Prace Naukowe Wyższej Szkoły Pedagogicznej w Częstochowie
Matematyka IV, Częstochowa 1998

# The Principles of Creating the Critical Path of the Formation

*Adam Borowik, Henryk Piech*

*Abstract*

Assumptions and the algorithm for creating the critical path the formation of variables have been presented for programming procedures. The algorithm is based on the parallelization map and information related to the actions that are performed when calculating the values of variables in the particular phases of parallelization.

Also, information was used in the algorithm for the analysis, which related to the time of the execution of the particular actions, and each time the time of the operation of each node was determined. i.e. the time of the calculation of a variable in the particular phase of its modification. The critical paths created are used for the optimization of distribution in multi–processor systems with a limited number of conversion nodes.

## 1. Introduction

Algorithms for the conversion of matrix structures and algorithms containing hierarchically subordinated cycles have particularly great capabilities of processing parallelization.

Paralleization processes require sequential procedure lines to be built. These lines, permanently or periodically, can be executed in parallel. Minimalization of downtimes in the execution of the sequence of procedures is another task that can be called as the optimalization of processes distribution. This, however, is not optimization in the full sense of this word, but only an approximation to the optimum solution.

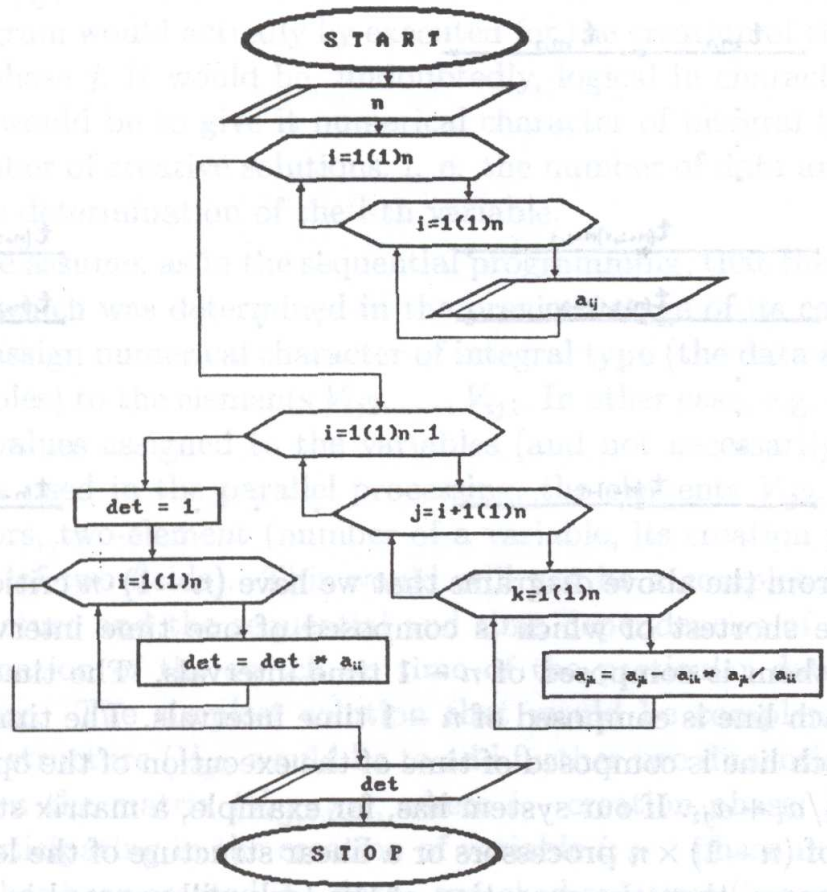
The optimization process can be executed more effeiently and quicker due to the proper isolation of procedures sequence lines, so called „critical lines" in such a manner that they also do not have gaps in the processing sequence [1]. Those gaps may lead to:

— downtimes of the processors,

— increase in the conversion time due to waiting for the creation of data.

