time algorithms. Gonzales [5] and Tanaev et al. [10] give polynomial algorithms for the problems $[O/p_{ij} = 1/C_{\max}]$ and $[O/p_{ij} = 1/\sum C_i]$. Adiri and Amit [1] and Tanaev

*Corresponding author.

et al. [10] modified the algorithm such that optimal solutions for both criteria C_{\max} and $\sum C_i$ are obtained. Lawler et al. (cf. [7]) and Tanaev et al. [10] solved the $[O/p_{ij} = 1/\sum w_i C_i]$ problem in polynomial time. In [8] Liu and Bulfin developed polynomial algorithms for the problems $[O/p_{ij} = 1/\sum T_i]$ and $[O/p_{ij} = 1/\sum U_i]$. Lawler et al. (cf. [7]) considered the case of release dates $r_i \geq 0$ and gave polynomial algorithms for the problems $[O3/p_{ij} = 1, r_i \geq 0/C_{\max}]$ and $[O2/p_{ij} = 1, r_i \geq 0/L_{\max}]$. Bräsel [2] showed that the $[O/p_{ij} = 1, r_i \geq 0/C_{\max}]$ problem can be solved in polynomial time.

When this paper was submitted, minimal open problems within the class of unit time open shop problems were $[O2/p_{ij} = 1, \text{outtree}/\sum C_i]$ and $[O2/p_{ij} = 1, \text{tree}/C_{\max}]$. In this paper and in [3] we give polynomial time algorithms for both problems with an arbitrary number of machines. Recently, Brucker et al. [4] have solved a class of unit time open shop problems by reductions to special preemptive scheduling problems on m identical parallel machines. However, we will give an algorithm for $[O/p_{ij} = 1, \text{outtree}/\sum C_i]$ with a lower complexity than in the case of using the ideas in [4].

Results for problems with $p_{ij} = 1$ are also interesting because such investigations lead to structural assertions for the corresponding problems with arbitrary processing times. In fact the more general problems with arbitrary processing times are NP-hard in most cases. Therefore, such structural investigations can be used for the development of effective heuristics.

In Section 2 we introduce some necessary notations which are useful for further considerations. Section 3 contains some properties of the corresponding problem without tree constraints (problem P) which are necessary for solving P^T . In Section 4 we prove that P^T can be solved by decomposing the problem into different subproblems. We describe the algorithm in detail.

2. Mathematical model

For the open shop problem a schedule S may be represented by a graph $G^S = (V, E)$. V denotes the set of operations $(i, j) \in I \times J$. The set E contains horizontal and vertical arcs which describe the machine and job orders. S is feasible iff G^S does not contain any cycle.

An example of such a graph is shown in Fig. 1. In this example we have the machine order $M_3 \rightarrow M_2 \rightarrow M_4 \rightarrow M_1$ for job J_2 , and on machine M_3 we have the job order $J_2 \rightarrow J_1 \rightarrow J_3$.

If we have $p_{ij} = 1$ for all operations $(i, j) \in I \times J$, the following Gantt chart (see Fig. 2) describes the schedule given in Fig. 1. Note that another possibility to represent a schedule is given by the matrix of the completion times of all operations. We now consider the problem P^T . $G^T = (I, E^T)$ denotes the given graph of precedence constraints between the jobs. The edge set E^T represents the tree structure (outtree), i.e. each job J_i has at most one direct predecessor in G^T . The graph G^T imposes restrictions to the set of feasible schedules. To illustrate, let G^T be as shown in Fig. 3(a)

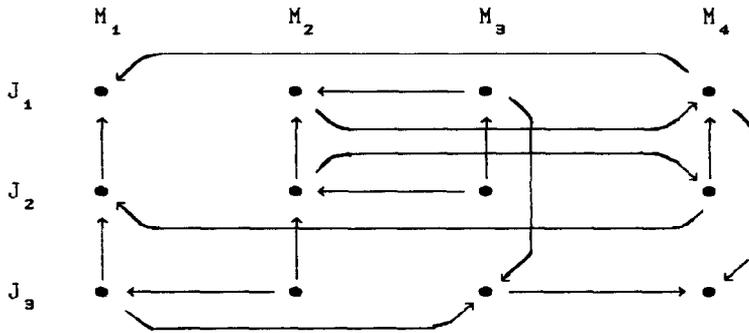


Fig. 1. Graph $G^S = (V, E)$.

