SCHEDULING PREEMPTABLE TASKS ON UNRELATED PROCESSORS WITH ADDITIONAL RESOURCES TO MINIMIZE SCHEDULE LENGTH

Roman Słowiński

Institute of Control Engineering, Technical University of Poznań,
60-965 Poznań, Poland

ABSTRACT The problem considered is that of scheduling n preemptable tasks on m parallel processors, when each task requires for its processing a processor and one resource unit from the set of additional resources. The processing times of a task on different processors are unrelated. We present the method for solving this problem which is composed of two stages. In the first stage, a linear programming problem is solved giving the minimum schedule length and optimal task processing times on particular processors. On the basis of this solution, in the second stage the optimal schedule is constructed taking into account the resource constraints. Theorems are proved concerning the feasibility of the second stage algorithm, and the upper bound on the number of preemptions in the optimal schedule. The cases of independent and dependent tasks are considered.

1. INTRODUCTION

In recent years we have been able to observe increased interest in scheduling problems associated with a certain model of a multiprocessor computing system /see [7] for a survey/. Much effort has been applied to problems concerned with cases where each task only requires one processor for its processing. In this paper, we consider an augmented multiprocessing model which allows for the possibility that certain tasks may require the use of various limited resources during their processing. Some special cases of this model were studied in [2,4,5,6,8] for various performance measures. However, all the previous studies assumed the processors to be identical. Even under this assumption, almost all problems are NP-complete [7] and hence they are computationally intractable. We shall be concerned with a problem which seems to be also NP-

complete, where processors are unrelated, i.e. the processing times of tasks on different processors are arbitrary. This problem, without additional resource constraints, has been considered in [3,9].

In Section 2, we describe the model of the computing system and give some basic definitions. In Section 3 and 4, a two-stage method for solving the problem is presented for the cases of independent and dependent tasks, respectively. Also in Section 3, theorems concerning the feasibility of the second-stage algorithm, and the upper bound on the number of preemptions in the optimal schedule are proven. Section 5 contains some final remarks.

2. THE MODEL OF THE COMPUTING SYSTEM

Let us describe the model of the computing system considered in this paper. Three finite sets are given, which are the main components of the model:

- the set of tasks $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$,
- the set of unrelated processors $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$,
- the set of additional resources $\mathcal{R} = \{R_1, R_2, \ldots, R_p\}$.

Each task T_j requires for its processing a processor and one unit of a specified resource. Let S_j be the set of processors which may execute T_j. Associated with each task T_j is the vector $\bar{t}_j = [t_{ij}]$, where t_{ij} is the time required for the execution of T_j by processor $P_i \in S_j$, provided that the specified additional resource unit is allotted to T_j. The processing of each task may be arbitrarily interrupted and restarted later without any time penalty, possibly on another processor. Moreover, permissible task orderings are determined by a set of precedence constraints given in the form of a "task-on-edge" directed acyclic graph with only one origin and only one terminal. The graph nodes /events/ are numbered from 1 to N in such a way that node j occurs not later than node k, if $j<k$. Such an ordering is always possible but may not be unique in a given precedence graph. We shall assume that only one ordering is imposed for a given problem.

Each processor P_i is able to process at most one task at a time. The set of tasks which may be processed on P_i will be denoted by C_i.

For each resource R_k there is a bound B_k which gives the total number of the resource units available at any given time. The set of tasks which require resource R_k will be denoted by D_k.

The objective is to minimize the finishing time /schedule length/ T of the set of tasks subject to the imposed constraints.

3. SCHEDULING INDEPENDENT TASKS

Let us consider the case of independent tasks, i.e. those which can be processed simultaneously.

We shall present a two-stage scheduling method which is a generalization of the method given in [3,9] for the model without additional resources. In the first stage, we find a generating schedule that is one which minimizes the schedule length T and gives optimal processing times of tasks on particular processors, but which does not necessarily fulfil the feasibility condition that some parts of a task are not executed simultaneously on more than one processor. In the second stage, on the basis of generating schedule, the optimal schedule is constructed which ensures that the feasibility condition and resource constraints are satisfied.

THE FIRST STAGE

Let $x_{ij} \in \langle 0, t_{ij} \rangle$ be the total processing time of task T_j on processor $P_i \in S_j$, $j=1,2,\ldots,n$. In order to find the generating schedule, we have to solve the following linear programming /LP/ problem.