By analyzing the algorithms that contain a large number of potentially parallel sequences of procedures, we can create sets of critical lines with the minimized sum time of waiting for the processing of the created variable.

Example 1. Calculation of a determinant. This algorithm is based on bringing the structure of square matrix elements into a triangular system (overdiagonal matrix). This algorithm is illustrated in Fig. 1.



By transposing the diagrams shown in the map into the lines of the creation of variables, we obtain:

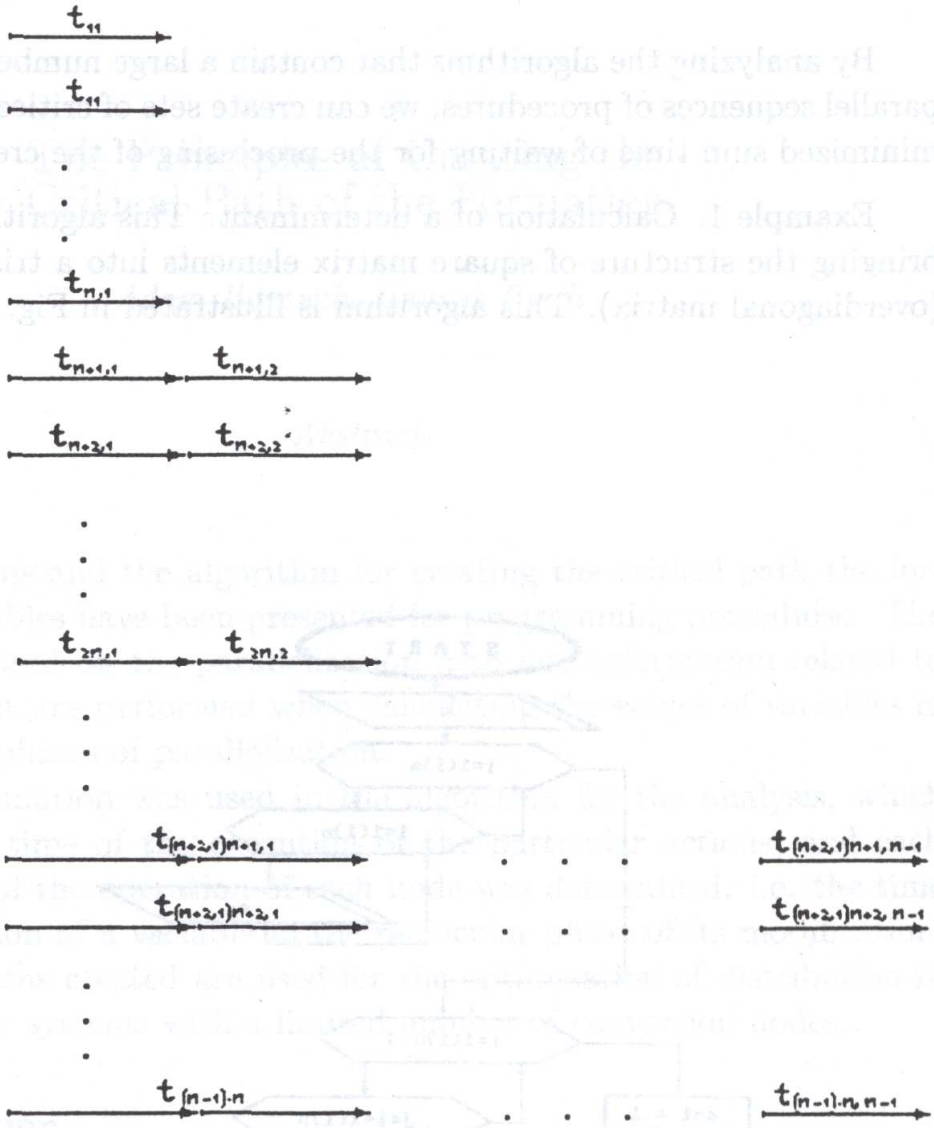



Fig.2

It results from the above diagrams that we have $(n-1)\cdot n$ critical lines to execute, the shortest of which is composed of one time interval, and the longest of them is composed of $n-1$ time intervals. The time of the execution of each line is composed of $n-1$ time intervals. The time of the execution of each line is composed of time of the execution of the operation $(*,/,+)a_{ik}\cdot a_{ji}/a_i + a_{ji}$. If our system has, for example, a matrix structure of dimensions of $(n-1)\times n$ processors or a linear structure of the length of $(n-1)n$ processors, then the execution of the task will proceed in $(n-1)$ phases. If we have a structure of a smaller number o processor, then we have to decide on which lines or their fragments will be executed first, and which as subsequent.

Dependencies that are sequential in charakter influence to a large extent the order in which the particular lines are executed.

In Fig. 3, we do not have a complete wiew of sequential dependecies and time information. In Figures 3a,b, and c, attempts are shovn to create manually the critical lines for the creation of a selected variable.

These illustrations are an accurate evidence for such (manual) approach to be inconvenient to carry out and does not give a complete information (concerning, for example, the time of the handling of particular stages). Also the final result is neither sufficiently clear nor convenient for program implementation (due to a high density of multiple crossing lines).

In Fig. 3d for example, there is no information concerning the data sources, i.e. the place where they have been created or read in.

The above drawbacks motivated us to introduce a two-dimensional tables, two of which would coincide with the co-ordinates in Fig. 2 (variable name, execution phase), andf the third one would be a vector of sequential dependencies. i. e. those on the elements $V_{ij2}, \ldots, V_{ij1}$, where $l$ is the maximum number of data used for the creation of one of the variables. The element $V_{ij1}$ would determine whether the data as previously calculated in the program would actually by executed for the creation of the i-th variable in the phase $j$. It would be, undoubtedly, logical in character, still better variant would be to give it numerical character of integral type indicating the number of creative solutions, i. e. the number of data and components used the determination of the i-th variable.

If we assume, as in the sequential programming, that the variable value is used which was determined in the previous stage of its conversion, then we can assign numerical character of integral type (the data as the numbers of variables) to the elements $V_{ij2}, \ldots, V_{ij1}$. In other case, e.g. assuming that snaller values assigned to the variables (and not necessarily the last one) could be used in the parallel processing, the elements $V_{ij2}, \ldots, V_{ij1}$ could be vectors, two-element (number of a variable, its creation phase) sets, or records (of two fields). This would still not be a complete description of the structure and the sequential and time dependencies of our data. The determination of the switch on time of the particular data $k(k = 2\ldots)$ is missing. The simplest solution that would be complementary to the existing structure ($V_{ijk}$ would be to add further two dimensions. We would have then the matrix $(V_{i,j,k,p,t})$, where $i$ – creation phase; $k$ – number of data participating in the creation of variable $i$; $p$ – phase in which the $k$ – th data has been created; $t$ – data „switched on" time (i. e. duration of the operation whose argument will be the data of number $k$). The zero value of parametr $p$ indicates that the recently determined value of the data $k$ will be used.

Generally, it can be assumed that the elements of this table will be numerical of real type. The program implementation requires, however,

that due to cost–saving reasons (in relation to storage occupancy) several tables be declared, which will have different type elements. Thus, declaration of the following tables can be proposed, which together fully describle the structure and the logic and time dependencies of our data:

$VA_{ij}$ – balance map ($i, j$ – as above): integer type;

$VB_{ijk}$ – able of logic dependencies (of $a$ „result–data" type); information on which data participate in the creation of the result: integer type;

$VC_{ijk}$ – table of phase complements, where the data $VB_{ijk}$ participating in the process $VA_{ijk}$ will be derived from the phase $VC_{ijk}$: integer type.

$VD_{ijk}$ – table of time structures, which defines the time of using the data $VB_{ijk}$ in the creation of the variable of number i in the pase $j$: real type.

If the elements of the table $VA_{ij}$ are non–zero, they will indicate the degree of the occupancy of the table $VB_{ijk}$, i.e. thet will show the number of data participating in the creation of the variable i in the phase $j$.

Before the algorithm was built, the following assumptions had been made and auxiliary variables taken:

$Max_{ij}$ – present part of the created line, which is characterized by the maximum time of its execution (part of the critical line);

$SM_{ij}$ – a set of numbers of variables which corresponds to the successive stages of the execution of lines of the maximum execution time $Max_{ij}$;

$SP_{ijk}$ – a set of numbers of variables which which corresponds to the successive stages of the execution of lines (except for the lines with the time parameter $Max_{ij}$). This line is characteristic in that it provides the $k$–th data for the creation of the variable with number i in the $j$–th phase. This is, at the same time, the ending of the creation line from the $k$–th variable side.

$Z_{ijk}$ – table of logic character, which indicates the places of ending the particular creation lines. If $k = 0$, then we have the critical line ($t = Max_{ij}$);

$R_{ij}$ – table of logic type branch.

Assumptions:

— tables $VA, VB, VC$, and $VD$ are introduced and this process will not be allowed for in the algorithm;

— table $Z$ is pre–zeroed.

The proposed algorithm can have the following form:

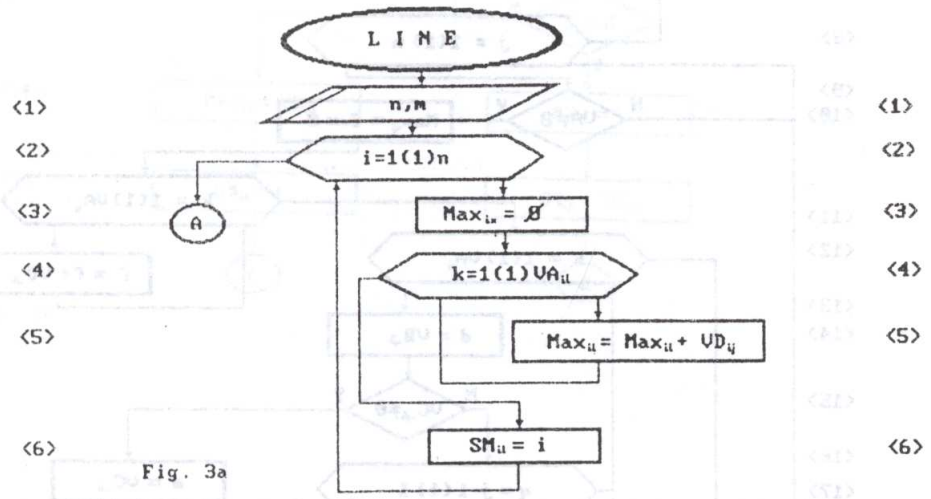



Fig. 3a

The above fragment of the algorithm refers to phase 1 of the execution of the task. We assume that the variables created in this phase are built exclusively on the basis of numberical values or the values of other variables whose initial values were read in during the initial (zero) phase.

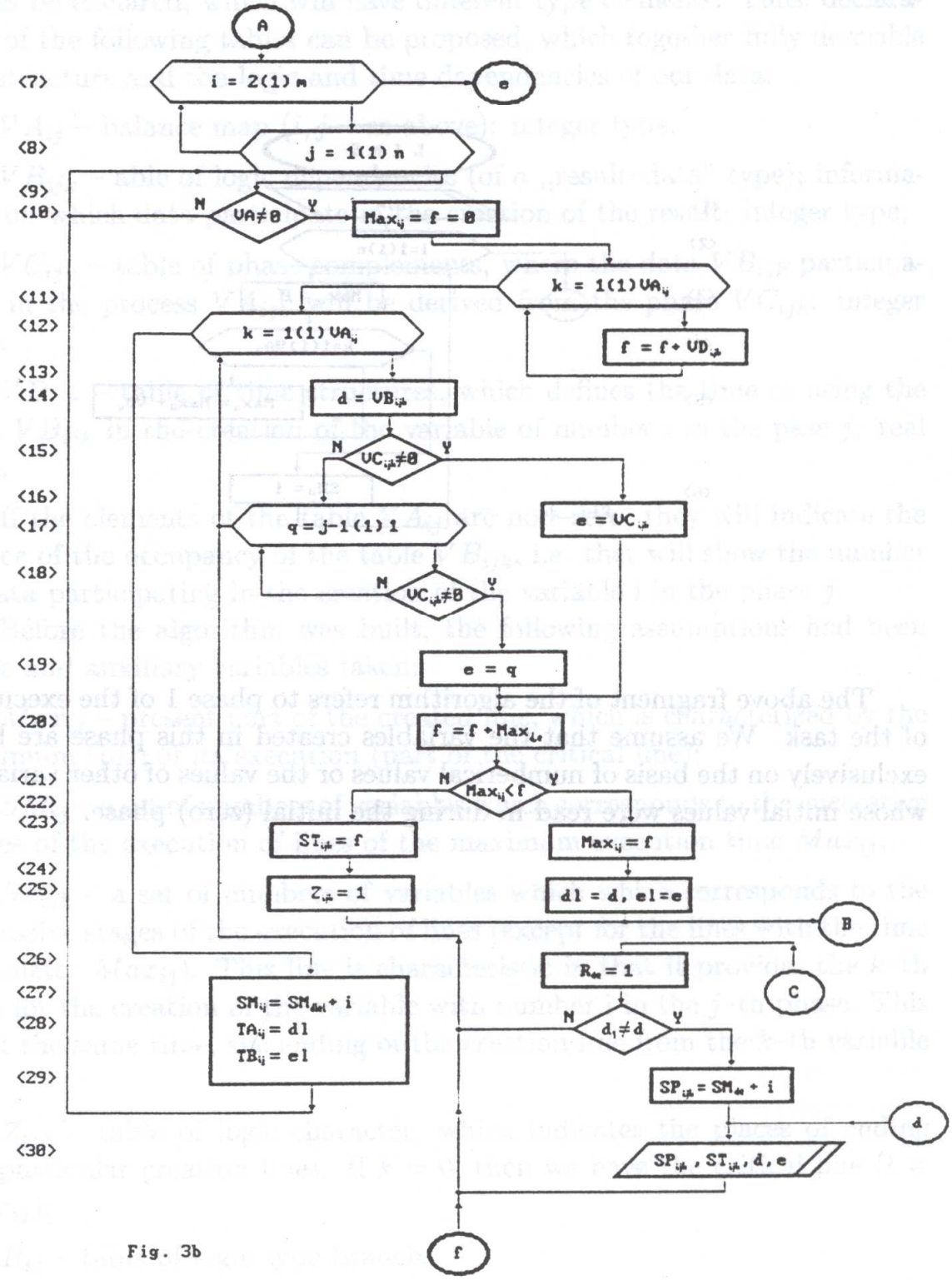

A
<7>
i = 2(1) m
e
<8>
j = 1(1) n
<9>
<10>
N
UA≠0
Y
Max_ij = f = 0
k = 1(1) UA_ij
<11>
<12>
k = 1(1) UA_ij
f = f + UD_ij
<13>
<14>
d = UB_ij
<15>
N
UC_ij≠0
Y
<16>
<17>
q = j-1(1) 1
e = UC_ij
<18>
N
UC_iq≠0
Y
<19>
e = q
<20>
f = f + Max_iq
<21>
N
Max_ij < f
Y
<22>
<23>
ST_ij = f
Max_ij = f
<24>
<25>
Z_ij = 1
dl = d, el = e
B
<26>
R_de = 1
C
<27>
<28>
SM_ij = SM_det + i
TA_ij = dl
TB_ij = el
N
d_i ≠ d
Y
<29>
SP_ij = SM_de + i
d
<30>
SP_ij , ST_ij , d , e
f

Fig. 3b

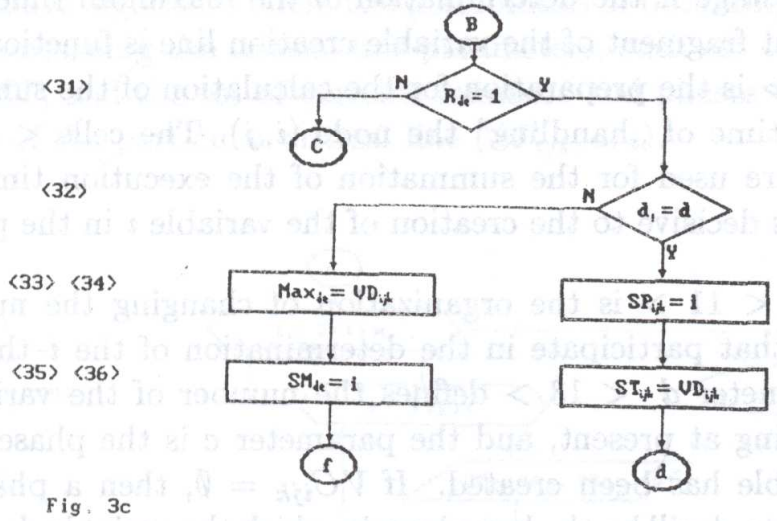



Fig. 3c

Algorithm description – principles for the creation of the critical line:

1. In the first phase, we use the parameters $n$ (number of created or argumented variables), $m$ (estimated number of phases in the execution of the parallelized process) $\rightarrow < 1 >$ .

2. In the row number „$i$", for each variable which is created in the 1–st phase, we determine the parameter $Max_{i1}$. In this case, this is the time of th creation of the $i$–th variable through the execution of $VA_{i1}$ operations. The execution time for each operation is defined by the parameter $VD_{i,1,k}$. The preparation for this task is described by the cells $< 2 >$ and $< 3 >$ . The determination of the summary time is provided by the cells $< 4 >, < 5 >$ . In the cell $< 6 >$, one of the methods is shown for the collecting of information about which variables will enter the set of the critical line. The element of this line set, which can be used in the successive phases is indicated by the variable $SM_{i1}$.

3. The next stage: $< 7 >, < 8 >$ is the organization of the analyzing procedure of the table $VA$, and depending on the values of its elements, the penetration and the analysis of the tables $VB, VC$, and $VD$ is performed.

4. If $VA_{ij} \neq \emptyset$, then its valued is indicated by the number of variables participating in the creation of the $i$–th variable in the $j$–th phase $< 9 >$ .

5. The next stage is the determination of the maximum time in which the present fragment of the variable creation line is functioning. The cell $< 10 >$ is the preparation for the calculation of the sum defining the total time of „handling" the node $(i, j)$. The cells $< 12 >$ and $< 14 >$ are used for the summation of the execution times for all operations decisive to the creation of the variable $i$ in the phase $j$.

6. The loop $< 11 >$ is the organization of changing the numbers of the data that participate in the determination of the $i$–th variable. The parameter $d$ $< 13 >$ defines the number of the variable that we are using at present, and the parameter $e$ is the phase in which this variable has been created. If $VC_{ijk} = \emptyset$, then a phase of the created data $d$ will be the last phase in which the variable $d$ underwent modification $< 15 > \ldots < 19 >$.

7. In the cell $< 20 >$, a correction of the total time of handling the node $(i, j)$ is made by the maximum time of the execution of the previous fragment of the critical line (without allowing for the node $(i, j)$).

8. The decision cell $< 21 >$ allows the elements of the line with the maximum execution time $< 23 >$ $< 25 >$ to be selected. The other elements will form the ending of the lines derived from the data that do not give the maximum execution time $< 22 >$ $< 24 >$.

9. In the cell $< 31 >$, we verify wheter a branch occurred previously or not in the node $(d, e)$ from which the present data comes, i. e. whether this node has already its continuation (an extension to the situation $R_{d,e} = 1$. If this node does not have its continuation, then it will obtain one since we use the data $d < 26 >$ in the node $(i, J)$.

10. The variables which we use in the node $(i, j)$, not giving the maximum time continuation $(d \neq d1, e \neq e1)$, achieve the ending for their line in this node $z_{iju} = 1 < 24 >$. Other parameters of the ended lines, i.e. the total time of the execution of the lines $ST_{i,j,u}$, the set of the numbers of the variables being the intermediate stages of the execution, which are at the same time the components of description of the nodes of a given line $SP_{ijk}$, will all be determined in the cells $< 22 >$ and $< 29 >$.

11. In the situation where the node $(d, e)$ has already been used by another line $(R_{d,e} = 1)$, a beginning of a new critical line will be formed in it with the maximum time run $(d_1 = d) < 33 >, < 35 >$ or with nonmaximum time run $< 34 >, < 36 >$.

12. The above algorithm offers the possibilities of obtaining the information concerning the critical line parameters, such as: total execution time $ST$, and the set of variable numbers and phases that make the nodes of the particular critical line $(SP_{ijk}, d, e)$.

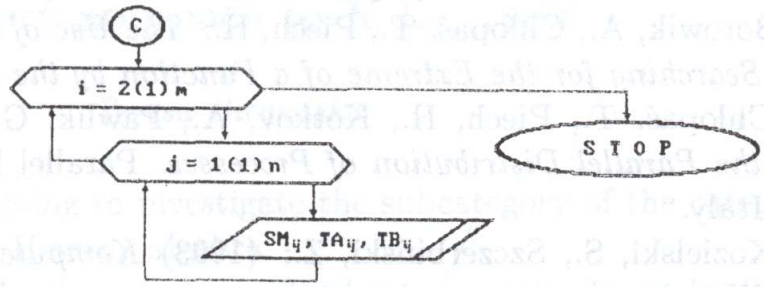



Fig. 3d

13. The remaining printouts related to the parameters od the critical lines with the maximum run time will be obtained in the stages $< 37 >$ $\ldots < 40 >$.

### Conclusions

1. The above presented algorithm connects the nodes that are characterized by the maximum time of the handing of each node.
2. The critical lines that create the variables of a given program constitute a information basis for the algorithm of the optimization of process distribution in multi–processor systems.
3. The distribution optimization criterion will be carried out through the selection of a critical line with the maximum time of creating the resulted variable.
4. The most evident results of the use of the critical lines with the maximum handling time in distribution optimization procedures are found in matrix calculus, stochastic conversion and in proving theorems [3].

## References

[1] Borowik, A., Chłopaś, T., Gubarieni, N., Piech, H.: *Verification of Theorem Truthfulness in Three-Valued logics with Parallel Systems Applications.* Parrallel Numerics'96, str. 73-85. Słowenia '96.

[2] Borowik, P.: (1995) *Przykładowy algorytm provera automatycznego dowodzenia twierdzeń.* Streszczenie wykładów i komunikatów. XXIV Ogólnopolska Konferencja Zastosowań Matematyki, Zakopane'95, 13-14.

[3] Borowik, A., Piech, H.: *Procedures of the Assignment of Tasks to the Processors in Parallel Systems.* The'96 Zakopane Conference.

[4] Borowik, A., Chłopaś, T., Piech, H.: *The Use of Parallel Structures for Searching for the Extreme of a Function by the Fastest Drop Method.*

[5] Chłopaś, T., Piech, H., Kotkov, A., Pawlik, G.: *On the Problem of the Parallel Distribution of Processes.* