HEIDEMARIE BRÄSEL
DAGMAR KLUGE
FRANK WERNER

# Polynomial time algorithms for special open shop problems with precedence constraints and unit processing times

*RAIRO. Recherche opérationnelle*, tome 30, nº 1 (1996), p. 65-79
<http://www.numdam.org/item?id=RO_1996__30_1_65_0>

Recherche opérationnelle/Operations Research
(vol. 30, n° 1, 1996, pp. 65-79)

# POLYNOMIAL TIME ALGORITHMS FOR SPECIAL OPEN SHOP PROBLEMS WITH PRECEDENCE CONSTRAINTS AND UNIT PROCESSING TIMES (*) (†)

by Heidemarie BRÄSEL (1), Dagmar KLUGE (1) and Frank WERNER (1)

Abstract. – *In this paper we consider different open shop problems with unit processing times. For the problem with two machines and arbitrary precedence constraints among the jobs, we give a polynomial time algorithm for the minimization of the makespan with a better worst case complexity than a previous algorithm known from the literature if the number of arcs is of linear order. The complexity of the open shop problem with unit processing times and intree constraints among the jobs was open up to now if the sum of completion times of the jobs has to be minimized. By means of the first result we give a polynomial time algorithm for this problem with two machines.*

Keywords: Open shop scheduling, unit processing times, polynomial time algorithms.

Résumé. – *Le but de cet article est de proposer des algorithmes polynomiaux pour deux problèmes d'ordonnancement de type open shop à deux machines avec contraintes de précédence et temps opératoires unitaires. Le premier algorithme qui permet d'optimiser la durée totale a une meilleure complexité que le meilleur algorithme connu de la littérature lorsque le nombre de contraintes est proportionnel au nombre de jobs. Le deuxième algorithme, qui est basé sur le premier, permet de résoudre un problème ouvert qui est celui de la minimisation de la durée moyenne d'achèvement des jobs quand les contraintes de précédences forment une anti-arborescence.*

Mots clés : Ordonnancement, open shop, temps opératoires unitaires, algorithmes polynomiaux.

## 1. INTRODUCTION

In an open shop problem we have $m$ machines 1, 2, ..., $m$, and $n$ jobs 1, 2, ..., $n$. Each job $i$ consists of $m$ operations $O_{ij}$ where $O_{ij}$ has to be performed on machine $j$ for $p_{ij}$ time units without preemption. We assume that each machine can process at most one operation at a time and each job can be processed by at most one machine at a time. Both, the machine and the job orders, can be chosen arbitrarily. In this paper we consider problems where precedence constraints among the jobs are given. Such a

(*) Received April 1994.
(†) Supported by Deutsche Forschungsgemeinschaft, project ScheMA.
(1) Otto-von-Guericke-Universität, Universitätsplatz 2, 39106 Magdeburg, Deutschland.

constraint $i \rightarrow k$ means that the first operation of job $k$ can only start with processig when the last operation of job $i$ has been completed. The problem is to determine a feasible combination of the machine and job orders which minimizes a certain criterion.

We follow the classification scheme $\alpha|\beta|\gamma$ of scheduling problems suggested by Graham *et al.* [7] where $\alpha$ describes the machine environment, $\beta$ gives some job characteristics and additional requirements and $\gamma$ is the optimality criterion.

In the case of arbitrary processing times most of the open shop problems are NP-hard. If the makespan $C_{\max}$ has to be minimized, the 2-machine problem $O\,2||C_{\max}$ can be solved in polynomial time. However, the problem $O\,2|\text{tree}|C_{\max}$ (*i.e.* the precedence constraints form a tree) is already NP-hard.

A variety of polynomial algorithms has been given for the special case of unit processing times indicated by $p_{ij} = 1$. (*cf.* Gonzalez and Sahni [8], Liu and Bulfin [10], Tanaev *et al.* [12], Bräsel [1], Bräsel and Kleinau [2], Tautenhahn [13] and Brucker *et al.* [5] and so on). Complexity results are given in [9]. In [5] and [13] it has been proven, that in order to solve open shop problems with unit processing times, it is sufficient to solve a corresponding preemptive problem on $m$ identical parallel machines where all jobs have the processing time $m$ and preemptions are allowed at integer times. If a schedule for this parallel machine problem has been determined, a machine assignment procedure constructs a schedule for the open shop problem.

Because this assignment procedure has in the case of a fixed number of machines a complexity of $O\,(n^2)$ or in a more refined version, where ideas from edge coloring are used, a complexity of $O\,(n \log n)$, each algorithm for a unit time open shop problem which uses a known algorithm for the preemptive parallel machine problem mentioned above has at least this complexity. A survey is given in [5]. However, by determining an optimal schedule for special unit time open shop problems directly, we can possibly obtain algorithms with a lower complexity. For instance, in [3] and [4] algorithms are given for the problems $O|p_{ij} = 1, \text{tree}|C_{\max}$ and $O|p_{ij} = 1, \text{outtree}|\sum C_i$, which have a complexity of $O\,(nm)$.