Minimize T

subject to:

$$T - \sum_{T_j \in C_i} x_{ij} \geq 0 \qquad i=1,2,\ldots,m \qquad /1/$$

$$T - \sum_{P_i \in S_j} x_{ij} \geq 0 \qquad j=1,2,\ldots,n \qquad /2/$$

$$B_k T - \sum_{T_j \in D_k} \sum_{P_i \in S_j} x_{ij} \geq 0 \qquad k=1,2,\ldots,p \qquad /3/$$

$$\sum_{P_i \in S_j} x_{ij}/t_{ij} = 1 \qquad j=1,2,\ldots,n \qquad /4/$$

$$x_{ij} \geq 0 \quad \text{for all } P_i \in S_j, \quad j=1,2,\ldots,n \qquad /5/$$

Condition /1/ ensures that the active time of any one processor will not exceed T, condition /2/ - that each task is completed by time T, and condition /3/ - that the time of using any resource type will not exceed T. Condition /4/ ensures that each task is completed.

Solving the above LP problem we obtain the optimal values of x_{ij}, $P_i \in S_j$, $j=1,2,\ldots,n$, which minimize T. However, we do not know the task part start times which make the optimal schedule. Below we present an algorithm which constructs an optimal schedule in polynomial time.

THE SECOND STAGE

In the second stage, knowing the generating schedule, we shall construct the optimal schedule.

Let X denote the m×n matrix of nonnegative elements which are the optimal values of x_{ij}, $P_i \in S_j$, j=1,2,...,n, obtained in the first stage. Column j /task T_j/ of matrix X will be called critical if $\sum_{i=1}^{m} x_{ij} = T$. Similarly, resource R_k will be called critical if $B_k T = \sum_{T_j \in D_k} \sum_{i=1}^{m} x_{ij}$. Let us also define the m×m diagonal matrix Y of nonnegative processor idle times: $y_{ii} = T - \sum_{j=1}^{n} x_{ij}$, i=1,2,...,m. The columns of Y will represent dummy tasks which do not require additional resources. We shall denote by Z the m×(n+m) matrix composed of matrices X and Y as indicated below:

$$Z = \left[\boxed{X} \; \boxed{Y} \right]$$

Let us introduce the set NC, called the generating set, containing m positive elements of matrix Z which are:

- exactly one element in each critical column,
- exactly one element in each of B_k columns representing tasks requiring the critical resource R_k,
- no more than one element in the remaining rows and columns.

The resource requirements of tasks represented in NC cannot exceed the resource constraints, i.e.

$$\left| \left\{ T_j : \exists_i z_{ij} \in NC \wedge T_j \in D_k \right\} \right| \leq B_k, \quad k=1,2,...,p. \qquad /6/$$

For the set NC we have to calculate the parallel processing time DELTA of the task parts represented in NC.

The construction of the optimal schedule proceeds in the following way:

1° Find the generating set NC.

2° Calculate DELTA from the following formula

$$DELTA = \begin{cases} z_{min} & \text{if } T - z_{min} \geq z_{max}, \\ T - z_{max} & \text{otherwise,} \end{cases}$$

where $z_{min} = \min_{z_{ij} \in NC} [z_{ij}]$, $z_{max} = \max_{\substack{z_{ij} \notin NC \\ k}} \left[z_{ij}, \sum_{T_j \in D_k} \sum_{i=1}^{m} z_{ij}/B_k \right]$.

3° Decrease T and all $z_{ij} \in NC$ by DELTA. If T = 0 then end the procedure, otherwise go to step 1°.

It can be seen that set NC is constructed in such a way that at the end of each iteration the elements of matrix X, as well as T, fulfil conditions /1/ - /3/. Let us also note that for each set NC, DELTA is chosen such that either one of the positive elements in the matrix Z is reduced to zero, or one more column or resource type becomes critical. Each of these events may occur a finite number of times which ensures that the optimal schedule will be obtained in a finite number of iterations. However, in order to prove this, we have to demonstrate the existence of NC for each schedule fulfilling conditions /1/ - /3/ /in particular, for the generating schedule/.

THEOREM 1 For each schedule fulfilling conditions /1/ - /3/, $n \geq m$ and $T > 0$, there exists a generating set NC.

