J. Operations Research Soc. of Japan
Vol. 14, No. 2, September 1971.


# GENERAL SCHEDULING ALGORITHMS WITH APPLICATIONS TO PARALLEL SCHEDULING AND MULTIPROGRAMMING SCHEDULING

**ICHIRO NABESHIMA**
*University of Electro-Communications*
*Chofu-shi, Tokyo*

(Received January 22, 1971 and Revised March 19, 1971)

**Abstract**

Branch-and-bound algorithms are presented on related disjunctive graph for general job-shop scheduling problems with various objective functions for two cases, such that set-up times are included in processing times in one case and not in the other case. They are applied to parallel scheduling and multiprogramming of programs and/or multiproject sheduling.

## 1. Introduction

The branch-and-bound method is a principal application of the fundamental state equation approach in state transformation process defined for combinatorial problems such as job-shop scheduling problem. [5].

Three branch-and-bound algorithms 1, 2, and 3 are presented on disjunctive graph for general job-shop scheduling problem with various objective functions for two cases where set-up times are included in processing times in one case and not in the other case. Algorithm 3 is applied to parallel scheduling and multiprogramming of computer programs and/or multiproject scheduling.

The research for finding an optimal schedule on disjunctive graph, which minimizes the total elapsed time (makespan) of N operations of n

jobs on m machines along the required technological orders, had been started from a model introduced by Roy and Sussman [7], and Nghiem [6]. Recently Balas [1], [2] has proposed branch-and-bound algorithms for this problem by finding a critical path in a disjunctive graph with a complete set of arcs at each stage. On the other hand, Charlton and Death [3] has proposed a more efficient branch-and-bound algorithm for this problem by finding a critical path in a disjunctive graph with a specified subset of arcs at each stage. The algorithms presented in this paper are revised and extended forms of the latter algorithm, treating the problems with various objective functions uniformly. They can be applied to parallel scheduling and also to multiprogramming or multi-project scheduling, which makes them general scheduling algorithms.

## 2. General Problem where Set-up Times are Included in Processing Times

General job-shop scheduling problems considered first in this paper are assumed to satisfy the following restraints:

1). n jobs must be processed on m machines where these is no requirement for each job to visit all machines or to visit each machine only once.

2). Each job must be processed by specified machines along the required technological order which is independent of that of any other job, and which may be of PERT diagram type.

3). Processing of each job by each machine is called an operation. Each operation to be performed by each machine must be completed without interruption by another operation on the same machine.

4). Each machine can perform one and only one operation at a time. The available time of each machine is assumed to be the same as a strarting time of the processing of the first operation in m machines. For the case where non-equality of the available time of each machine occurs, the algorithms presented later also can be

applied with slight modifications.

5). Set-up times are included in processing times of the operations. The other case will be discussed in Section 7 and so on.

6). Objective function f to be minimized is assumed to be the one which increases its value in the wide sense when the completion time of any operation in a schedule is delayed without changing the order of operations on each machine in this schedule.

Hence objective functions with this property include: total elapsed time (makespan), weighted mean completion time of jobs, mean waiting time consumed by each operation to be processed by the next machine where the sum of all waiting times is equal to the sum of the completion times of all jobs minus the sum of the given processing times of all jobs, mean machine idle time where the sum of all machine idle times is equal to the sum of the completion times of all last operations on m machines minus the sum of the given processing times of all jobs, maximum lateness (tardiness), and weighted mean lateness (tardiness), where the weights are assumed to be non-negative, lateness $L_k$ of job $k$ is defined as the completion time $C(k)$ of the last operation for job $k$ minus due date $d_k$ of job $k$, that is, $L_k=C(k)-d_k$, and tardiness $T_k$ of job $k$ is defined to be Max $(L_k, 0)$.

Also any cost function which is an increasing function of some or all of these objectives can be taken as the objective function.

## 3. Formulation on Disjunctive Graph

Let job $k(k=1\sim n)$ involve $N_k$ operations and let the technological order for job $k$ be a required order $\left(\sum_{l=1}^{k} N_{l-1}+1, \sum_{l=1}^{k} N_{l-1}+2, \cdots, \sum_{l=1}^{k+1} N_{l-1}\right)$ of $N_k$ operations $\sum_{l=1}^{k} N_{l-1}+1, \sum_{l=1}^{k} N_{l-1}+2, \cdots, \sum_{l=1}^{k+1} N_{l-1}$ of job $k$ where $N_0=0$. Then the total number of operations of $n$ jobs is $N\equiv\sum_{k=1}^{n} N_k$.

Let $S_p$ be a set of all operations performed by machine $p$ ($p=1\sim m$), then any two operations in $S_p$ cannot be performed simultaneously by machine $p$ from assumption 4) in Section 2.

Then, let $X$ be a set of $N+2$ nodes where nodes 0, and $N+1$ are the source, and the sink, respectively, which represent dummy nodes, and all other $N$ nodes represent $N$ operations.

And let $G=(X, Z)$ be a finite directed graph, with $X$ as its set of nodes and $Z$ as its set of directed arcs which represent the required technological order of the operations of any job, that is, directed arc $(i, j)$ belongs to $Z$ if the operation corresponding to node $i$ directly precedes the operation corresponding to node $j$, or if $i=0$ or $j=N+1$. With each arc $(i, j)$ is associated a real positive number $p_i$ which represents the processing time of operation $i$, where $p_0=0$.

Let $t_i$ be the earliest starting time of operation $i$. Then $C_i=t_i+p_i$ is the earliest completion time of operation $i$ ($i=1\sim N$) where $t_{N+1}=$ Max $(C(1), \cdots, C(n))$ is the total elapsed time.

Hence it is sufficient to determine the value of $t_q$ in order to determine the value of $C_q$ for each last operation $q$ of any job or on any machine. This leads to the fact that we can represent any objective function $f$ as a function of all variables $t_q$ where operation $q$ represents the last operation of each job for any $f$ except for mean machine idle time, or the last operation on each machine for mean machine idle time $f$.