In this paper we consider special open shop problems with unit processing times and precedence constraints among the jobs. The paper is organized as follows. In Section 2 we consider the problem $O\,2|p_{ij} = 1, \text{prec}|C_{\max}$. For this problem we derive an algorithm without solving the corresponding parallel machine problem and, consequently, we do not need the above mentioned machine assignment procedure. The complexity of problem $O|p_{ij} = 1, \text{intree}|\sum C_i$ was open up to now. In Section 3 we give a polynomial time algorithm for this problem with two machines. However, the complexity status of the general $O|p_{ij} = 1, \text{intree}|\sum C_i$ problem remains open.

## 2. THE PROBLEM $O2|p_{ij} = 1, \text{prec}|C_{\max}$

We consider the open shop problem with $n$ jobs, 2 machines and unit processing times. Let the graph $G = [I', E']$ of precedence constraints between the jobs be given. The set of vertices $I'$ is the set of jobs and each arc $(i, k) \in E'$ corresponds to a precedence constraint $i \rightarrow k$. We introduce a sink $s$ representing a fictitious job which leads to the graph

$$G^P = [I' \cup \{s\}, E' \cup \{(i, s) : i \text{ is a sink in } G\}] = [I, E].$$

Hence, all jobs are ancestors of the fictitious job $s$ which also consists of 2 unit time operations. Let $C^*_{\max}(I)$ denote the optimal objective function value for the set $I$ of jobs. Then we have

$$C^*_{\max}(I) - 2 = C^*_{\max}(I'). \tag{2.1}$$

We denote by $rk(i)$ and $rk^*(i)$ the ranks of vertex $i$, *i.e.* the number of vertices on a longest path from a source to the vertex $i$ and from the vertex $i$ to the sink $s$, respectively. Let $rkmax := rk(s)$. Now we form sets $S_k$ and $L_k$ that contain the jobs with the same rank, *i.e.*

$$S_k = \{i \in I | rk(i) = k\} \quad \text{and} \quad L_k = \{i \in I | rkmax - rk^*(i) + 1 = k\}$$

for $k = 1, \ldots, rkmax$. These sets have the following properties:

PROPERTY 1: For each $i \in S_k$, $k > 1$, there exists a predecessor in $S_{k-1}$. If $|S_k| = 1$, say $S_k = \{u\}$, then all jobs from $\bigcup_{r=k+1}^{rkmax} S_r$ are descendants of $u$.

PROPERTY 2: For each $i \in L_k$, $k < rkmax$, there exists a successor in $L_{k+1}$. If $|L_k| = 1$, say $L_k = \{v\}$, then all jobs from $\bigcup_{r=1}^{k-1} L_r$ are ancestors of $v$.

Using the introduced sets $S_k$ and $L_k$, we can easily give two feasible schedules of the problem being considered. Obviously, we can schedule the jobs of the sets $S_k$ and $L_k$, respectively, within the time interval $[\underline{t}_k, \bar{t}_k]$ where in the first case

$$\underline{t}_k = \sum_{r=1}^{k-1} \max\{2, |S_r|\} \qquad \text{and} \qquad \bar{t}_k = \underline{t}_k + \max\{2, |S_k|\}$$

and in the second case

$$\underline{t}_k = \sum_{r=1}^{k-1} \max\{2, |L_r|\} \qquad \text{and} \qquad \bar{t}_k = \underline{t}_k + \max\{2, |L_k|\}.$$

is fulfilled.

Here we use that a set $\{i_1, i_2, \ldots i_\mu\}$ of jobs can be processed in $[\underline{t}_k, \bar{t}_k]$ as follows:

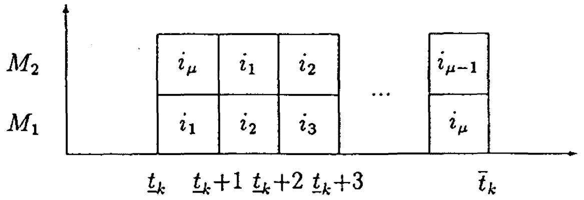

**Figure 1. – Gantt chart in the case $\mu > 1$.**

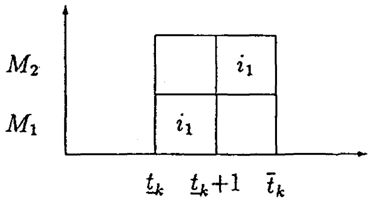

**Figure 2. – Gantt chart in the case $\mu = 1$.**

For the schedules described above we obtain

$$C_{\max}^1 = \sum_{r=1}^{rkmax} \max\{2, |S_r|\} = n + h_1$$

where $h_1$ is equal to the number of sets $S_k$, $1 \le k \le rkmax$, with $|S_k| = 1$.

$$C_{\max}^2 = \sum_{r=1}^{rkmax} \max\{2, |L_r|\} = n + h_2$$

where $h_2$ is equal to the number of sets $L_k$, $1 \leq k \leq rkmax$ with $|L_k| = 1$. In an optimal schedule, we must have a minimal number $h$ of unavoidable idle times on each machine, *i.e.* $C_{\max} = n + h$. Clearly, the first constructed schedule is optimal if there does not exist a set $S_k$, $1 \leq k \leq rkmax - 1$, with $|S_k| = 1$ and the second schedule is optimal if these conditions hold for the sets $L_k$.

The following lemma gives the possibility to describe the existence of unavoidable idle times.

LEMMA 1: *Assume that there exists an index k with* $1 \leq k \leq rkmax$ *such that* $|S_k| = 1$ *holds and that* $S_k = \{i\}$. *Moreover, for the optimal objective value* $C^*_{\max}(PC_i)$ *for the set* $PC_i$ *of ancestors of job* $i$ *we have*

$$C^*_{\max}(PC_i) = \sum_{r=1}^{k-1} |S_r|,$$

*then it is not possible to process all jobs of the sets* $S_1, S_2, \ldots, S_k$ *without any idle time.*

*Proof:* Assume that there exists an index $k$ with the above properties. Then, within the time interval $[0, C^*_{\max}(PC_i)]$ it is possible to process completely all jobs of the set $\bigcup_{r=1}^{k-1} S_r$. We notice that $C^*_{\max}(PC_i)$ is the earliest starting time of job $i$. On the other hand all jobs of the sets $S_u$, $u > k$, are descendants of job $i$. Therefore, within the time interval $[C^*_{\max}(PC_i), C^*_{\max}(PC_i) + 2]$, only job $i$ can be processed, *i.e.* an idle time is unavoidable. ■

The above condition is easy to formulate but hard to handle because the determination of $C^*_{\max}(PC_i)$ is an optimization problem, too. Now we will divide the problem into subproblems by means of the existence of unavoidable idles times.

LEMMA 2: *Let k be the smallest integer with* $|L_k| = 1$ *and assume that* $L_k = \{i\}$. *Moreover, let*

$$S = \bigcup_{r=1}^{k} S_r \setminus \bigcup_{r=1}^{k} L_r.$$

*If* $S = \emptyset$, *then in the time period* $\left[0, \sum_{r=1}^{k-1} |L_r| + 2\right]$ *an idle time is unavoidable and, if* $S \neq \emptyset$, *it is avoidable in this time period.*

*Proof:* Again let $PC_i$ denote the set of all ancestors of job $i \in L_k$. Because $|L_k| = 1$, we obtain $PC_i = \bigcup_{r=1}^{k-1} L_r$ (*cf.* Property 2). Since $|L_r| \geq 2$ for $1 \leq r < k$ the optimal objective value for the set $PC_i$ of jobs is equal to the cardinality of this set. Moreover, $|PC_i|$ is the earliest starting time for job $i \in L_k$.

If $S = \emptyset$ then, due to $\bigcup_{r=1}^{k} L_k \subseteq \bigcup_{r=1}^{k} S_k$, we obtain $\bigcup_{r=1}^{k} L_k = \bigcup_{r=1}^{k} S_k$ and, therefore, $|S_k| = 1$ holds. Thus, $C^*_{\max}(PC_i) = \left| \bigcup_{r=1}^{k-1} L_r \right| = \left| \bigcup_{r=1}^{k-1} S_r \right|$ is satisfied and, by Lemma 1, an idle time is unavoidable in $[0, |PC_i| + 2]$.

If $S \neq \emptyset$, then it contains a job $j \neq i$ executable in

$$\left[ \left| \bigcup_{r=1}^{k-1} L_r \right|, \left| \bigcup_{r=1}^{k-1} L_r + 2 \right| \right]$$

and this idle time is avoidable. ■

By means of Lemma 2 we can design an algorithm for solving the problem as a partition into $\mu$ subproblems $P_r$, $1 \leq r \leq \mu$. We denote the set of jobs of problem $P_r$ by $I_r$. Each set $I_r$ has one of the following properties:

1. there exists a job $i \in I_r$ with the property that all other jobs of $I_r$ are ancestors of job $i$,

2. there exist two jobs $i, j \in I_r$ with the property that job $j$ is not an ancestors of job $i$ and all other jobs of $I_r$ are ancestors of job $i$.

The first case stands for the occurence of an unavoidable idle time when the processing of the job $i \in I_r$ begins and in the second case an idle time is avoidable in this situation by scheduling job $i$ and $j$ in parallel. We obtain $C^*_{\max}(I) = \sum_{r=1}^{\mu} |I_r| = n + h$ and $h$ is the number of unavoidable idle times.

