

A. ADRABIŃSKI*, J. GRABOWSKI and M. WODECKI* (Wrocław)

ONE-MACHINE SEQUENCING WITH RELEASE DATES, DELIVERY TIMES AND PRECEDENCE CONSTRAINTS

Abstract. The purpose of this paper is to present an algorithm for scheduling jobs on a single machine in order to minimize maximum completion time, subject to release dates and delivery times. Moreover, precedence constraints are available. A branch and bound method is developed which incorporates some elements of the method of blocks from a critical path. The results of some computational experiments are also discussed. The problem is very important from a practical point of view as well as in theoretical considerations, since its resolution provides a bound on the makespan of more complicated sequencing problems such as the job shop.

1. Introduction. The problem can be formulated as follows. Each of n jobs J_1, J_2, \dots, J_n is to be processed without preemptions on a single machine which can handle at most one job at a time. The job J_i ($i = 1, 2, \dots, n$) becomes available for processing at time r_i , it has a non-zero processing time p_i and a delivery time q_i . Moreover, precedence constraints between jobs may be included. The objective is to find a processing order minimizing the time by which all the jobs are delivered.

This problem is also formulated in the following two ways:

1. Let M_1 and M_3 be non-bottleneck machines of infinite capacity and let M_2 be a bottleneck machine of capacity one (M_2 can process only one job at a time). Each job J_i is to be processed on the triple of machines (M_1, M_2 and M_3 in that order) and has to spend time r_i (head) on M_1 , p_i (body) on M_2 , and q_i (tail) on M_3 . We want to minimize the completion time (makespan) of all jobs.

2. A *due date* of job J_i is defined by $d_i = K - q_i$ for a constant K . Now, minimizing the time by which all jobs are processed and delivered is equivalent to minimizing the maximum lateness with respect to the due dates.

It is well known that the problem is NP-hard in the strong sense [8], which implies that a polynomial bounded algorithm for its solving is rather unlikely to exist. However, various enumerative algorithms exist (e.g., Baker and Su [2], Bratley et al. [3], Carlier [4], Lageweg et al. [7] and McMahon and

* This research was supported by Grant R.P.I.09 from the Institute of Informatics, University of Warsaw.

Florian [9]). In [7], a comparison between two of them (presented in [2] and [9]) is made, as well as a new efficient algorithm is described, which allows a very fast solution of problems with up to 80 jobs. Carlier [4] proposed an algorithm which resolved problems even with up to 10000 jobs.

In this paper, we present a branch and bound algorithm which includes new elements based on the method of blocks from a critical path. This method has been proposed by Grabowski [5] for the two-machine sequencing problem. It has been subject of extensive studies and has been applied to construct some algorithms for the job shop problems [1]. The present paper extends the results proposed there.

2. Basic definitions and properties. In this section we present a convenient representation of the problem provided by a disjunctive graph which was introduced by Roy and Sussman [12]. We follow the approach from [11], where the disjunctive graph model for job-shop has been applied. Then we introduce the idea of a block of jobs from the critical path.

Disjunctive graph. A disjunctive graph $G = (V, E)$, where V is the set of nodes and E is the set of directed arcs, is associated with an instance of the problem.

The set V represents the jobs, including fictitious initial and final jobs J_0 and J_* :

$$V = \{0, 1, \dots, n, *\},$$

and to each node $v \in V$ a weight p_i is assigned with $p_0 = p_* = 0$.

The set E of arcs consists of two subsets C and D ($E = C \cup D$), where C is the set of conjunctive arcs:

$$C = \{(0, i) \mid i = 1, 2, \dots, n\} \cup \{(i, *) \mid i = 1, 2, \dots, n\},$$

and D is the set of disjunctive arcs:

$$D = \{(u, v) \mid u, v \in V, u \neq v\},$$

representing the possible processing order on the machine. Each arc $(0, i)$ is weighted by r_i so that job i cannot start before time r_i and each arc $(i, *)$ is weighted by q_i so that job i cannot be delivered before time q_i after its processing by the machine.

The disjunctive graph of an example with three jobs defined in Table 1 is drawn in Fig. 1.

TABLE 1

Jobs	1	2	3
r_i	3	5	2
p_i	6	2	9
q_i	7	4	8

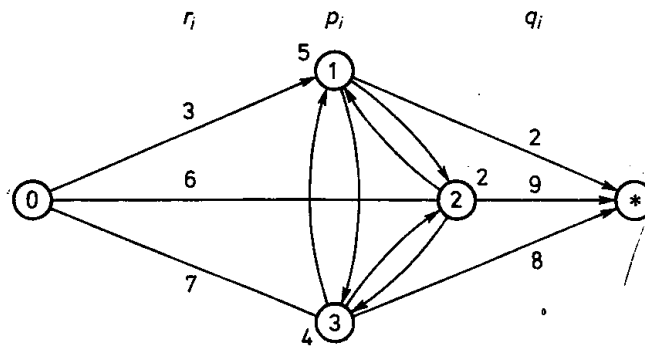


Fig. 1. A disjunctive graph

Any two jobs J_u and J_v are connected by a pair of disjunctive arcs $\{(u, v), (v, u)\}$. A pair of disjunctive arcs is called *fixed* if one of them has been chosen and the other one has been *rejected*; by choosing (u, v) , the precedence constraint of J_u over J_v is imposed (denoted by $J_u \ll J_v$), $J_u \ll J_v$ means that job J_v can begin after the completion of J_u . All fixed arcs form a subset $F \subset D$ of chosen arcs. It should be mentioned that all initial precedence constraints given in an instance of the problem are realized by fixing corresponding arcs connected with them.

A subset $S \subset D$ such that

- (i) $(u, v) \in S$ if and only if $(v, u) \in D - S$, and
- (ii) the graph $G(S) = (V, C \cup S)$ is acyclic

is called a *selection of the pairs of the disjunctive arcs*. However, if the transitive arcs are thrown out from S , the resulting subset $S' \subset S$ is called a *feasible partial schedule*. In what follows we associate with S' a feasible schedule S^h which defines a complete sequence of jobs (we write $S^h \simeq S'$).

Block. The *value of a feasible schedule S^h* is defined as the weight of a critical path in the graph $G(S^h) = (V, C \cup S^h)$. Our approach in solving the problem is to find a *minimaximal path* in G , i.e., a critical path in $G(S^h)$, $S^h \simeq S' \subset S$, that is minimal over all subsets $S \subset D$ satisfying the conditions (i) and (ii). To this end we define a block.

Let P be a critical path in $G(S^h)$ passing through the set B of jobs called a *block*. Let the first job on the path be u and the last one be v ; obviously, $u, v \in B$. The set $\bar{B} = B - \{u, v\}$ is called an *internal block* for B . Now, we can define the length of the critical path P as:

$$L(P) = r_u + \sum_{i \in B} p_i + q_v.$$

The 6-job problem specified in Table 2 is drawn in Fig. 2. The critical path is marked by dashed lines and its block is marked by solid lines. The critical path length is equal to 23.

We can now prove an important property of a block B in a critical path P , which is of fundamental significance in the construction of our algorithm.

TABLE 2

Jobs	1	2	3	4	5	6
r_i	3	4	7	5	9	2
p_i	2	1	4	3	1	2
q_i	4	7	5	2	8	5

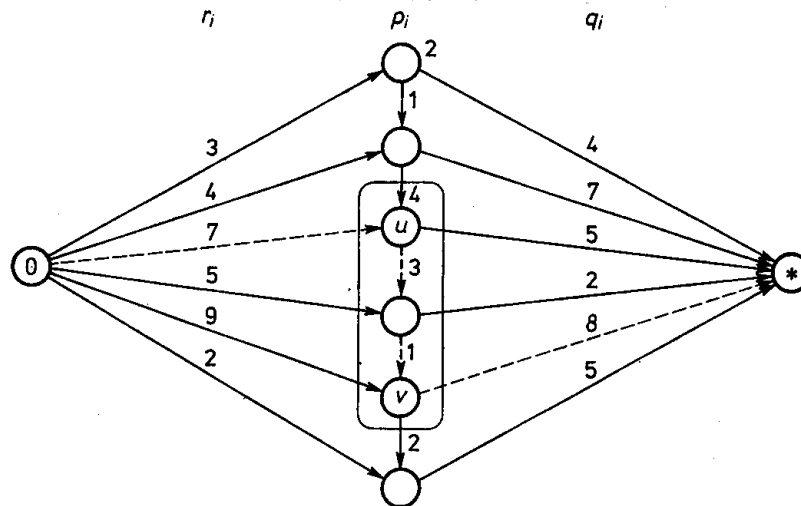


Fig. 2. Definitions of P and B

THEOREM 1. Let S_x^h be a feasible schedule and let B_x be a block in a critical path P_x in $G_x = (V, E \cup S_x^h)$. If this schedule is not optimal, then there is a feasible schedule S_y^h such that at least one job from $B_x - \{u_x\}$ (or from $B_x - \{v_x\}$) precedes (succeeds) all the remaining jobs of B_x in S_y^h .

Proof. Consider a feasible schedule S_y^h of length shorter than L_x . Such a schedule exists, because S_x^h is not optimal.

Let us suppose that no job in $B_x - \{u_x\}$ (and in $B_x - \{v_x\}$) precedes (succeeds) all the remaining jobs of B_x in S_y^h . Then, for each sequence of jobs in \bar{B}_x , there exists a path in G_y containing all the jobs of B_x of length at least L_x . This contradicts the assumption that G_y has a shorter critical path than L_x .

Theorem 1 implies that only certain changes of the sequence of jobs in the block can reduce the length of a critical path in the case where it is not minimal.

In the next section a branch and bound algorithm is described. With each node generated during the tree search we associate a subset $F \subset D$ of chosen disjunctive arcs. A lower bound on the length of all possible schedules S^h (such that $S^h \simeq S'$ and $S \supset F$) is calculated as well. The jobs in the block determine the branches in the enumerative scheme.

3. The algorithm. We now provide a complete description of our branch and bound algorithm. Initially, the set F_0 of chosen disjunctive arcs is constructed by adding to it each arc associated with the precedence constraints given in the definition of an instance. We can assume that the graph

$G_0 = (V, C \cup F_0)$ is acyclic since otherwise the instance is ill defined. Obviously, the selection S_0 is equal to F_0 . A search tree contains the initial graph G_0 and the initial selection S_0 connected with it, represented in the tree by the root node. We systematically apply Theorem 1 to obtain new descendant nodes in the search tree.

Consider a current node, called a *parent node*. Suppose that a graph

$$G(S) = (V, C \cup S)$$

is associated with this node, where S is a selection of pairs of disjunctive arcs. We construct for S a partial feasible schedule S' and apply a heuristic method to obtain a feasible schedule S^h associated with S' . The length $UB(S^h)$ of S^h is calculated in order to be compared with the current (best) upper bound UB^* . Good results are obtained with Shrage's heuristic which can be stated as follows [10]:

Upper bound. Suppose that some jobs have been already sequenced in the first k positions and the machine becomes available at time T_k . Let r_k be equal to the smallest release date of unsequenced jobs. Then the job to be sequenced in position $k+i$ is chosen from the set

$$\{i \mid i \text{ unsequenced, } r_i \leq \max(T_k, r_k)\}$$

so that its delivery time is as large as possible. If there is a choice, the job with the largest processing time is chosen.

The best solution S_*^h and the best upper bound UB^* are updated only if $UB(S^h) < UB^*$. In this case we set

$$UB^* := UB(S^h) \quad \text{and} \quad S_*^h := S^h.$$

Consider a critical path P passing through the block of jobs B . We may assume that the number of jobs in B is at least 1, otherwise we backtrack to the predecessor of the current node.

Now, we check if a job of $B - \{u\}$ could be a predecessor of the remaining jobs of the block. Such a job is called a *good job* and good jobs form the set of l -generators G_l . Similarly, we can form the set of r -generators G_r . Obviously, either u belongs to G_r or v belongs to G_l . Next we show how further generators can be eliminated from the sets G_l and G_r .

Branching rule. Selecting a job j ($j \in G_l$ or $j \in G_r$) we generate a new descendant in the search tree. Assume that $j \in G_l$, i.e., j should be processed before the remaining jobs of G_l in all descendants of the current node. Let us put

$$F_l^j = \{(j, k) \mid k \in G_l\}.$$

If $j \in G_r$, i.e., j should be processed after the remaining jobs of G_r in all descendants of the current node, we define

$$F_r^j = \{(k, j) \mid k \in G_r\}.$$

The complementary arcs of all arcs belonging to F_l^j (or to F_r^j) are rejected.

We next update S according to

$$S := \begin{cases} S \cup F_l^j & \text{if } j \in G_l, \\ S \cup F_r^j & \text{if } j \in G_r. \end{cases}$$

Now the graph $G(S) = (V, C \cup S)$ corresponds to a new node in the search tree and a lower bound is calculated.

Assume that $j \in G_l$. If $j \in G_r$, the argumentation is identical.

Lower bound. We seek to find a lower bound (LB) on the length of a critical path in $G(S'')$ over all $S'' \subset S$ (i.e., on the lengths of a critical path in all possible descendants of the parent node).

The adequate LB is calculated in the following way.

Firstly, it is simple to see that

$$LB_1 = \min_{i \in G_l} r_i + \sum_{i \in G_l} p_i + \min_{i \in G_l} q_i.$$

Secondly, it is obvious that

$$LB_2 = \max \{r_i + p_i + q_i \mid i = 1, 2, \dots, n\}.$$

Thirdly, and lastly, suppose that

$$q_i^* = \min \{q_i \mid i = 1, 2, \dots, n\}.$$

We obtain the value of LB^1 using Jackson's rule [6] for the one-machine sequencing problem with $\{r_i\}$, $\{p_i\}$ and precedence constraints (with respect to S). Symmetrically, we suppose that

$$r_i^* = \min \{r_i \mid i = 1, 2, \dots, n\}$$

and resolve the one-machine sequencing problem with $\{p_i\}$, $\{q_i\}$ and precedence constraints (over S) to obtain LB^2 . Then

$$LB_3 = \max \{LB^1, LB^2\}.$$

Now LB is given by

$$LB = \max \{LB_1, LB_2, LB_3\}.$$

If $LB \geq UB$, then the node is eliminated and we backtrack, otherwise it becomes an active node.

For each parent node of the search tree we choose a descendant with the lowest lower bound to branch from it.

Backtracking. If either $LB \geq UB^*$ or G_l and G_r are empty at the node generated by job j , we backtrack setting

$$S := \begin{cases} S - F_l^j & \text{if } j \in G_l, \\ S - F_r^j & \text{if } j \in G_r. \end{cases}$$

Remarks. 1. We may assume that if $j \in G_r$ is considered, all generators $k \in G_l$ have been already investigated. Then we can create a set

$$F^u = \{(u, k) \mid k \in G_l\}$$

of chosen arcs and update F_r^j setting

$$F_r^j := F_r^j \cup F^u.$$

2. We may simply implement the arcs belonging to S by adjusting the corresponding adequate r_i and g_i , similarly as was described in [7]. This is very effective during the computation of the heuristic schedule S^h as well as during the calculation of LB's.

3. Let L be the length of a critical path in $G = (V, C \cup S)$ at a current node. Now, we define

$$d_1 = L - \text{UB}.$$

Obviously, $d_1 \geq 0$.

We now show how the set G_l or G_r can be reduced.

LEMMA 1. *If a descendant is generated for $j \in G_l$ and*

$$r_j \geq r_u - d_1,$$

then

$$L' \geq \text{UB},$$

where L' is the length of a critical path in the graph associated with the node.

Proof. Let G'' be a graph associated with a descendant. Since there is a path in G'' whose length is not shorter than $L - (r_u - r_j)$, the proof is completed.

Symmetrically, let

$$d_2 = \min_{i \in G_r} \{d_1 - r_u + r_i\}.$$

Obviously, $d_2 \geq 0$.

LEMMA 2. *If a descendant is generated for $j \in G_r$ and*

$$g_j \geq q_v - d_2,$$

then

$$L' \geq \text{UB},$$

where L' is the length of a critical path in the graph associated with the node.

Proof. The motivations establishing the proof are similar to those for Lemma 1.

4. Computational experiments. The algorithm was coded in ALGOL4/5 on the Odra 1305 computer and it was tested on problems with 25, 50, 75, 100 and 150 jobs with randomly generated parameters. Integer processing times p_i were uniformly distributed in $[1, p_{\max}]$, where

$$p_{\max} = 10, 25, 50, 100,$$

and the release dates r_i and delivery times q_i were uniformly distributed integers in $[1, r_{\max}]$ and $[1, q_{\max}]$, respectively, where

$$r_{\max} = 25 \times n \quad \text{and} \quad q_{\max} = 25 \times m, \quad n, m = 1, 2, 3, 4, 6, 8, 10, 12.$$

All instances were solved optimally within the time limit of 30 seconds. About 75% of the tested problems were resolved optimally either by Shrage's heuristic or by LB's in the root node of the search tree.

References

- [1] A. Adrabiński, *The sequencing algorithms with parallel machines for discrete production systems*, PhD Thesis (in Polish), Wrocław 1983.
- [2] K. Baker and Z.-S. Su, *Sequencing with due dates and early start times to minimize maximum tardiness*, Naval Res. Logist. Quart. 21 (1974), pp. 171–176.
- [3] P. Bratley, M. Florian and P. Robillard, *On sequencing with earliest starts and due dates with application to computing bounds for the $(n/m/G/F_{\max})$ problem*, ibidem 20 (1973), pp. 57–67.
- [4] J. Carlier, *The one machine sequencing problem*, European J. Oper. Res. 11 (1982), pp. 42–47.
- [5] J. Grabowski, *On two-machine scheduling with release and due dates to minimize maximum lateness*, Opsearch 17 (1980), pp. 133–154.
- [6] J. R. Jackson, *Scheduling a production line to minimize maximum tardiness*, Research Report 43, Management Sci. Res. Project, Univ. of California 1955.
- [7] B. J. Lageweg, J. K. Lenstra and A. H. G. Rinnooy Kan, *Minimizing maximum lateness on one machine: computational experience and some applications*, Statist. Neerlandica 30 (1976), pp. 25–41.
- [8] J. K. Lenstra, A. H. G. Rinnooy Kan and P. Brucker, *Complexity of machine scheduling problems*, Ann. Discrete Math. 1 (1977), pp. 343–362.
- [9] B. McMahon and M. Florian, *On scheduling with ready times and due dates to minimize maximum lateness*, Oper. Res. 23 (1975), pp. 475–482.
- [10] C. N. Potts, *Analysis of heuristic for one machine sequencing with release dates and delivery times*, ibidem 28 (1980), pp. 1436–1441.
- [11] A. H. G. Rinnooy Kan, *Machine scheduling problems: classification, complexity and computations*, NIJHOFF, The Hague 1976.
- [12] B. Roy and B. G. Sussman, *Les problèmes d'ordonnement avec contraintes disjonctives*, Note D. S. no. 9 bis, SEMA, Montrouge 1964.

ANDRZEJ ADRABIŃSKI and MIECZYSLAW WODECKI
 INSTITUTE OF COMPUTER SCIENCE
 UNIVERSITY OF WROCLAW
 51-151 WROCLAW

JÓZEF GRABOWSKI
 TECHNICAL UNIVERSITY OF WROCLAW
 INSTITUTE OF TECHNICAL CYBERNETICS
 50-370 WROCLAW

Received on 1988.04.07