PROOF Let us construct an $(m+n) \times (m+n)$ matrix V as follows

$$V = \begin{bmatrix} X & Y \\ W & X^T \end{bmatrix}$$

where W is an $n \times n$ diagonal matrix of nonnegative elements:

$$w_{jj} = T - \sum_{i=1}^{m} x_{ij} \qquad j=1,2,\ldots,n.$$

As can be seen, each row sum and column sum of V is equal to T. Thus, in matrix $\frac{1}{T} V$, each row sum and column sum is equal to one. Since all elements of the square matrix $\frac{1}{T} V$ are nonnegative, this is a doubly stochastic matrix which is a convex combination of permutation matrices, as follows from the Birkhoff - von Neumann theorem [1]. It is evident that any one of the permutation matrices in such a convex combination can be identified with a generating set NC if it satisfies the resource constraints /6/ and contains B_k elements representing tasks $T_j \in D_k$, for each critical resource R_k. Condition /3/ ensures that at any time within the schedule length T, it is possible to find no more than B_k tasks which use resource R_k; thus, at least one of the permutation matrices in the above convex combination is identified with a generating set NC. ◇

Let us now pass to the problem of the bound on the number of preemptions in the optimal schedule. From the linear programming formulation posed in the first stage, follows that for $m > 2$, in the optimal basic feasible solution, there will be no more than $2n+m+p$ positive variables. In fact, there will be no more than $n+v_1+v_2+v_3$ positive variables, where v_1, v_2, v_3 are the numbers of inequalities /1/, /2/ and /3/ correspondingly, in which variables transforming them into equalities

are equal to zero. In other words, v_1 is the number of processors with zero idle time, v_2 is the number of critical tasks, and v_3 is the number of critical resources. But $1 \leqslant v_1 \leqslant m$, $0 \leqslant v_2 \leqslant m-1$ and $0 \leqslant v_3 \leqslant g$, where $g \leqslant m$ is the maximum number of resource types for which $\sum_k B_k \leqslant m$. Hence, in the optimal solution, there will be no more than $n+2m+g-1$ positive variables, one of which is T.

Thus, if $n > m$, there exists a generating schedule with no more than $n+2m+g-2$ positive x_{ij} values. If we could construct an optimal schedule without introducing additional preemptions, then the upper bound on the number of preemptions in the optimal schedule would be equal to $2m+g-2$. However, the second-stage algorithm generally introduces additional preemptions. We shall now establish an upper bound on this number.

First, let us make a certain modification to the matrix Z with the objective of reducing the number of preemptions in the optimal schedule. This modification is also beneficial for the running time of the algorithm for finding the generating set NC, which will be discussed later. The idea of this modification is to replace all the tasks /including the dummies/ using the same resource type, which are assigned to only one and the same processor in the generating schedule, by a new task. At the end of the second stage, we have to create a schedule for the original set of tasks by reassigning the time intervals DELTA obtained for the new tasks, to the tasks which they replaced.

THEOREM 2 The upper bound on the number of preemptions in the optimal schedule is equal to $2m^2-4m+m(g-1)+m(v_N-1)+2$, where $g \leqslant m$ and $v_N \leqslant \min[n-2m-g+2, mp]$.

PROOF The modified matrix Z' will contain v_0+v_N columns /tasks/ and no more than $v_0+v_N+v_1+v_2+v_3-1$ positive elements, where v_0 is the smallest number of original tasks and v_N - the maximum number of new tasks. Hence, $v_0 \leqslant v_1+v_2+v_3-1$, and v_N is bounded by $\min[n-v_0, mp]$. Since each iteration of the schedule construction procedure determines the parallel processing time of m task parts, the procedure will terminate with $[m\,(\text{number of iterations}) - (v_0+v_N)]$ preemptions. Hence, we may obtain a bound on the number of preemptions by bounding the number of iterations.

We already know that after each iteration, either one of the positive elements in the matrix Z' is reduced to zero, or one more column becomes critical, or finally, one more of the resource types becomes critical. Exactly m elements become zero in the last iteration. Thus, there will be no more than $v_0+v_1+v_2+v_3+v_N-m$ iterations of the first kind, $m-v_2$ iterations of the second kind, and $g-v_3$ iterations of

the third kind, hence at most $v_0+v_1+g+v_N$ iterations in all.

It is resonable to assume that no one of the new tasks is critical /if such a task exists, the problem can be reduced to m-1 processors/. Under this assumption, we can reduce the bound on the number of iterations by m. In the last iteration, m columns are critical, and some of them were critical at the beginning /having then at least two non-zero elements/. If a column becomes critical, and there exists exactly one positive element in that column, then at least one element of matrix Z' was reduced to zero in that iteration. If, however, the critical column has at least two positive elements, then in a certain iteration, when the number of positive elements in that column is reduced to one, at least two elements of matrix Z' are reduced to zero simultaneously. Thus, the total number of iterations is overestimated by at least m. It follows that the bound on the number of preemptions is equal to $m(v_0+v_1+g+v_N-m) - (v_0+v_N)$. Since $v_0 \leqslant 2m-2+g$, $v_1 \leqslant m$, and $n \geqslant m$, we obtain the thesis. ◇

Let us now pass to the description of the algorithm for finding the generating set NC. The algorithm makes use of the modified matrix Z'. Let us define a zero-one matrix A of the same size as matrix Z':

$$a_{ij} = \begin{cases} 1 & \text{if } z'_{ij} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

For simplicity of computation, it is better to find the set NC in matrix A. The elements of A selected for the set NC will be marked with the symbol Δ, to represent an assignment of the processor in that row to the task in that column. The block diagram of the algorithm is shown in Fig.1 and Fig.2.

This algorithm was programmed in Fortran for an ICL 1900 computer [10]. It may easily be shown that the presented algorithm finds the set NC in $O(v_N m^2)$ time.

EXAMPLE The following example steps through the algorithm for finding a generating set NC. The following fictitious data are given:

R_k:	R_1	R_2	R_1	R_2	R_1	R_2
	1	0	0	0	0	1
	0	1	0	1	1	1
A =	1	1	1	0	0	0
	0	1	1	0	0	0
	0	0	0	1	1	1
	*					*

* = critical column /task/

$B_1 = 3$

$B_2 = 2$

The construction of a generating set NC proceeds in the following way:

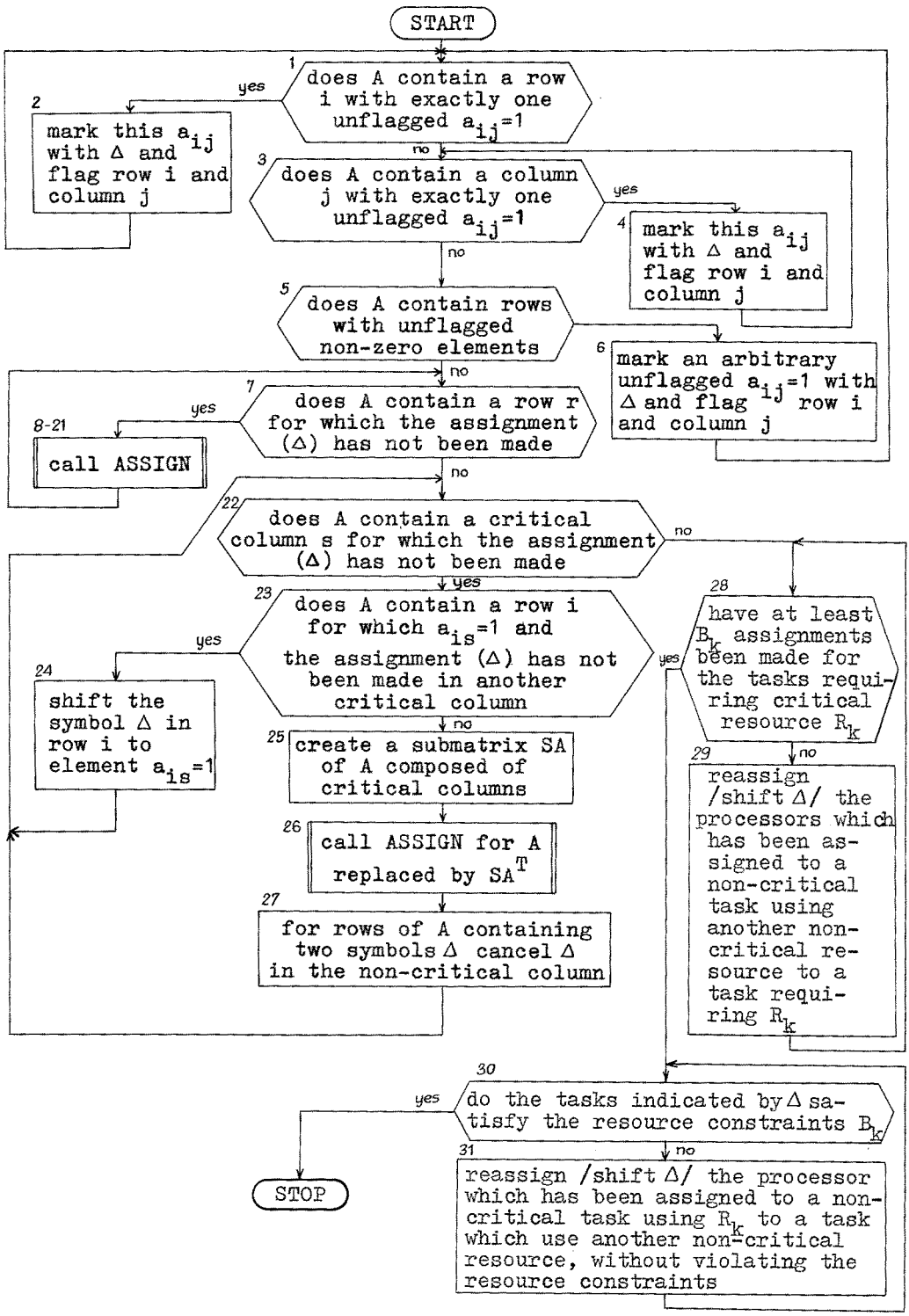


Fig. 1. The block diagram of the algorithm for finding a generating set NC.

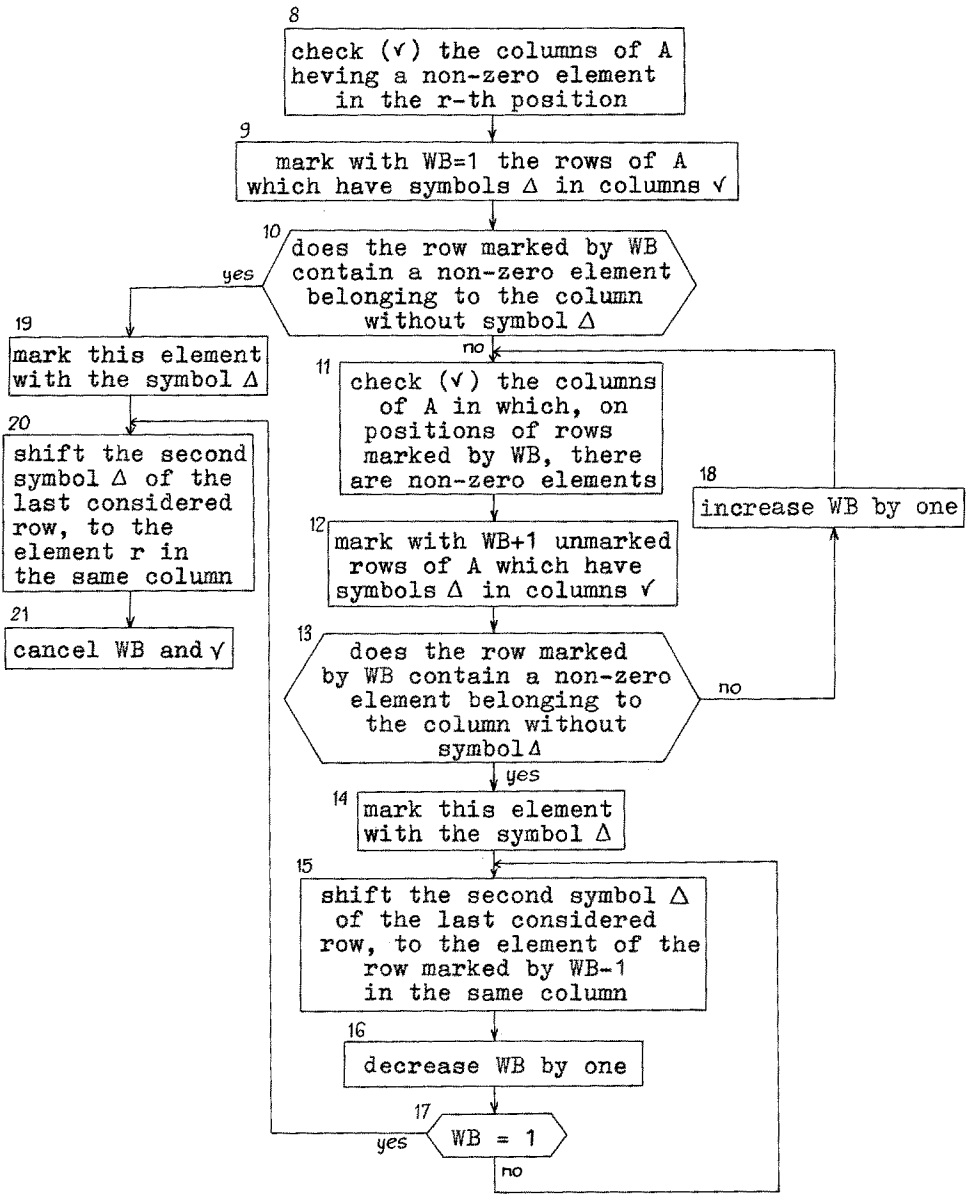
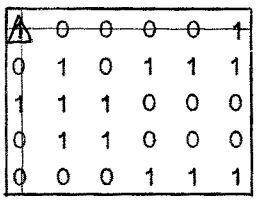
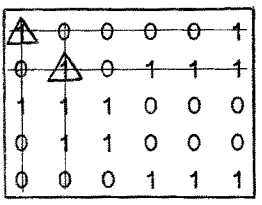


Fig. 2. The block diagram of procedure ASSIGN.

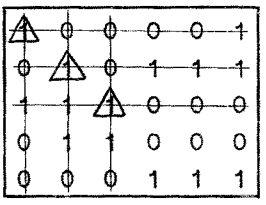
step: 1,3,5,6,1,3,5,



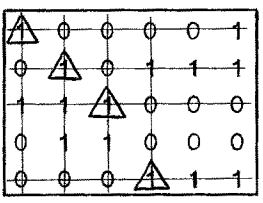
6,1,



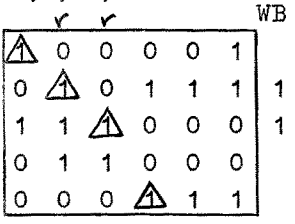
2,3,1,



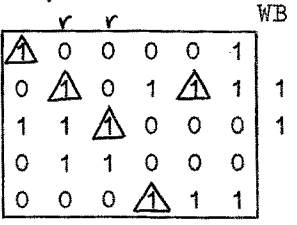
step: 4,1,3,5,7,



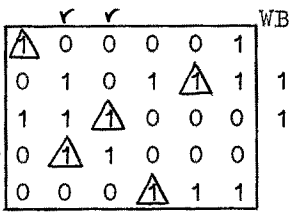
8,9,10,



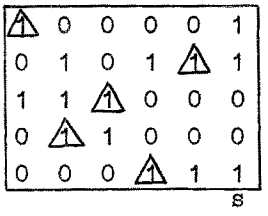
19,



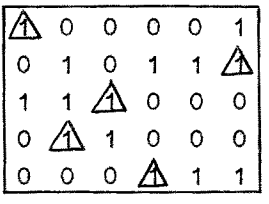
step: 20,21,7



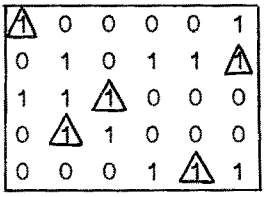
22,23,



24,22,28,30,



step: 31,30,stop.



Finally, $NC = \{z'_{11}, z'_{26}, z'_{33}, z'_{42}, z'_{55}\}$.

4. SCHEDULING DEPENDENT TASKS

Let us now consider the case where, in the task set $\mathcal{T}$ precedence constraints are given in the form described in Section 2.

Let F_l denote the set of all tasks which may be processed between the occurence of node l and l+1 in the precedence graph. Sets F_l, l= =1,2,...,N-1, will be called the main sets. Since the parts of tasks performed in particular main sets are independent, we may apply the approach proposed for the case of independent tasks to them. Let us introduce some additional definitions:

T^l - the schedule length for the set F_l,

G_j - the set of the numbers of main sets containing task T_j,

H_l - the set of processors which may process the tasks from F_l,
K_l - the set of resources required by tasks from F_l,
D_k^l - the set of tasks which use resource R_k in F_l,
$x_{ijl} \in \langle 0, t_{ij} \rangle$ - the total processing time of task T_j on processor P_i in the main set F_l; $P_i \in S_j$, $l \in G_j$, $j=1,2,\ldots,n$.