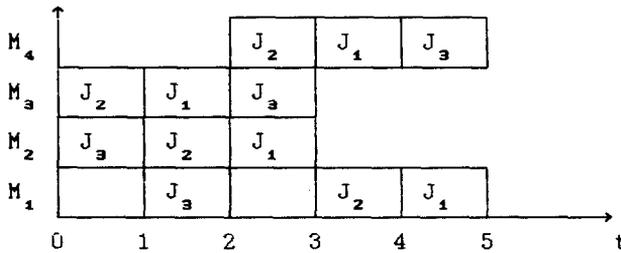


Fig. 2.

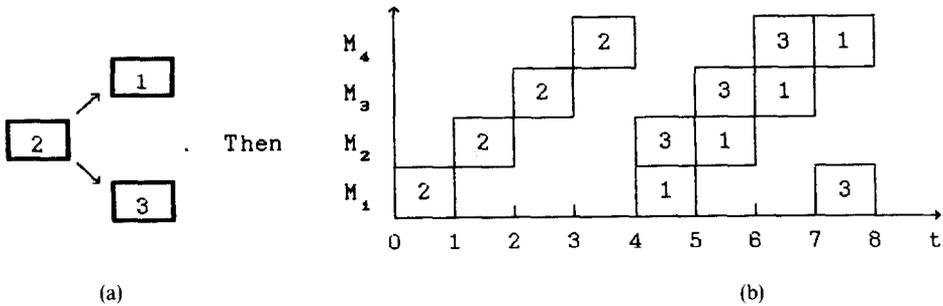


Fig. 3.

and let $m = 4$. Then Fig. 3(b) represents a feasible schedule. To avoid unnecessary complications, we will identify the jobs as in G^T only by their indices shown in the above Gantt chart.

3. Some properties of the problem P

This problem can be solved in $O(nm)$ time. To explain the structure of an optimal schedule for this problem, we make the following considerations. Consider the parallel

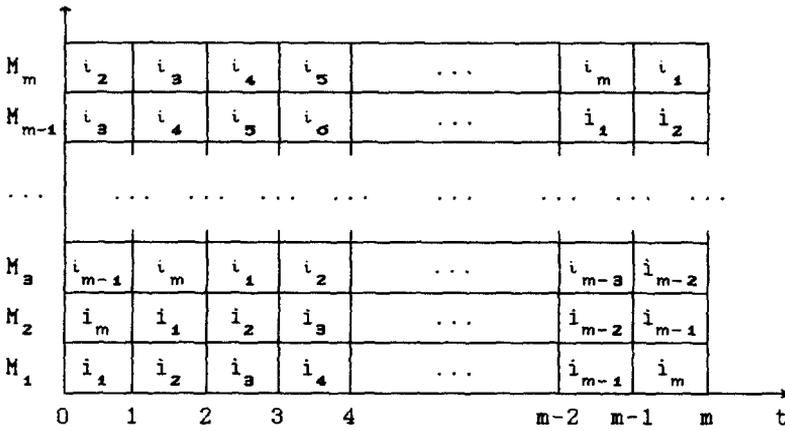


Fig. 4.

machine problem $[Pm/p_i = m, pmtn/\sum C_i]$ which is a relaxation of the problem $[Om/p_i = 1/\sum C_i]$. In [9] McNaughton has proved that there is no schedule for the parallel machine problem with a finite number of preemptions which yields a smaller objective value than an optimal nonpreemptive schedule. Hence, the optimal objective value for $[Pm/p_i = m, pmtn/\sum C_i]$ is a lower bound for the problem $[Om/p_i = 1/\sum C_i]$.

Let $n = g \cdot m + h$ with $g \geq 0$ and $0 \leq h < m$. Due to the above considerations, we obtain an optimal schedule for the problem P if we build g blocks of m jobs such that these jobs of each block completely processed within m time units, i.e. the jobs of the k th block are processed in the time period $[(k - 1) \cdot m, k \cdot m]$ for $k = 1, \dots, g$. In the case of $0 < h < m$ the remaining jobs are also completely processed within m time units.

A set $B = \{i_1, i_2, \dots, i_m\}$ of m jobs of a block can be scheduled within m time units as shown in Fig. 4. If we have $0 < h < m$, the jobs can be scheduled in a block within m time units in the same way by deleting the last jobs in the above Gantt chart. Thus, the jobs of a set $B^* = \{i_1, i_2, \dots, i_h\}$ with $h < m$ can be scheduled as shown in Fig. 5.

The insertion of the jobs into the blocks can be done arbitrarily, for instance according to the job numbers. We note that a schedule with the described structure has an objective value which is equal to the lower bound given by the objective function value of the problem $[Pm/p_i = m, pmtn/\sum C_i]$.

Example 1. Let $n = 11$ and $m = 4$. Therefore, we have $g = 2$ and $h = 3$. According to the above explanation the jobs of $B_1 = \{1, 2, 3, 4\}$ are processed in the period $[0, 4]$, the jobs of $B_2 = \{5, 6, 7, 8\}$ are processed in the period $[4, 8]$ and the remaining jobs of $B_3 = \{9, 10, 11\}$ are processed in the period $[8, 12]$. Thus, we obtain Fig. 6. The objective value is 84.

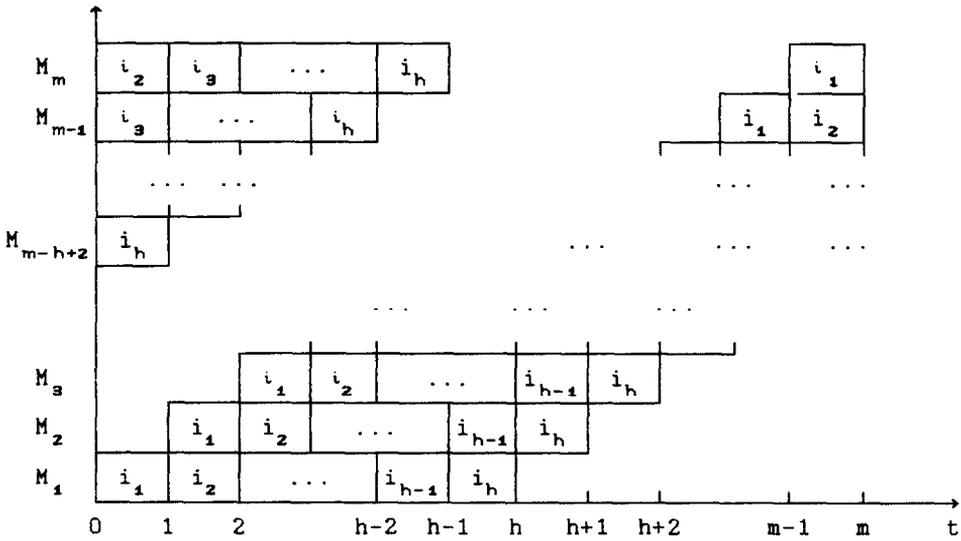


Fig. 5.

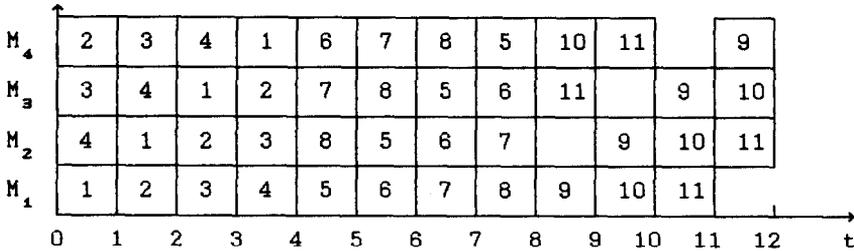


Fig. 6.

We will show in the next section that an optimal solution of the outtree problem decomposes into subschedules of the type given in Example 1.