Then the general job shop scheduling problem with any objective function $f$ defined in 6) of Section 2 can be formulated as a critical path problem I on a disjunctive graph as shown below by considering the precedence relations of any two operations on the same machine $p$, that is, those included in the set $S_p$ ($p=1\sim m$):

Determine the values of all $t_q$ where operation $q$ is the last operation of each job or on each machine such that $f$ is minimized subject to

$$t_j-t_i\geqq p_i, \quad \text{for } (i, j)\in Z \tag{3.1}$$

$$(3.2)\qquad \left.\begin{array}{l} t_j - t_i \geqq p_i \\ \text{or } t_i - t_j \geqq p_j\ , \end{array}\right\} \text{ for } i,\ j \in S_p \qquad (p=1\sim m)$$

$$t_i \geqq 0\ . \qquad (i=0\sim N+1)$$

In any feasible schedule, one of inequalities (3.2) must hold for each disjunctive pair. An arc which satisfies one constraint of the disjunctive pair is called a disjunctive arc. An example is shown in Table 1 and Fig. 1 where dotted lines represent disjunctive arcs.

Table 1. Technological Order of Operations 1~8.

| | Machine | | |
|---|---|---|---|
| | $M_1$ | $M_2$ | $M_3$ |
| Job 1 | 1 | 2 | 3 |
| Job 2 | | 4 | 5 |
| Job 3 | 7 | 6 | 8 |

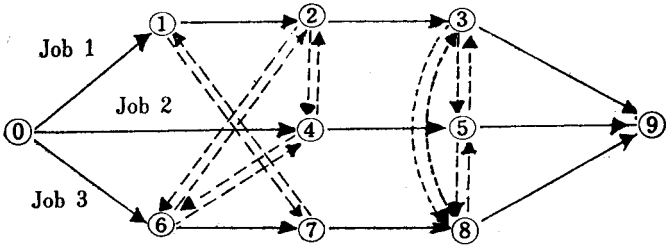


Fig.1. Related Disjunctive Graph where
X= {0, 1, 2, 3, 4, 5, 6, 7, 8, 9},
Z= {(0,1), (1, 2), (2, 3), (3, 9), (0, 4),
(4, 5), (5, 9), (0, 6), (6, 7), (7, 8), (8, 9)},
$S_1$= {1, 7}, $S_2$= {2, 4, 6}, $S_3$= {3, 5, 8},

By defining a set $S$ of disjunctive arcs which consists of one disjunctive arc from each disjunctive pair; that is, if $i, j \in S_p$ $(p=1\sim m)$ then $(i, j)\in S$ or $(j, i)\in S$; then the above formulation can be transformed to the following problem I:

**Problem I.** Determined the set $S$ along with the values of all earliest starting times $t_q$ above defined, such that $f$ is minimized subject to

$$t_j - t_i \geqq p_i, \quad \text{for } (i, j)\in Z\cup S$$
$$t_i \geqq 0. \quad (i=0\sim N+1)$$

Hence the general job shop scheduling problem can be converted to a critical path problem on related disjunctive graphs $G=(X, Z\cup S)$ where $S$ assumes the possible combinations of disjunctive arcs.

## 4. Branch-and-Bound Algorithms 1 and 2

In order to solve the critical path problem I on a disjunctive graph defined in Section 3, we must determine the optimal set $S$. Since the complete determination of the set $S$ by taking one arc from each disjunctive pair seems to make the algorithm for this problem complex, it would be better to follow a method similar to that proposed in Ref. [3], because, in case where set-up times are included in processing times, it does not make any circuit on the related disjunctive graph at each stage as shown in Section 5.

The method of branching of the following branch-and-bound algorithm 1 follows the newest active node search procedure as shown below:

***Branch-and-bound algorithm 1.***

*Step 1.* Solve Problem I with $S=\phi$ (null set) (node (1)) by using the recurrence relation:

$$t_j = \max_{(i, j)\in Z\cup S} (t_i+p_i)\,. \tag{4.1}$$

Proceed to Step 2.

*Step 2.* If the solution (the values of all $t_i$ and $f$) satisfies all disjunctive pairs (3.2), then this solution gives an optimal schedule.

Otherwise proceed to Step 3.

*Step 3.* Generally, if the solution of Problem I with $S$ does not satisfy all of disjunctive pairs, the solution value of $f$ gives a lower bound LB(f) of f for the optimal schedule, at this node. If this LB(f) is not smaller than the least upper bound where the upper bound is equal to the value of f for a feasible schedule already obtained, then go to Step 5. Otherwise, find a disjunctive pair for which $t_j-t_i<p_i$ and $t_i-t_j<p_j$ hold for $i, j\in S_p$ for certain $p$'s ($p=1\sim m$); introduce one disjunctive arc $(i, j)$ into $S$ (for example, for a positive $t_j-t_i$, or for a minimum $p_i-(t_j-t_i)$). In any way, the new set $Z\cup S$ of arcs does not contain a circuit. (cf. Section 5) Solve Problem I with the new $S$ (next node).

Proceed to Step 4.

*Step 4.* Generally, a) if the solution of Problem I does not satisfy all of disjunctive pairs, go to Step 3. b) If the solution of Problem I with $S$ at node $(n)$ satisfies all disjunctive pairs, then this gives a feasible schedule where the value of f is an upper bound UB(f), at this node, of f for the optimal schedule. If the least UB(f) is not greater than any LB(f) at all other nodes of degree 1 or 2 where degree of a node is defined as the number of arcs incident to this node, the feasible schedule with the least UB(f) gives an optimal schedule.

Otherwise proceed to Step 5.

*Step 5.* Backtrack up the tree to the nearest node $(n-k)$ of degree 2 or to the first node (1) among the nodes with a LB(f) smaller than the least UB(f), and proceed to Step 6.

*Step 6.* Remove from the present $S$ the disjunctive arcs which have been already introduced to the nodes passed by, and introduce the other disjunctive arc $(j, i)$ into the set $S$ at node $(n-k)$, in the case

where a disjunctive arc $(i, j)$ of the same pair has been already examined at the next node $(n-k+1)$. Then solve Problem I with this $S$ (node $(n+1)$). Go to Step 4.

Obviously another branch-and-bound algorithm can be formulated where the method of branching is the frontier search; that is,

***Branch-and-bound algorithm 2.***

Algorithm 2 differs from Algorithm 1 in the following:

1). Two branches corresponding to two disjunctive arcs $(i, j)$, $(j, i)$ are constructed when the disjunctive pair does not hold for $i, j \in S_p$ in Step 3 of Algorithm 1, and Problem I for each new node is solved. Then the first part of Step 6 is omitted.

2). Branching is made to a node which has the smallest lower bound among all nodes of degree 1.

3). An optimal schedule is found at a node with the least UB(f) which is not greater than LB(f) at any other node of degree 1.

## 5. Justification of the Algorithms and Some Remarks

1). The fact that the new set $Z \cup S$ of arcs, which is generated by introducing one disjunctive arc $(i, j)$ into the old $S$ when a disjunctive pair does not hold for $t_i$ and $t_j$ for $i, j \in S_p$ where $S$ is obtained from the solution of Problem I for the old set $Z \cup S$, does not contain any circuit (Steps 3, 6) is justified as follows: there is no circuit for the first graph $G=(X, Z \cup S)$ where $S=\phi$. If there is no circuit in the graph $G_1=(X, Z \cup S)$ at any stage, for any $i, j \in S_p$ which construct a disjunctive pair such that they can make a circuit with the arcs in $G_1$ either by arc $(i, j)$ or $(j, i)$, for example, if $t_j > t_i$ holds for $G_1$ (Fig. 2), then obviously $t_j > t_i + p_i$ holds, hence the disjunctive pair must hold for these $i, j \in S_p$.

As a consequence, any $i, j \in S_p$ for which the disjunctive pair does not hold for the old graph do not make a circuit in the new

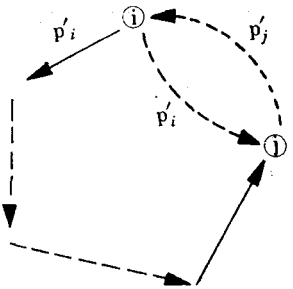


Fig. 2. $t_j > t_i$

graph obtained by introducing either arc $(i, j)$ or $(j, i)$ into the old graph.

2). Since there are finite number of disjunctive pairs for the problem, the algorithm gives an optimal schedule after finite steps.

3). Whenever Problem I must be solved for each new set $Z \cup S$, it is sufficient to calculate only the values of $t_i$ at nodes which can be reached by directed path from the extreme node of a newly introduced disjunctive arc, including the extreme node itself.

4). We can obtain an approximate schedule by suitably stopping the newest active node search procedure of Algorithm 1. (cf. Section 9).

5). If one takes the time $d_{ij}$ for $p_i$ for each arc $(i, j)$ where $d_{ij}$ represents the processing time $p_i$ plus set-up times incurred when operation $j$ follows operation $i$, then the procedure of Algorithms 1 and 2 may make a circuit as shown is example 4 in Section 6, which shows a counter example for the algorithm proposed in Ref. [3].

These generalized problems will be treated in Section 7.

## 6. Examples by Algorithm 1

Three examples 1, 2, 3: (min-makespan problem, min-mean completion time problem, and min-mean tardiness problem, respectively) will be solved by Algorithm 1, and it will be shown that Algorithm 1 which does not check circuits at any stage makes a circuit in the solution procedure for example 4 where $d_{ij}$ is taken for $p_i$.

***Example 1.*** (*Min-Makespan Problem*)

Three jobs 1, 2, 3, three machines $M_1$, $M_2$, $M_3$, the technological order for each job, and processing times and related operation (node) numbers are given in Table 2 where, for example, $\underset{(1)}{M_1^2}$

Table 2

| Job | Technological Order (Operation Number) | | | |
|---|---|---|---|---|
| 1 | $M_1^2$ (1) | $M_2^6$ (2) | $M_3^1$ (3) | |
| 2 | $M_3^4$ (4) | $M_2^2$ (5) | $M_1^3$ (6) | $M_2^5$ (7) |
| 3 | $M_1^3$ (8) | $M_3^2$ (9) | | |

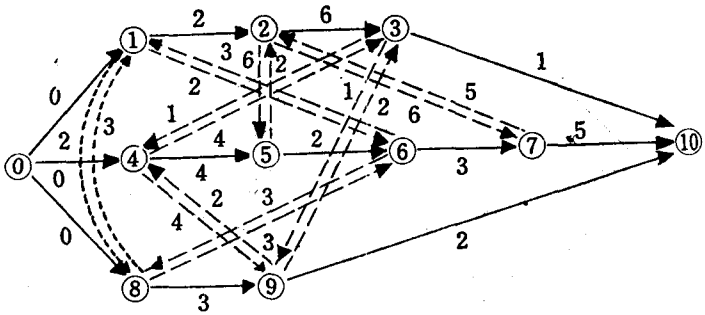


Fig.3 Disjunctive Graph for Example 1.

means that job 1 is first processed by machine $M_1$ with 2 units of processing time and this operation has operation (node) number 1 in the related disjunctive graph. (Fig. 3).

*Solution.* It is sufficient to minimize $t_{10}=\text{Max}\ (C_3, C_7, C_9)$.

The tree obtained by Algorithm 1 is shown in Fig. 4 where node numbers 1~11 show the order of branching and the number at each node represents the lower bound or upper bound.

In this solution procedure a disjunctive arc $(i, j)$ with the smallest

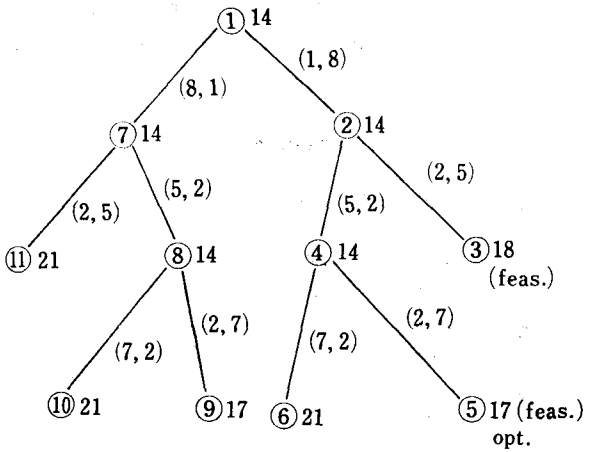


Fig. 4. Tree for Example 1.

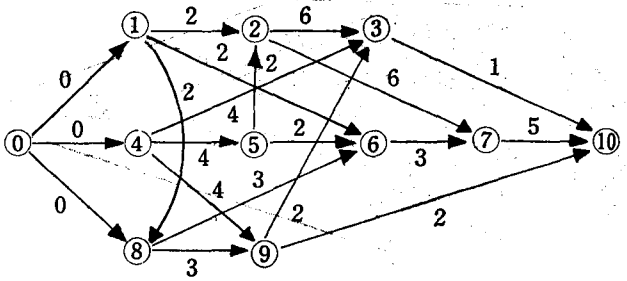


Fig. 5. Graph of the Optimal Schedule

node number $i$ or $j$ and with smaller $p_i-(t_j-t_i)$ or positive $t_j-t_i$ for these $i$, $j\in S_p$ was introduced into $S$. (Step 3).

An optimal schedule is a feasible schedule corresponding to node 5 with a makespan of 17 where $G=(X, Z\cup\{(1,8), (5,2), (2,7)\})$ as shown in Fig. 5. The corresponding Gantt Chart is shown in Fig. 6 where each number denotes a job number.

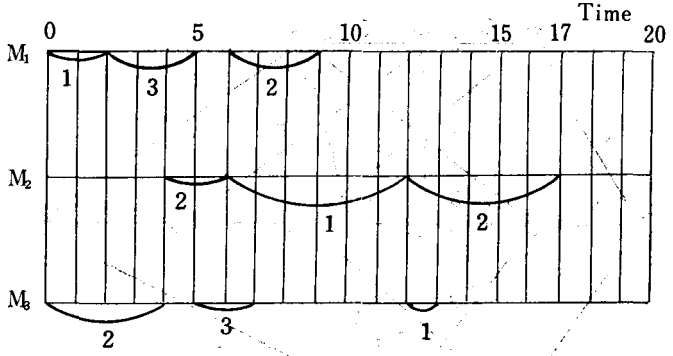


Fig.6. Gantt Chart of the Optimal Schedule

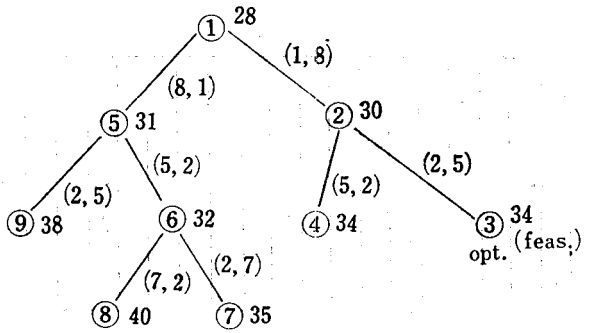


Fig.7. Tree for Example 2

***Example 2.*** *(Min-Mean Completion Time Problem)*

For the same data as example 1, it is sufficient to minimize the sum of the completion times of three jobs $C=C_3+C_7+C_9=(t_3+p_3)+(t_7+p_7)+(t_9+p_9)$.

The solution tree is shown in Fig. 7. An optimal schedule is a feasible schedule corresponding to node 3 with the minimum $C=34$,

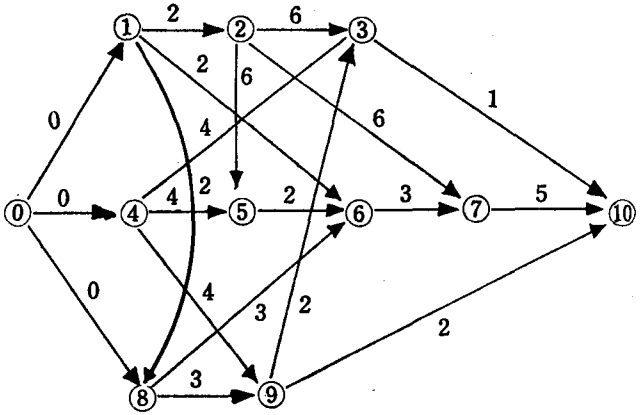


Fig.8. Graph of the Optimal Schedule

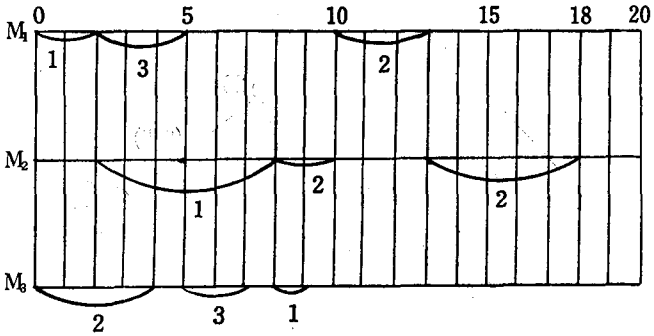


Fig.9. Gantt Chart of the Optimal Schedule

that is, a mean completion time of $11\frac{1}{3}$ where $G=(X, Z\cup\{(1,8), (2,5)\})$. The makespan for this schedule is 18, which is not minimum. The corresponding graph and Gantt Chart are shown in Figs. 8 and 9, respectively.

***Example 3.*** *(Min-Mean Tardiness Problem)*

For the same data as example 1 and additional data for due dates shown in Table 3, it is sufficient to minimize the sum of tardiness of three jobs $T=\text{Max}\ (C_3-18,\ 0)+\text{Max}\ (C_7-23,\ 0)+\text{Max}\ (C_9-5,\ 0)$.

Table 3. Due Dates.

| Job $i$ | Due date $d_i$ |
|---|---|
| 1 | 18 |
| 2 | 23 |
| 3 | 5 |

Since $p_1+p_2+p_3<d_1$, $p_4+p_5+p_6+p_7<d_2$ and $p_8+p_9=5=d_3$ hold,

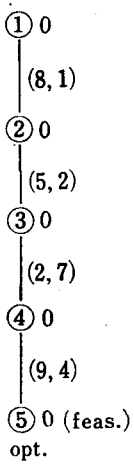


Fig.10. Tree for Example 3

one disjunctive arc $(i,j)$ which is introduced at Step 3 of Algorithm 1 is taken as $i=8$ or 9 whenever $i$ or $j$ is equal to 8 or 9.

The solution tree is shown in Fig. 10. An optimal schedule is a feasible schedule corresponding to node 5 with the minimum $T=0$, or the mean tardiness $=0$, where $G=(X, Z\cup\{(8,1), (5,2), (2,7), (9,4)\})$. The makespan for this schedule is 22.

The corresponding graph and Gantt Chart are shown in Figs. 11 and 12, respectively.