Keeping T for the schedule length, we obtain the following LP problem.

Minimize

$$T = \sum_{l=1}^{N-1} T^l \qquad /7/$$

subject to:

$$T^l - \sum_{T_j \in F_l} x_{ijl} \geq 0 \qquad \text{for all } P_i \in H_l, \quad l=1,2,\ldots,N-1 \qquad /8/$$

$$T^l - \sum_{P_i \in S_j} x_{ijl} \geq 0 \qquad \text{for all } T_j \in F_l, \quad l=1,2,\ldots,N-1 \qquad /9/$$

$$T^l B_k - \sum_{T_j \in D_k^l} \sum_{P_i \in S_j} x_{ijl} \geq 0 \qquad \text{for all } R_k \in K_l, \quad l=1,2,\ldots,N-1 \qquad /10/$$

$$\sum_{l \in G_j} \sum_{P_i \in S_j} x_{ijl}/t_{ij} = 1 \qquad j=1,2,\ldots,n \qquad /11/$$

$$x_{ijl} \geq 0 \qquad \text{for all } l \in G_j, \; P_i \in S_j, \qquad j=1,2,\ldots,n \qquad /12/$$

Conditions /8/ - /10/ correspond to conditions /1/ - /3/ for each F_l. As the solution of the above LP problem, we obtain a generating schedule analogous to that obtained in the first stage of the method described in Section 3. In order to construct the optimal schedule, the second-stage algorithm for independent tasks should be applied to each main set.

5. FINAL REMARKS

Scheduling problems, in which additional resource constraints are taken into account, are almost all NP-complete, and thus computationally intractable. Models of computer systems involving additional resources, however, are better adapted to practical situations and for this reason they are worth considering. In this paper one such problem has been examined, when tasks need a processor and one resource unit from the set of additional resources for their processing. This may be, for example, a situation in which a task requires a processor and an input/output device.

The method described finds an optimal schedule in two stages: in the first, an LP problem is solved, and in the second, which is polynomial in time, an optimal schedule is found. This method can also be applied

in the case of dependent tasks, when the set of nodes /events/ in the precedence " task-on-edge" graph is ordered.

REFERENCES

1. C. Berge: Théorie des graphes et ses applications, Chapt. X, Dunod, Paris 1958.
2. J. Błażewicz: Mean flow time scheduling under resource constraints, Preliminary Report PR-19/77, Technical University of Poznań /Poland/ March 1977.
3. J. Błażewicz, W. Cellary, R. Słowiński, J. Węglarz: Deterministic problems of scheduling tasks on parallel processors. Part I. Sets of independent tasks, /in Polish/ Podstawy Sterowania 6, June 1976, 155-178.
4. M.R. Garey, R.L. Graham: Bounds for multiprocessor scheduling with resource constraints, SIAM J. on Computing 4, 1975, 187-200.
5. M.R. Garey, R.L. Graham, D.S. Johnson, A.C.-C. Yao: Resource constrained scheduling as generalized bin packing, J. Combinatorial Theory Ser. A, 21, 1976, 257-298.
6. M.R. Garey, D.S. Johnson: Complexity results for multiprocessor scheduling under resource constraints, SIAM J. on Computing 4, 1975, 397-411.
7. R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan: Optimization and approximation in deterministic sequencing and scheduling: a survey, Report BW 82/77, Mathematisch Centrum, Amsterdam, October 1977.
8. K.L. Krause, V.Y. Shen, H.D. Schwetman: Analysis of several task-scheduling algorithms for a model of multiprogramming computer system, J.ACM 22, 1975, 522-550, 24, 1977, 527.
9. E.L. Lawler, J. Labetoulle: Scheduling of parallel machines with preemptions, Proc. of the IXth Internat. Symp. on Mathematical Programming, Budapest, August 1976 /to appear/.
10. R. Słowiński: Algorithm FEASCHE - computer program in Fortran, Preliminary Report PR 10/77, Technical University of Poznań,/Poland/ May 1977.