In the following Algorithm 1 we construct a schedule by inserting each job $i$ into a block $B_k$ of jobs, $1 \leq k \leq rkmax$, such that the resulting schedule contains only unavoidable idle times. All jobs in $B_k$ are processed in the time interval $[\underline{t}_k, \bar{t}_k]$ with

$$\underline{t}_k = \sum_{r=1}^{k-1} \max\{2, |B_r|\} \qquad \text{and} \qquad \bar{t}_k = \underline{t}_k + \max\{2, |B_k|\} \qquad (2.2)$$

Let $\underline{k}(i)$ and $\bar{k}(i)$ be the smallest and greatest possible value of the index $k$ of the block in which job $i$ can be inserted, *i.e.* we have $i \in S_{\underline{k}(i)}$ and $i \in L_{\bar{k}(i)}$. We call a job $i$ critical if $\underline{k}(i) = \bar{k}(i)$.

In Algorithm 1 we first insert all critical jobs into the blocks. If after this insertion there exists an index $k$ with $|B_k| = 1$ we determine exactly one job $i$ in the set of unscheduled jobs which can be inserted in $B_k$. If this set of possible jobs is not empty, we choose the job $i$ with minimal $\bar{k}(i)$. This can be realized in $O(n \max\{1, \max\{\bar{k}(i) - \underline{k}(i) | 1 \leq i \leq n\}\})$ if we assume that the jobs are ordered by nondecreasing $\bar{k}(i)$ values, *i.e.* $\bar{k}(1) \leq \bar{k}(2) \leq \ldots \leq \bar{k}(n)$. Furthermore, if there exist two jobs $i$ and $i+1$ with $\bar{k}(i) = \bar{k}(i+1)$ then $\underline{k}(i) \leq \underline{k}(i+1)$ should be satisfied. This ordering can be done in $O(n + r)$ time if $r$ is the number of arcs in the graph of precedence constraints (note that this ordering is simply a classical topological ordering of the vertices of a directed graph without circuits). Hence we get the following complexity of Algorithm 1:

THEOREM 1: *Algorithm 1 solves the problems* $O2|p_{ij} = 1, \text{prec}|C_{\max}$ *in* $O(n \max\{1, \max\{\bar{k}(i) - \underline{k}(i) | 1 \leq i \leq n\}\} + r)$ *time.*

Algorithm 1: Determination of the blocks $B_k$ for the problem $O2|p_{ij} = 1, \text{prec}|C_{\max}$

```
 begin
1.   for $k := 1$ to $rkmax$ do $B_k = \emptyset$;
2.   for $i := 1$ to $n$ do
3.       if $\underline{k}(i) = \bar{k}(i)$ then
             begin
4.               $k := k(i)$;
5.               $B_k := B_k \cup \{i\}$;
             end,
6.   for $i := 1$ to $n$ do
7.       if $\underline{k}(i) < \bar{k}(i)$ then
             begin
8.               $k := \underline{k}(i)$;
9.               while $(|B_k| \geq 2$ or $k < \bar{k}(i))$ do $k := k + 1$;
10.              $B_k := B_k \cup \{i\}$;
             end;
 end.
```

Now we determine the corresponding schedule by means of the sets $B_k$, $1 \leq k \leq rkmax$ and the corresponding values of $\underline{t}_k$ and $\bar{t}_k$ (which can be calculated by 2.2) as shown in Figure 1 and Figure 2.

Note that Algorithm 1 does not successively determine the individual subproblems. Nevertheless, these subproblems have been obtained when Algorithm 1 stops. Also the number $\mu$ of subproblems has only been determined at the end of Algorithm 1. We still have to prove that Algorithm 1 works correctly.

THEOREM 2: *Algorithm 1 generates an otpimal solution.*

*Proof:* Consider the first block $B_k$ with one of the properties

1. $|B_k| = 1$ or
2. $|B_k| = 2$ and there exists a job $j \in B_k$ with $\bar{k}(j) > k$.

In both cases $|B_r| \geq 2$ for $1 \leq r < k$ holds and the algorithm has only ordered all jobs $l$ with $\bar{k}(l) < k$ into $\bigcup_{r=1}^{k-1} B_r$. Thus, the equality $\bigcup_{r=1}^{k-1} B_r = \bigcup_{r=1}^{k-1} L_r$ is satisfied. Furthermore, the condition $|B_k \cap L_k| = 1$ holds and for each of the above described properties of $B_k$ the equality $|L_k| = 1$ follows. According to lemma 2 there exists an unavoidable idle time in case 1 and we can avoid this idle time in case 2. The set $I_1$ of jobs of the first subproblem $P_1$ is $I_1 = \bigcup_{r=1}^{k} B_r$ and we obtain $C^*_{\max}(I_1) = \sum_{r=1}^{k} \max\{|B_k|, 2\}$. Clearly, for the further considerations we have to delete the job $j \in B_k$ from the set $L_{\bar{k}(j)}$. Now we can separately consider the remaining blocks $B_r$ with $k < r \leq rkmax$. We repeat the above argument, which proves the theorem. ■

To illustrate the above algorithm, we consider the following example. Let $n = 18$ and the graph $G^P$ is given in Figure 3. The jobs are numbered according to the required order.

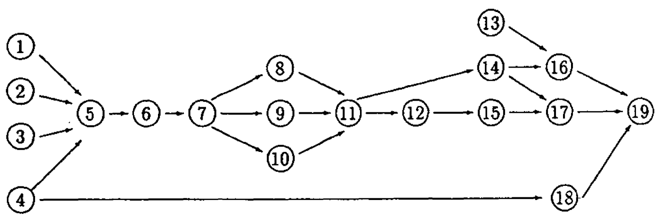

**Figure 3. – The graph $G^P$.**

Notice that vertex 19 is the fictitious sink. All jobs in $I \backslash \{13, 14, 16, 18\}$ are critical. In step 1-5 the algorithm determines the blocks $B_1 = \{1, 2, 3, 4\}$, $B_2 = \{5\}$, $B_3 = \{6\}$, $B_4 = \{7\}$, $B_5 = \{8, 9, 10\}$, $B_6 = \{11\}$, $B_7 = \{12\}$, $B_8 = \{15\}$, $B_9 = \{17\}$ and $B_{10} = \{19\}$. In step 6-10 job 13 is inserted into $B_2$, job 14 into $B_7$, job 16 $B_8$ and finally job 18 is inserted into $B_3$ (note that $\underline{k}(13) = 1$, $\underline{k}(14) = 7$, $\underline{k}(16) = 8$ and $\underline{k}(18) = 2$ hold). Because of the cardinalities of the sets $B_k$, $1 \leq k \leq 10$, all jobs of the set $B_k$ are processed in the time interval $[\underline{t}_k, \bar{t}_k]$ with

| $k$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\underline{t}_k$ | 0 | 4 | 6 | 8 | 10 | 13 | 15 | 17 | 19 | 21 |
| $\bar{t}_k$ | 4 | 6 | 8 | 10 | 13 | 15 | 17 | 19 | 21 | 23 |

Therefore, the optimal objective function values is $C_{\max} = 21$.

## 3. THE PROBLEM $O\,2|p_{ij} = 1, \text{intree}|\sum C_i$

In this section we give a polynomial time algorithm for the problem $O\,2|p_{ij} = 1, \text{intree}|\sum C_i$. $G^P$ denotes again the graph of precedence constraints including the fictitious sink. The above problem will be solved by partitioning the original problem into two subproblems $P_1$ and $P_2$ with the following properties:

• the set $I_1$ of jobs of the first subproblem $P_1$ is the largest subset that can be processed without any idle time on the machines, *i.e.* the last job is completed at time $|I_1|$ and

• the set of jobs $I_2$ of the second problem $P_2$ forms a chain in the graph of precedence constraints.

Notice that it is possible that all jobs form a chain. Let $S_k$, $L_k$ and $rkmax$ be defined as in Section 2. We use the considerations about unavoidable idle times in Section 2 for determining the subproblems. Clearly, Algorithm 1 constructs also an optimal schedule for the problem $O\,2|p_{ij} = 1, \text{intree}|C_{\max}$ such that there exists an index $k$ with $|B_r| \geq 2$ for $1 \leq r < k$ and $|B_r| = 1$ for $k \leq r \leq rkmax$. Therefore, we obtain the above described partition $I = I_1 \cup I_2$ with $I_1 = \bigcup_{r=1}^{k-1} B_r$ and $I_2 = \bigcup_{r=k}^{rkmax} B_r$.

All jobs not contained in $I_1$ form a chain in the graph of precedence constraints and $C^*_{\max}(I_1)$ is the earliest possible starting time of the first job of this chain. Hence, having determined an optimal schedule for the set $I_1$ of jobs with respect to $\sum C_i$ that is also $C_{\max}$ optimal, one can concatenate this optimal schedule with the remaining set $I_2 := I \backslash I_1$ of jobs such that the jobs of $I_2$ are processed in the interval $[C^*_{\max}(I_1), C^*_{\max}(I_1) + 2 \cdot |I_2|]$. Note that one machine is always idle in this period.

Let us now return to the solution of the first subproblem with the set $I_1$ of jobs. This problem will be solved by transforming the intree problem into an outtree problem by reversing the direction of all arcs between jobs of $I_1$ and applying an algorithm, which has been given in [4] for the problem $O|p_{ij} = 1, \text{outtree}|\sum C_i$.

First we derive a lower bound for $O2|p_{ij} = 1, \text{outtree}|\sum C_i$. Consider the parallel preemptive machine problem $P2|p_i = 2, pmtn|\sum C_i$, which is a relaxation of the problem $O2|p_{ij} = 1|\sum C_i$. In [11] 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. Hence, the optimal objective value for $P2|p_i = 2, pmtn|\sum C_i$ is a lower bound for $O2|p_{ij} = 1|\sum C_i$. Thus, an optimal schedule for this open shop problem is given by scheduling successively blocks of two jobs within 2 time units (only the last block contains only one job if $n_1 := |I_1|$ is odd).

The corresponding objective value is

$$f_1 = 2 \cdot 2 + 2 \cdot 4 + \ldots + 2 \cdot n_1 = 4 \cdot \binom{\frac{n_1}{2} + 1}{2}$$

if $n_1$ is even or

$$f_2 = 2 \cdot 2 + 2 \cdot 4 + \ldots + 2 \cdot (n_1 - 1) + (n_1 + 1) = 4 \cdot \binom{\frac{n_1 + 1}{2}}{2} + n_1 + 1$$

if $n_1$ is odd.

However, if $n_1$ is odd the above schedule is not optimal for the $C_{\max}$ criterion. But it is easy to see that this can be obtained by forming one block containing 3 jobs without changing the value of $\sum C_i$. For the further considerations we only need such an assertion for the following types of blocks in this case:

1. $|B'_1| = 3,\ |B'_2| = \ldots = |B'_{\lfloor n_1/2 \rfloor}| = 2$
2. $|B''_1| = \ldots = |B''_{\lfloor n_1/2 \rfloor - 1}| = 2,\ |B''_{\lfloor n_1/2 \rfloor}| = 3.$

Scheduling the blocks $B_k$ in the time intervals $[\underline{t}_k, \bar{t}_k]$ as described in Section 2, we obtain in case *a*) the value

$$\begin{aligned} f_2' &= 1 \cdot 2 + 2 \cdot 3 + 2 \cdot 5 + \ldots + 2 \cdot n_1 \\ &= 1 \cdot 2 + 2 \cdot 2 + 2 \cdot 4 + \ldots + 2 \cdot (n_1 - 1) \cdot \frac{n_1 - 1}{2} \\ &= 4 \cdot \binom{\frac{n_1+1}{2}}{2} + n_1 + 1 = f_2. \end{aligned}$$

Analogously, this can be shown for case *b*, *i.e.* we have $f_2'' = f_2$.

In [4] there has been given an $O(nm)$ algorithm for the problem $O|p_{ij} = 1, \text{outtree}|\sum C_j$ by decomposing the original problem into subproblems and processing blocks of jobs within $m$ time units. However, in our case we have only one subproblem for the outtree problem with $I_1$. Let $S_j'$ be the corresponding rank sets for the outtree problem. It is shown in [4] that for an arbitrary number $m$ of machines we have no idle time in the period $[0, rm]$ when

$$\sum_{j=1}^{u} |S_j'| \geq u \cdot m, \qquad u = 2, \ldots, r \tag{3.1}$$

holds. However, using the blocks $B_{k-1}, B_{k-2}, \ldots, B_1$ determined by Algorithm 1, it is immediately clear that condition (3.1) holds for $m = 2$ and all $u$ because each of these sets contains at least 2 jobs and we have no precedence constraint among the jobs of each block $B_j$. Hence, the Algorithm from [4] works for $O\,2|p_{ij} = 1, \text{outtree}|\sum C_i$ as follows:

Algorithm 2: Determination of the blocks for the problem $O\,2|p_{ij} = 1, \text{outtree}|\sum C_i$

```
  begin
1.   $U_1 := S_1'$;
2.   $j := 1$;
3.   $jmax := \lfloor n_1/2 \rfloor$;
4.   while $j \leq jmax$ do
5.       begin
6.           determine the block $B_j$ by selecting 2 jobs from $U_j$ having the largest number of
             jobs on a longest path from this job to a sink; $j := j + 1$;
7.           determine $U_j$ by replacing in $U_{j-1}$ all jobs of $B_{j-1}$ by their direct successors
       end;
8    if $n_1$ is odd then $B_{jmax+1} := U_{jmax+1}$.
  end.
```

Hence, applying the above algorithm to an odd number $n_1$ of jobs the last block contains only one job, *i.e.* we have idle times. In this case we modify the obtained solution by combining the last two blocks into a block of three jobs. Because of the above considered case *b*), this does not change the value of $\sum C_i$. The following lemma shows that this modification is always possible.

LEMMA 3: *Let $n_1$ be odd. Then we have no precedence constraint among the jobs of the last two blocks obtained by Algorithm 2.*

*Proof:* Assume that we have a precedence constraint between two jobs $u \in B_{jmax}$ and $v \in B_{jmax+1}$. Let job $r \in B_w$ be an ancestor of job $v$ such that any two adjacent blocks in the chain $C$ from $r$ to $v$ have been inserted into adjacent blocks and $w$ is as small as possible (*see* Figure 4).

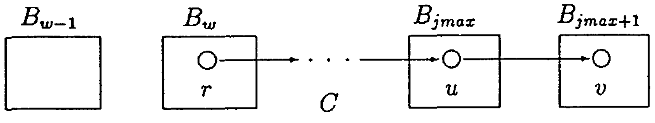

**Figure 4.**

Then we consider two cases:

1. $w = 1$: Due to the definition of $k$ when determining both subproblems, it is impossible that a job of $I_1$ has the rank $jmax + 1$.

2. $w > 1$: Then $B_{w-1}$ must contain two jobs from which a chain exists in the graph of precedence constraints (*i.e.* the graph of outtree constraints among the jobs of the set $I_1$) with at least as many jobs as in the chain $C$. Because of the outtree constraints these 3 chains are disjoint. However, this constradicts the fact that we have altogether only $2\,jmax + 1$ jobs.

Hence, we get the assertion of the lemma. ■

Thus, using the above modification we obtain a schedule that is optimal for both criteria $\sum C_i$ and $C_{\max}$ with respect to $I_1$ and outtree constraints. Now, processing the blocks of jobs in reversed order we get an optimal schedule for the intree constraints and both criteria (note that, if $n$ is odd, now the first block contains 3 jobs, which does not change the value of $\sum C_i$ due to case *a*), *i.e.* $f_2' = f_2$).

To evaluate the complexity of the above algorithm, note that the determination of the sets $S_k$ and $L_k$ can be done in $O(n)$ time for the intree constraints. At the same time the required ordering to non-decreasing $\bar{k}(i)$

values is automatically generated. Because we use Algorithm 1 to determine the set of jobs for problem $P_1$, the above algorithm has the same complexity. However, because we have only to determine the first unavoidable idle time, we only note here, that in the case of intree constraints algorithm 1 can be modified to run in $O(n)$ time. hence we obtain the following result:

THEOREM 3: *Algorithm 2 solves the problem* $O2|p_{ij} = 1, \text{intree}|\sum C_i$ *in* $O(n)$ *time.*

To illustrate the above algorithm, we consider the following example. Let $n = 12$ and the intree graph $G$ be as in Figure 5. For simplicity we avoid to add a fictitous sink because we have only one intree component.

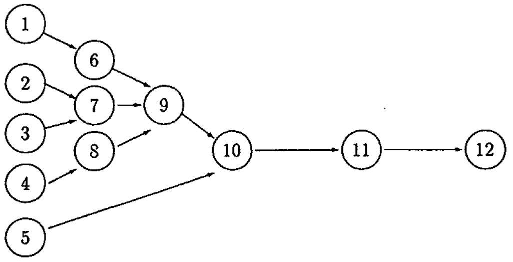

**Figure 5. - The graph** $G$.

Here we have $S_1 = \{1, 2, 3, 4, 5\}$, $S_2 = \{6, 7, 8\}$, $S_3 = \{9\}$, $S_4 = \{10\}$, $S_5 = \{11\}$ and $S_6 = \{12\}$. Moreover we obtain $L_1 = \{1, 2, 3, 4\}$, $L_2 = \{6, 7, 8\}$, $L_3 = \{5, 9\}$, $L_4 = \{10\}$, $L_5 = \{11\}$ and $L_6 = \{12\}$. Applying Algorithm 1, we get $I_1 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and $I_2 = \{10, 11, 12\}$. Transforming the precedence constraints among the jobs $I_1$ into an outtree, using the algorithm for $O2|p_{ij} = 1, \text{outtree}|\sum C_i$, $C_{\max}$ and processing the blocks of jobs in reversed order, we get the following blocks for the intree problem: $B_1 = \{1, 2, 3\}$, $B_2 = \{4, 6\}$, $B_3 = \{7, 8\}$ and $B_4 = \{5, 9\}$ with the makespan value $C_{\max} = 9$. Thus, the jobs 10, 11, 12 are processed consecutively in [9, 15] and we obtain the optimal objective value $\sum C_i = 89$.

## 4. CONCLUDING REMARKS

In this paper we gave polynomial time algorithms for two 2-machine open shop problems with unit processing times and different types of precedence constraints. The presented algorithms can also be used for solving the corresponding parallel machine problems with a simple straightforward modification. Whereas for the problem $P\,2|p_{ij} = 1, \text{prec}|C_{\max}$ there already exist some algorithms mentioned in the introduction, for the problem $P\,2|p_{ij} = 1, \text{intree}|\sum C_i$ there is no known algorithm from the literature.

However, the algorithm presented in Section 3 for the problem $O\,2|p_{ij} = 1, \text{intree}|\sum C_i$ does not necessarily lead to an optimal solution for the case of an arbitrary number of machines. To illustrate, we consider the following instance of this problem. We have to process 16 jobs on 3 machines and the graph of intree constraints is given in Figure 6.

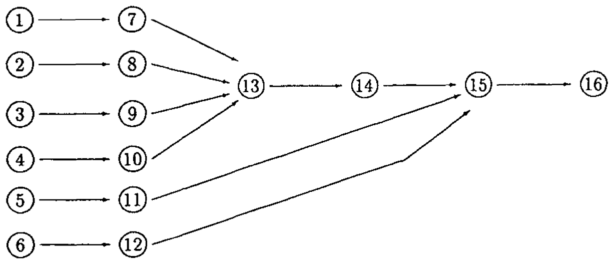

**Figure 6. – The graph** $G$ **for a problem** $O\,3|p_{ij} = 1, \text{intree}|\sum C_i$.

If we apply Algorithm 2 to this problem, we obtain $B_1 = \{1, 2, 3\}$, $B_2 = \{4, 7, 8\}$, $B_3 = \{9, 10, 5\}$ and $B_4 = \{6, 11, 13\}$, $B_5 = \{12, 14\}$, $B_6 = \{15\}$ and $B_7 = \{16\}$. The corresponding objective function value is $\sum C_i = 159$. However, if we choose $B_1 = \{1, 2, 3, 4\}$, $B_2 = \{7, 8, 9, 10\}$, $B_3 = \{13, 5, 6\}$, $B_4 = \{14, 11, 12\}$, $B_5 = \{15\}$ and $B_6 = \{16\}$ we can construct a schedule with $\sum C_i = 158$. In this case the optimal schedule has the property that for $B_1$ one job is finished at time 3 and the other three jobs are finished at time 4, for $B_2$ one job is finished at time 7 and the other tree jobs are finished at time 8 and so on.

However, some polynomially solvable cases of both problems $O|p_{ij} = 1, \text{prec}|C_{\max}$ and $O|p_{ij} = 1, \text{intree}|\sum C_i$ can easily be given. Here we mention only two cases:

1) for all $k$ with $1 \leq k < rkmax$ the condition $|S_k| \leq m$ hold or

2) for all $k$ with $1 \leq k < rkmax$ the condition $S_k = L_k$ holds.

In both cases it is easy to see that one can form blocks $B_j$ consisting of the jobs of the set $S_j$ and process each block $B_j$ within $\max\{|B_j|, m\}$ time units to determine an optimal solution. Therefore, the complexity status of the problem $O|p_{ij} = 1, \text{intree}|\sum C_i$ is still open.

## REFERENCES

1. H. Bräsel, *Lateinische Rechtecke und Maschinenbelegung*, Habilitationsschrift, TU Magdeburg, 1990.
2. H. Bräsel and M. Kleinau, On number problems for the open shop problem. System Modelling and Optimization, *Proceedings of the 15th IFIP Conference*, Zurich, Switzerland, Springer-Verlag, 1992, pp. 145-155.
3. H. Bräsel, D. Kluge and F. Werner, A polynomial algorithm for the $[n/m/O,\ t_{ij},\ \text{tree}/C_{\max}]$ open shop problem, *European J. Oper. Res.*, 1994, *72*, pp. 125-134.
4. H. Bräsel, D. Kluge and F. Werner, A polynomial algorithm for an open shop problem with unit processing times and tree constraints, to appear 1994 in Discrete Applied Mathematics.
5. P. Brucker, B. Jurisch and M. Jurisch, Open shop problems with Unit time operations, *ZOR*, 1993, *37*, pp. 59-73.
6. P. Brucker, B. Jurisch, T. Tautenhahn and F. Werner, Scheduling unit time open shops to minimize the weighted number of late jobs, *OR Letters*, 1994, *14*, pp. 245-250.
7. R. E. Graham, E. L. Lawlwer and J. K. Lenstra, Rinnooy Kan, A. H. G.: Optimization and approximation in deterministic sequencing and scheduling – a survey, *Ann. Discrete Mathematics*, 1979, *5*, pp. 287-326.
8. T. Gonzales and S. Sahni, Open shop scheduling to minimize finish time, *J. Assoc. Comput. Mach.*, 1976, *23*, pp. 665-679.
9. W. Kubiak, C. Sriskandarajah and K. Zaras, A note on the complexity of open-shop scheduling problems, *INFOR*, 1991, *29*, pp. 284-294.
10. C. Y. Liu and R. L. Bulfin, Scheduling open shops with unit execution times to minimize functions of due dates, *Operations Research*, 1988, *36*, No. 4, pp. 553-559.
11. R. McNaughton, Scheduling with deadlines and loss functions, *Management Science*, 1959, *6*, pp. 1-12.
12. V. S. Tanaev, Sotskov, N. Yu and V. A. Strusevich, *Scheduling theory. Multistage system*, Kluwer Academic Publishers, 1994.
13. T. Tautenhahn Open-shop-Probleme mit Einheitsbearbeitungszeiten, Dissertation, Otto-von-Guericke-Universität, Magdeburg, 1993.