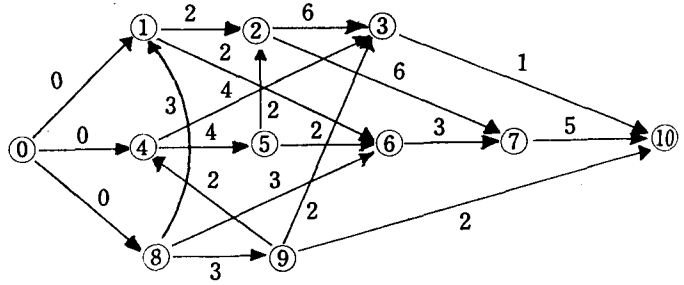


Fig.11. Graph of the Optimal Schedule

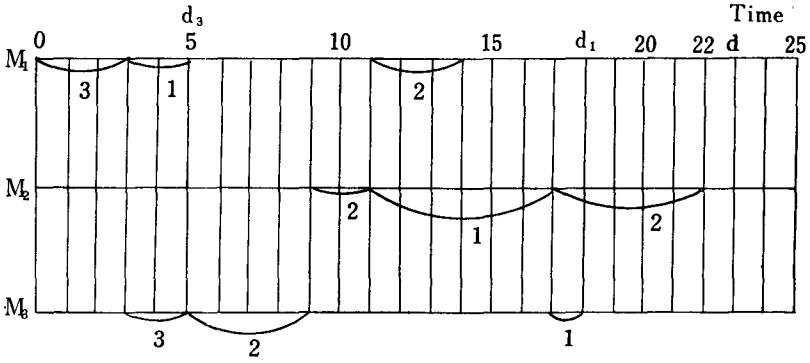


Fig.12. Gantt Chart of the Optimal Schedule

***Example 4.*** *(Min-Makespan Problem where set-up times are not included in processing times and sequence dependent)*

This example illustrates that Algorithms 1 and 2 can not be applied to the case where explicit set-up times exist.

Let two jobs 1, 2 be processed on two machines $M_1$, $M_2$ along the technological order shown in Table 4. The duration $d_{ij}$ which is the sum of the processing time $p_i$ of operation $i$ and set-up time $S_{ij}$ incurred to perform operation $j$ after $i$ is shown in Table 5.

Table 4. Technological Order.

| Job | Technological Order (Operation Number) |
|---|---|
| 1 | $M_1$ (1) $M_2$ (2) $M_1$ (3) |
| 2 | $M_1$ (4) $M_2$ (5) |

Table 5. Duration $d_{ij}$.

| $i$ \ $j$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 0 | 0 | | | 0 | | |
| 1 | | 3 | | 3 | | |
| 2 | | | 1 | | 1 | |
| 3 | | | | 5 | | 5 |
| 4 | 4 | | 10 | | 4 | |
| 5 | | 6 | | | | 6 |

The corresponding disjunctive graph is shown in Fig. 13.

Each disjunctive arc which must be introduced into $S$ was taken

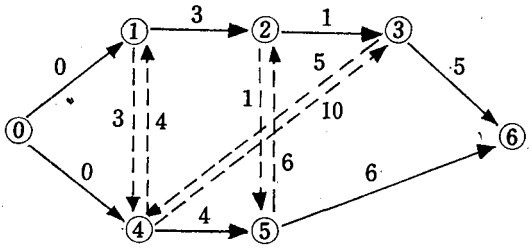


Fig.13. Disjunctive Graph for Example 4

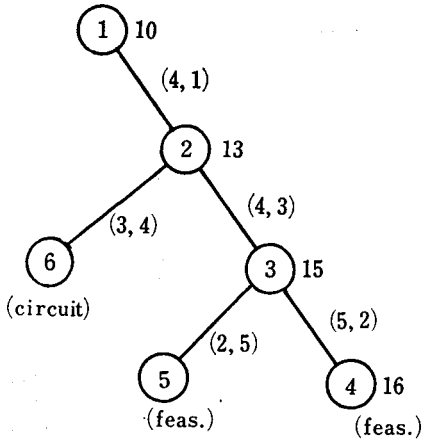


Fig.14. Tree for Example 4

as shown in the tree. (Fig. 14)

Then a circuit 1 2 3 4 1 is constructed at node 6 where $G=(X,\ Z\cup\{(4,1),\ (3,4)\})$.

Hence certain checks for the circuit must be made at Step 3 in the present algorithms 1 and 2 in case where explicit set-up times exist; that is, where durations $d_{ij}$ are taken for processing times $p_i$.

Algorithms for this generalized problem will be presented in the next section.

## 7. Generalized Branch-and-Bound Algorithms for Generalized Problem with Explicit Set-up Times

The generalized problem is defined as a problem which is different from the general problem presented at Section 2 in the restraint 5) such that set-up times are not included in processing times and then duration $d_{ij}$ is taken for $p_i$ for each arc $(i, j)$ in the related disjunctive graph.

Branch-and-bound algorithms 1 and 2 for this generalized problem can be constructed only by adding the following statements for checking circuits at Step 3 and Step 6 in the previous Algorithms 1 and 2; that is

***Generalized Algorithm 1.***

Steps 1, 2, 4, 5 are the same as those in Algorithm 1 of Section 4. Steps 3 and 6 are improved as below:

At Step 3, always introduce one disjunctive arc $(i, j)$ where $t_j-t_i$ is positive. If both $t_j-t_i$ and $t_i-t_j$ equal zero, introduce one disjunctive arc $(i, j)$ which makes no circuit with the present $Z \cup S$. If this is impossible for any candidate, the problem has no feasible solution.

At Step 6, check the circuit which may be constructed by introducing the other disjunctive arc $(j, i)$.

If there happens no circuit, the procedure proceeds as stated in Algorithm 1. Otherwise, the corresponding node $(n+1)$ must be omitted and back to Step 5 in order to backtrack to the next nearest node, at Step 6.

Similarly *the generalized branch-and-bound algorithm 2* taking the frontier search procedure is constructed by the same improvements of the corresponding Algorithm 2 of Section 4.

## 8. Example for Generalized Problem

Example 4 presented in Section 6 can be solved by Generalized Algorithm 1 if we continue the branching along the same procedure as

that shown in Fig. 14. Then the tree for this example is completed as shown in Fig. 15.

An optimal schedule is a feasible schedule corresponding to node 5 with a makespan of 15 where $G=(X,\ Z\cup\{(4,\ 1),\ (4,\ 3)\ (2,\ 5)\})$.

The corresponding graph and Gantt Chart are shown in Figs. 16 and 17, respectively.

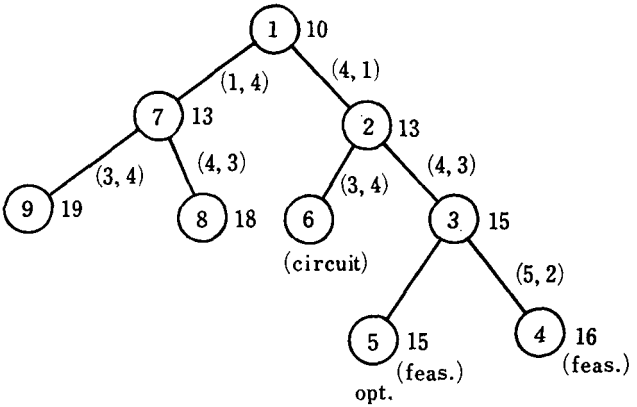


Fig.15. Tree for the Example

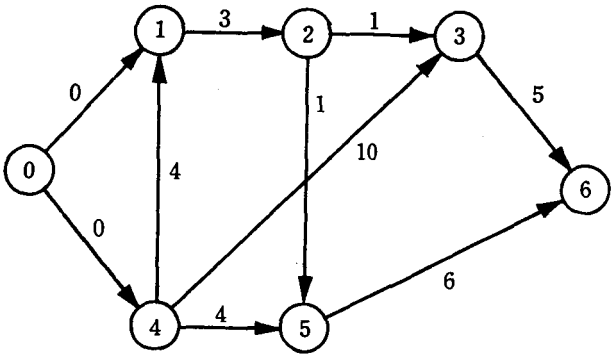


Fig.16. Graph of the Optimal Schedule

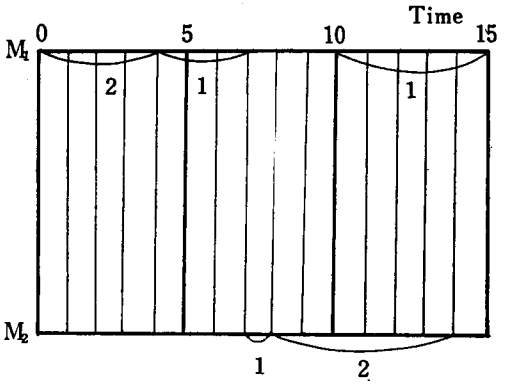


Fig.17. Gantt Chart of the Optimal Schedule

## 9. Heuristic Program for Obtaining an Approximate Schedule

For a problem with excessively large number of operations, an approximate schedule of any problem stated in Sections 2 and 7 can be obtained by using Algorithm 1, with stopping its procedure when it terminates to the feasible schedule first obtained, for example.

Then the percentage of the relative error (PRE) of the value of f of this approximate schedule is bounded as shown in the following inequality:

(9.1)

$$\mathrm{PRE}=\frac{(\text{Value of f of Appr. Schedule})-(\text{Min. f of Opt. Schedule})}{(\text{Min. f of Opt. Schedule})}\times 100$$

$$\leqq\frac{(\text{Value of f of Appr. Schedule})-(\text{Lower bound of f found})}{(\text{Lower bound of f found})}\times 100$$

For example, this upper bound (9.1) for each of the examples 1~4 becomes 28.6%, 21.4%, 0%, 60% respectively when the lower bound of f is taken as that at the first node (node 1). Also they become 28.6%, 13.3%, 0%, 23.1% respectively when the lower bound of f is taken as the minimum value of two lower bounds at two nodes which follow directly the first node.

The largest lower bound of f can be known at each stage. Hence, by giving a prescribed value of the upper bound (9.1), say 20% or 30%, one can stop the procedure whenever a feasible schedule is found. By checking whether or not it gives a new upper bound (9.1) which is not greater than the prescribed value, one can take this feasible schedule as a desired approximate schedule when it gives.

## 10. Algorithm 3 for Sensitivity Analysis and Parallel Scheduling

In this section another algorithm, called Algorithm 3, is presented for optimal scheduling of $n$ jobs to one shop. This algorithm is somewhat complex, but useful for sensitivity analysis and especially applicable to parallel scheduling of $n$ jobs to $q$ stations (shops) each of which has the same kind of machines, in order to minimize the maximum among $q$ minimal values of the objective function f at $q$ stations.

Algorithm 3 has a feature which resembles the procedure of dynamic programming and uses Algorithm 1 or 2 presented in the previous sections at each stage; that is, it is composed of $n$ stage decision process in state transformation process [5] as shown below:

***Algorithm 3.***

*Stage 1.* For each job $i$, calculate the value of f. Let $F_1(i)$ be this value for job $i$ ($i=1\sim n$), and store all $F_1(i)$.

The above can be done by following Step 1 of Algorithm 1 or 2.

*Stage 2.* For each combination of any two jobs $(i_1, i_2)$ among $n$ jobs, apply Algorithm 1 or 2 to a tree starting from the first node with a lower bound Max $(F_1(i_1), F_1(i_2))$ in order to obtain the minimum value of $f$, $F_2(i_1, i_2)$, for the job-set $i_1$, $i_2$. Store the combination $(i_1, i_2)$, $F_2(i_1, i_2)$, the value of the bound of $f$, and the set $S$ of disjunctive arcs assigned for each node of degree 1 or 2 in the solution tree; these are the contents of each solution-tree necessary for the next stage.

*Stage 3.* For each combination of any three jobs $(i_1, i_2, i_3)$ among

$n$ jobs, take out the stored nodes and the contents of the solution tree for definite two jobs $i_t$, $i_u$ ($t \neq u$; $t$, $u=1$, 2, 3) with the largest $F_2(i_t, i_u)$ defined at Stage 2, and replace the stored bound at each node of degree 1 or 2 by the maximum value between the stored bound and $F_1(i_w)$ for $w \neq t$, $u$, where $w=1$, 2, 3.

Then continue the procedure of Algorithm 1 or 2 in order to grafting the tree with the first fictitious node 1, when the set of stored nodes does not include a node with $S=\phi$, and with each other stored nodes having revised bound as each other nodes which directly follow node 1, finally obtaining the minimum value of $f$, $F_3(i_1, i_2, i_3)$, for the job-set $i_1, i_2, i_3$.

Store the combination $(i_1, i_2, i_3)$, $F_3(i_1, i_2, i_3)$, the value of the bound of $f$, and the set $S$ of disjunctive arcs assigned for each node of degree 1 or 2 in the solution tree; these are the contents of each solution tree necessary for the next stage.

Proceed in the same way to Stage $k$ ($k=3, 4, \cdots, n-1$):

*Stage k.* ($k=3, 4, \cdots, n-1$).

For each combination of any $k$ jobs $(i_1, i_2, \cdots, i_k)$ among $n$ jobs, take out the stored contents of the solution tree for definite $(k-1)$ jobs among these $k$ jobs with the largest $F_{k-1}$ defined at Stage $(k-1)$, and replace the bound at each node of degree 1 or 2 by the maximum value between this stored bound and $F_1$ for one remaining job. Continue the procedure of Algorithm 1 or 2 in order to grafting the tree composed of the fictitious first node, if necessary, and the set of stored nodes with the revised bound as the set of nodes which directly follow the first node, finally obtaining the minimum value of $f$, $F_k(i_1, i_2, \cdots, i_k)$, for the job-set $i_1, i_2, \cdots, i_k$. Store the combination $(i_1, i_2, \cdots, i_k)$, $F_k(i_1, i_2, \cdots, i_k)$, the value of the bound of f, and the set $S$ of disjunctive arcs assigned for each node of degree 1 or 2 in the solution tree; these are the contents of each solution tree necessary for the next stage.

*Stage n.* For all $n$ jobs, take out the stored contents of the solution tree for definite $(n-1)$ jobs with the largest $F_{n-1}$ defined at Stage $(n-1)$, and replace the stored bound at each node of degree 1 or 2 by the maximum value between this stored bound and $F_1$ for one remaining job. Continue the procedure of Algorithm 1 or 2 in order to grafting the tree constructed as before, finally finding the minimum value of $f$, $F_n$, and an optimal schedule.

Similarly Generalized Algorithm 3 for the generalized problem with explicit set-up times is formulated by taking Generalized Algorithm 1 or 2 for Algorithm 1 or 2, respectively, in Algorithm 3.

It must be remarked that, at each stage $k$ $(k=1\sim n-1)$ of Algorithm 3, the optimal schedule of any $k$ jobs among given $n$ jobs for the same objective function and its minimal value are determined. This shows the usefulness of Algorithm 3 for sensitivity analysis and for parallel scheduling as shown in the next section.

## 11. Parallel Scheduling

For the parallel scheduling of $n$ jobs to $q$ stations, where each of them has the same kind of machines, Algorithm 3 is applied first for total $n$ jobs until the procedure at stage $(n-q+1)$ is finished. Then, by determining the minimum value of the maximum among $q$ minimal values of $f$ at $q$ stations for each combination $(J_{n1}, J_{n2}, \cdots, J_{nq})$, $n=n_1+n_2+\cdots+n_q$, where $(J_{n1}, J_{n2}, \cdots, J_{nq})$ denotes an exhaustive and mutually exclusive separation of the set of $n$ jobs into $q$ subsets, the optimal parallel schedule with this minimum value can be found. A simple example will be shown in the next section.

## 12. Example by Algorithm 3

Algorithm 3 is applied to Example 1 of Section 6, and the optimal parallel schedule for two similar stations is found as below:

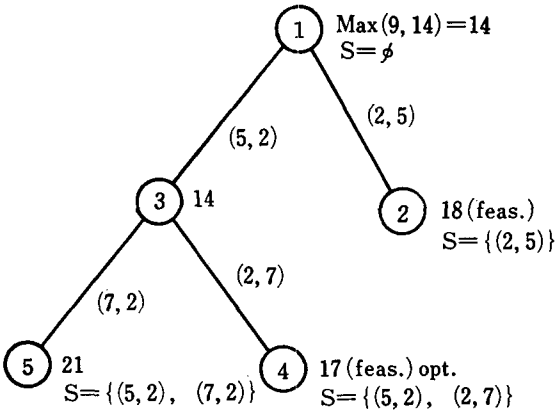


Fig.18. Tree for Two Jobs 1, 2

**I.** First, solve the min-makespan problem for all three jobs.

*Stage 1.* For each job $i$, determine the $F_1(i)$ by relation (4.1). The results are: $F_1(1)=9$, $F_1(2)=14$, and $F_1(3)=5$, which are stored.

*Stage 2.* For each set of two jobs $i_1$, $i_2$, determine the $F_2(i_1, i_2)$ by Algorithm 1 and store the necessary contents.

(1) For jobs 1, 2, the solution tree becomes Fig. 18. The results to be stored are: (1, 2); $F_2(1, 2)=17$; node 2 (18, $S=\{(2, 5)\}$), node 4 (17, $S=\{(5, 2), (2, 7)\}$), and node 5 (21, $S=\{(5, 2), (7, 2)\}$).

(2) For jobs 1, 3, the solution tree becomes Fig. 19. The results to be stored are: (1, 3); $F_2(1, 3)=9$; node 1 (9, $S=\phi$), and node 2 (9, $S=\{(1, 8)\}$).

(3) For jobs 2, 3, the solution tree becomes Fig. 20. The results to be stored are: (2, 3); $F_2(2, 3)=14$; node 1 (14, $S=\phi$), and node 2 (14, $S=\{(4, 9)\}$).

*Stage 3.* (*Final stage*)

Since the maximum value among all $F_2(i_1, i_2)$ at Stage 2 is $F_2(1, 2)=17$, the bound at each of stored nodes 2, 4, 5 is replaced by the maximum value between this stored bound and $F_1(3)$ stored at

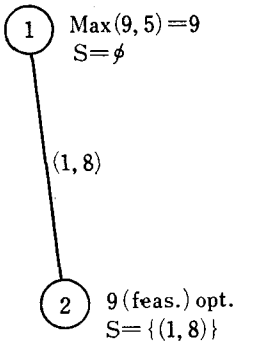


Fig.19. Tree for Two Jobs 1, 3

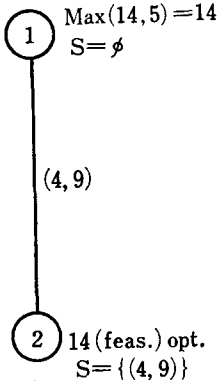


Fig.20. Tree for Two Jobs 2, 3

Stage 1, that is, Max (18,5)=18 for node 2, Max (17,5)=17 for node 4, Max (21,5)=21 for node 5. Thus the solution tree shown in Fig. 21 is constructed by continuing the procedure of Algorithm 1 to the revised tree and the same optimal schedule as that in Example 1 is determined.

**II.** Next, find the optimal parallel schedule using the results obtained at Stages 1 and 2.

For a separation (1; 2, 3), Max $[F_1(1), F_2(2,3)]$=Max [9,14]=14;
for (2; 1,3), Max $[F_1(2), F_2(1,3)]$=Max [14,9]=14;
for (3; 1,2), Max $[F_1(3), F_2(1,2)]$=Max [5,17]=17,
hence the minimum value is 14 for the separation (1; 2,3) or (2; 1,3) which gives the optimal parallel schedule.

***Remark*:** If it is assumed that $n$ jobs ($n \geqq 4$) including these three jobs 1, 2, 3 in the above example must be in optimum scheduled by Algorithm 3, then at Stage 3, for jobs 1, 2, 3 the following must be stored: node 2 (18, $S=\{(2,5)\}$), node 4 (17, $S=\{(5,2), (2,7)\}$), node 6 (17, $S=\{(5,2), (2,7), (1,8)\}$), and node 5 (21, $S=\{(5,2), (7,2)\}$). The same tree as Fig. 21 must be the starting tree with a further revised bound when it happened to use the stored nodes and their contents

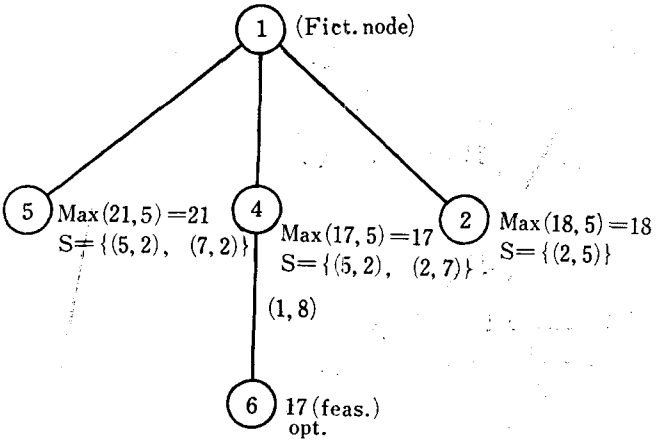


Fig.21. Tree for Three Jobs 1, 2, 3

for three jobs 1, 2, 3 at Stage 4.

## 13. Applications to Multiprogramming Scheduling

The computer system may be viewed as a machine shop, where the program corresponds to the job, the computer parts (input channels, processing units and out-put channels) to the machines of the machine shop, and the ordering of the subparts (subprograms) of the program on the computer parts to the technological ordering of the job.

By considering a computer with many such computer parts which can operate simultaneously, the flow diagram of each program is represented by a graph as shown in Fig. 22 where the subprogram (operation) number ($mji$) denotes the $j$th program on the $m$th computer part for the $i$th time, and the number $p_{mji}$ on each directed arc shows the processing time of operation ($mji$). [4]

Since one of the aims of multiprogramming is to schedule a set of programs such that the total elapsed time to complete all programs or the mean tardiness or any other objective function $f$ defined in Section 2 is

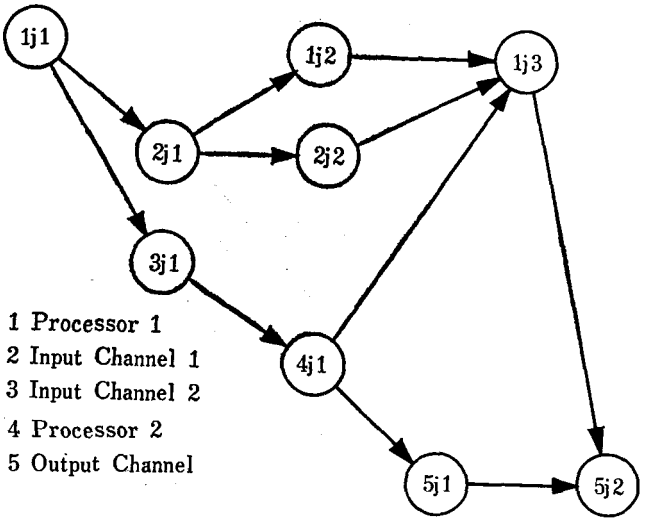


Fig.22. Example of a Graph for jth Program. [4]

a minimum, we can apply Algorithms 1, 2, or generally Generalized Algorithms 1, 2, to this complex scheduling problem for a computer. Similarly, we can apply Algorithm 3, or generally Generalized Algorithm 3, to parallel scheduling for separating the given finite set of programs to each of the finite number of computers where each of them has the same kind of computer parts, in order to minimize the maximum value among the minimal values $f$ for individual computers.

By the same reasons as above, Algorithms 1, 2, 3 are applied to the multiproject scheduling where each project is of PERT type.

## References

[1] Balas, E., "Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm," Operations Research, Vol. 17, No. 6, 1969, pp. 941-957.

[2] Balas, E., "Machine Sequencing: Disjunctive Graphs and Degree-Constrained Subgraphs," Nav. Res. Log. Quart., Vol. 17, No. 1, 1970, pp. 1-10.

[3] Charlton, J.M. and C.C. Death, "A Generalized Machine-Scheduling Algorithm," Oper. Res. Quart., Vol. 21, No. 1, 1970, pp. 127-134.

[4] Heller, J., "Sequencing Aspects of Multiprogramming," J. Ass. Comp. Mach., Vol. 8, No. 3, 1961, pp. 426-439.

[5] Nabeshima, I., "Dynamic Programming and State Transformation Process in Discrete Optimization Problems: Part I," Rep. of Univ. of Electro-Communications, No. 23, 1967, pp. 61-68.

[6] Nghiem, Ph. T., Les problèmes d'ordonnancement avec contraintes disjunctives, in book, Les problèmes d'ordonnancement, edt. by B. Roy, Dunod, 1964, pp. 136-151.

[7] Roy, B. and Sussman, B., Les problèmes d'ordonnancement avec contraintes disjunctives, SEMA, D.S. No. 9, 1964.