4. A polynomial algorithm for problem P^T

In this section we develop a polynomial algorithm for problem P^T . This is done by decomposition into several subproblems which can be separately solved using the block scheduling algorithm given in Section 3 and a second algorithm presented in this section. In each subproblem we form again blocks of jobs such that the processing of each block requires exactly m time units. Before describing the algorithm in detail, we introduce some notations. Let $rk(i)$ be the rank of vertex (job) i in G^T . The set S_k contains vertices with the same rank, i.e.

$$S_k = \{i \in I \mid rk(i) = k\}, \quad k = 1, \dots, k'$$

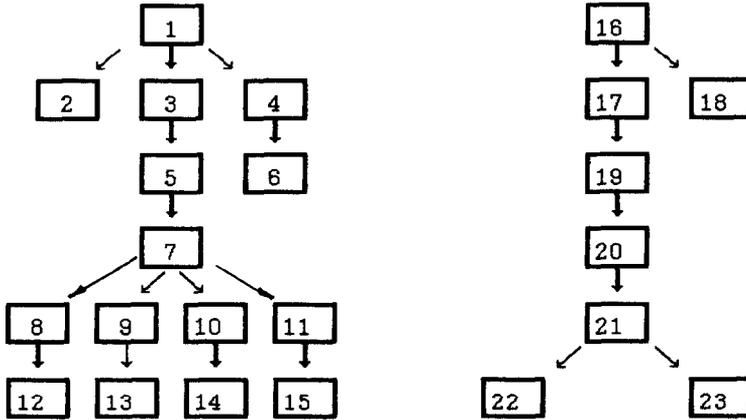


Fig. 7.

where

$$k' = \max \{ \text{rk}(i) \mid i \in I \}.$$

Furthermore, let $S_k = \emptyset$ for $k > k'$. Moreover we use $|S_k| = s_k$. The following example serves as illustration.

Example 2. Let $m = 4$ and the precedence constraints be as in Fig. 7. We obtain:

$$S_1 = \{1, 16\}, \quad s_1 = 2;$$

$$S_2 = \{2, 3, 4, 17, 18\}, \quad s_2 = 5;$$

$$S_3 = \{5, 6, 19\}, \quad s_3 = 3;$$

$$S_4 = \{7, 20\}, \quad s_4 = 2;$$

$$S_5 = \{8, 9, 10, 11, 21\}, \quad s_5 = 5;$$

$$S_6 = \{12, 13, 14, 15, 22, 23\}, \quad s_6 = 6;$$

$$S_k = \emptyset \text{ for } k > 6.$$

Now we divide the job set I into disjoint subsets I_μ ($\mu = 1, \dots, \mu^*$) and we will show that this decomposition fulfils the following conditions:

- the processing of each job of I_μ ($\mu \geq 2$) can only begin if all jobs of $I_{\mu-1}$ have been completed and
- the processing of all jobs of I_μ ($1 \leq \mu \leq \mu^*$) without idle times is not possible.

The following lemma answers the question for the first occurrence of unavoidable idle times.

Lemma 1. *If there exists a k^* with*

$$k^* = \min \left\{ k \mid \sum_{j=1}^k s_j < m \cdot k \right\}, \tag{2}$$

then we have unavoidable idle times before time $k^ \cdot m$.*

Proof. All jobs of S_j with $j \geq k^* + 1$ have an earliest starting time of at least $k^* \cdot m$. On the other hand all m machines can perform exactly $k^* \cdot m^2$ operations up to $k^* \cdot m$. Because of (2) we have less than $m \cdot k^*$ jobs in the time period being considered (i.e. less than $k^* \cdot m^2$ operations). Hence idle times of the machines are unavoidable. \square

If there does not exist such a k^* with (2), we do not have unavoidable idle times resulting from the precedence constraints. In this case we set $k^* = \lceil |I|/m = n/m$. If k^* has been obtained, then we determine the next subset with respect to the set $I \setminus I_1$ and so on.

Considering the tree constraints of Example 2 again, we obtain

$$I_1 = \{1, 16\} = S_1,$$

$$I_2 = \{2, 3, 4, 5, 6, 7, 17, 18, 19, 20\} = S_2 \cup S_3 \cup S_4,$$

$$I_3 = \{8, 9, 10, 11, 12, 13, 14, 15, 21, 22, 23\} = S_5 \cup S_6.$$

Now the first subproblem P_1^T with the job set $I_1 = \bigcup_{k=1}^{k^*} S_k$ will be solved. Due to the last section, it is sufficient to determine the job sets B_k for each block k , i.e. the jobs which are completely scheduled in the period $[(k - 1) \cdot m, k \cdot m]$. In the following, L_k denotes the set of jobs which can be inserted into B_k (i.e. all predecessors of job i with $i \in L_k$ have already been processed which means they have been inserted into previous blocks). Let $z_1(i)$ be the number of successors of vertex i with respect to I_1 and $z_2(i)$ denotes the number of vertices on a longest path to one of the sinks with respect to I_1 . To put m jobs from L_k into B_k for $k < k^*$, we propose two variants which both generate optimal schedules:

(V₁) Select m jobs from L_k with the greatest z_1 values!

(V₂) Select m jobs from L_k with the greatest z_2 values!

Then Algorithm 1 is as follows:

Algorithm 1. Solution of the first subproblem with the job set I_1 , {Input: $|I_1|$, m , k^* , tree constraints, S_1, \dots, S_{k^*} , for all $i \in I_1$, $z_1(i)$ or $z_2(i)$ with respect to the set I_1 in ordered form}:

begin $L_1 := S_1$; $k := 1$;
while $k < k^*$ **do**

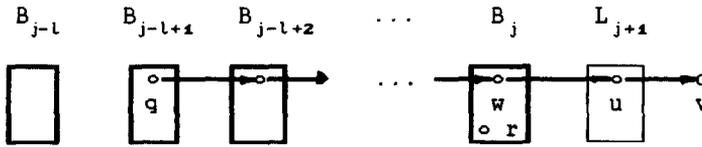


Fig. 8.

begin determine the job set B_k of block k by selecting m jobs from L_k according to criterion (V_1) or (V_2) ; $k := k + 1$;
 determine L_k by replacing in L_{k-1} all jobs of B_{k-1} by their direct successors;
end;
 $B_{k^*} := L_{k^*}$;
end.

In connection with the above algorithm, we still have to prove two assertions:

- (1) for each $k < k^*$ the set L_k contains at least m jobs and
- (2) all jobs not contained in $\bigcup_{k=1}^{k^*-1} B_k$ can be inserted into the set B_{k^*} .

This is done by the following two theorems at first for rule (V_1) .

Theorem 1. *Let $k^* > 1$. Then we can insert m jobs into each block $k = 1, \dots, k^* - 1$, i.e. we have $|L_k| \geq m$ for each k .*

Proof. The proof is done by induction. For $k = 1$ we obtain by considering the definition of k^* , $|L_1| = s_1 \geq m$. Assume that the theorem holds for $k = 1, \dots, j$, $j \leq k^* - 2$. Then we consider the following cases for $k = j + 1$:

(a) All jobs of B_j have at least one successor in G^T . Because of the outtree constraints, we obtain $|L_{j+1}| \geq |L_j|$ and this yields the assertion.

(b) B_j contains a job r without successor in G^T . Assume that $|L_{j+1}| < m$. Because of $\sum_{i=1}^{j+1} s_i \geq m(j+1)$, the set L_{j+1} contains at least one job u with a successor v from $\bigcup_{i=1}^{j+1} S_i$ in G^T . Moreover, B_j must contain a direct predecessor w of u otherwise this would be a contradiction to the insertion of r into B_j . Let $q \in B_{j-l+1}$ be a predecessor of u such that any two adjacent jobs in the chain from q to u have been inserted into adjacent blocks and B_{j-l} does not contain any predecessor of q (see Fig. 8).

Clearly, we have $j - l + 1 \geq 2$. Moreover, each job $i \in B_{j-l}$ has at least $l + 1$ successors because q was not taken into B_{j-l} by Algorithm 1. Let G^* be the graph obtained from G if all jobs from $\bigcup_{i=1}^{j-l} B_i$ and the arcs, which are adjacent with these vertices, have been deleted. Moreover, let s_i^* be the number of vertices with rank i in G^* . Because each of the sets $B_{j-l+1}, B_{j-l+2}, \dots, B_j$ contains m jobs in each case, we have $\sum_{i=1}^h s_i^* \geq h \cdot m$ for $h = 1, \dots, l - 1$.

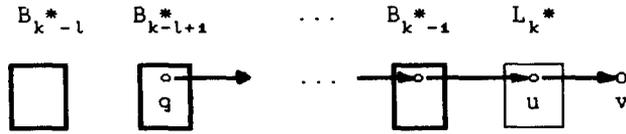


Fig. 9.

Because of the outtree constraints, $\sum_{i=1}^l s_i^* \geq (l + 1) \cdot (m + 1) > l \cdot m$ with $l \leq j$ holds, since q and all successors of job q and of each job i with $i \in B_{j-l}$ belong to the vertex set of G^* . Hence, we have $|L_{j+1}| \geq m$ which contradicts the assumption. \square

Theorem 2. All jobs which are not contained in $\bigcup_{k=1}^{k^*-1} B_k$ can be inserted into B_{k^*} .

Proof. Assume that there exists a precedence constraint between two jobs u and v not contained in $\bigcup_{k=1}^{k^*-1} B_k$. Let $q \in B_{k^*-l+1}$ be a predecessor of u such that any two adjacent jobs within the chain from q to u have been inserted into adjacent blocks and B_{k^*-l} does not contain any predecessor of q (see Fig. 9, possibly $q = u$).

Again, each job $i \in B_{k^*-l}$ has at least l successors because q was not taken into B_{k^*-l} by Algorithm 1. By considering Theorem 1 and the outtree structure of the precedence constraints, this yields a contradiction to the definition of k^* because the considered subproblem would contain more than $k^* \cdot m$ jobs. \square

Now we consider selection rule (V_2) . It is immediately clear that Theorem 1 also holds in this case because, if a vertex i has $z_2(i)$ vertices on a longest path to one of the sinks, this vertex has at least $z_2(i)$ successors. In the following, we call a job i critical at time $t = m \cdot j$ iff $k^* - j = z_2(i)$.

Theorem 3. Applying (V_2) , there exist at each time $t = m \cdot j$ with $j = 0, \dots, k^* - 1$ at most m critical jobs that have not yet been processed.

Proof. Clearly we have $s_1 \geq m$. We prove that L_1 contains at most m critical jobs. Assume we have $m + d$ critical jobs at time 0 with $d \geq 1$. Thus, I_1 would contain at least $(m + d) \cdot k^* > m \cdot k^*$ jobs which contradicts the definition of k^* . Therefore, we can put all critical jobs at time 0 into B_1 .

Assume that the theorem holds for $k = 0, \dots, j - 1 \leq k^* - 2$, i.e. for $k \leq j - 1$ all critical jobs at time $m \cdot k$ have been taken into B_{k+1} in each case, and we show that this is also true for $k = j$. Assume again that we have $m + d$ critical jobs at time $m \cdot j$ with $d \geq 1$. Each of these jobs must have at least $k^* - j - 1$ successors in I_1 .

Because $|B_1| = \dots = |B_j| = m$ holds, we obtain

$$|I_1| \geq m \cdot j + (m + d)(k^* - j) = m \cdot k^* + d \cdot (k^* - j) > m \cdot k^*.$$

This contradicts the definition of k^* again. Hence, we have at most m critical jobs which are all taken into B_{j+1} according to (V_2) . \square

Obviously Theorem 2 also holds if rule (V_2) is applied because all critical jobs at time $m \cdot j$ have been inserted into B_j .

Especially, it is clear now that the optimal objective value for problem P_1^T is equal to the optimal objective value for the problem P_1 when all precedence constraints are ignored.

Now we assume that we have determined the subsets I_1, \dots, I_μ and each of these problems has optimally been solved by Algorithm 1. Moreover, let k_j^* be the maximal rank of a job of the set I_j . Clearly, no job of I_j can begin processing before time $k_{j-1}^* \cdot m$. On the other hand, by our decomposition algorithm, the processing of the jobs of I_v ($v < j$) requires exactly $(k_v^* - k_{v-1}^*) \cdot m$ time units where $k_0^* = 0$. Hence, in the proposed algorithm the jobs of I_j will begin processing at time $k_{j-1}^* \cdot m$, with all m machines available. Therefore, by concatenating the optimal schedules for each subproblem, we obtain an optimal schedule for the overall problem.

Considering the tree constraints of Example 2 again, there exist 3 subproblems with the job sets I_1, I_2 and I_3 . In the case of rule (V_1) , our algorithm yields a schedule with the blocks $B_1 = \{1, 16\}$, $B_2 = \{2, 3, 4, 17\}$, $B_3 = \{5, 6, 18, 19\}$, $B_4 = \{7, 20\}$, $B_5 = \{8, 9, 10, 21\}$, $B_6 = \{11, 12, 13, 22\}$ and $B_7 = \{14, 15, 23\}$. Applying (V_2) , we obtain a schedule with the blocks $B_1 = \{1, 16\}$, $B_2 = \{2, 3, 4, 17\}$, $B_3 = \{5, 6, 18, 19\}$, $B_4 = \{7, 20\}$, $B_5 = \{8, 9, 10, 11\}$, $B_6 = \{12, 13, 14, 21\}$ and $B_7 = \{15, 22, 23\}$. The objective value is 380.

5. Concluding remarks

We presented a polynomial algorithm for the problem P^T under the assumption that the tree constraints form an outtree. As for problem P , the presented algorithm requires also $O(nm)$ time, when the tree constraints are given in the input form of Algorithm 1.

As already mentioned, problem P^T can also be solved by using the relations between parallel machine and open shop problems. However, the machine assignment procedure that transforms an optimal preemptive schedule for the parallel machine problem into an open shop schedule requires $O(n^2m)$ time or in a more sophisticated way $O(nm(\log(nm))^2)$ (cf. [4]). Thus, our algorithm which directly constructs a schedule for the open shop problem has a lower complexity.

In the case of an intree (i.e. each vertex has at most one direct successor in G^T), we only mention that for $m = 2$ a similar algorithm can be stated. For a machine number $m > 2$, we give a simple example that both criteria of the algorithm do not necessarily yield an optimal solution (see Fig. 10).

Let $m = 3$ and the intree constraints be as in Fig. 10. In the first step, both criteria can select jobs from the set $\{1, 2, 3, 4, 5\}$. Assume that the jobs 2, 3 and 4 have been

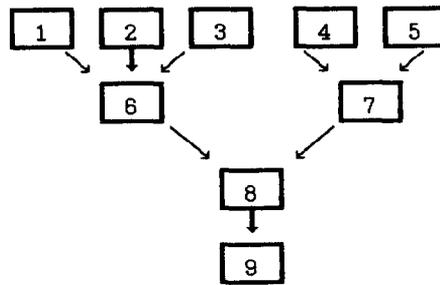


Fig. 10.

taken into B_1 . Then we can put only jobs 1 and 5 into B_2 , jobs 6 and 7 into B_3 , job 8 into B_4 and job 9 into B_5 . However, this solution is not optimal. The reader can easily verify that an optimal solution can be obtained by the following job sets: $B_1 = \{1, 2, 3\}$, $B_2 = \{4, 5, 6\}$, $B_3 = \{7\}$, $B_4 = \{8\}$ and $B_5 = \{9\}$. Thus, the intree problem is still open.

References

- [1] I. Adiri and N. Amit, Openshop and flowshop scheduling to minimize sum of completion times, *Comput. Oper. Res.* 11 (1984) 275–284.
- [2] H. Bräsel, Lateinische Rechtecke und Maschinenbelegung, Dissertation B, TU Magdeburg (1990).
- [3] H. Bräsel, D. Kluge and F. Werner, A polynomial algorithm for the $[n/m/O, t_{ij} = 1, \text{tree}/C_{\max}]$ problem *European J. Oper. Res.* 72 (1994) 125–134.
- [4] P. Brucker, B. Jurisch and M. Jurisch, Open shop problems with unit time operations, *Z. Oper. Res.* 37 (1993) 339–354.
- [5] T. Gonzales, Unit execution time shop problems, *Math. Oper. Res.* 7 (1982) 57–66.
- [6] B.J. Lageweg, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Computer aided complexity classification of deterministic scheduling problems, Report BW 138/81, Department of Operations Research (1981).
- [7] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, Sequencing and scheduling: algorithms and complexity, Report BS-R8909, Department of Operations Research, Statistics and System Theory (1989).
- [8] C.Y. Liu and R.L. Bulfin, Scheduling open shops with unit execution times to minimize functions of due dates, *Oper. Res.* 36 (1988) 553–559.
- [9] R. McNaughton, Scheduling with deadlines and loss functions, *Management Sci.* 6 (1959) 1–2.
- [10] V.S. Tanaev, Y.N. Sotskov and V.A. Strusevich, *Theory of Scheduling – Multistage Systems* (Nauka, Moscow, 1989) (in Russian).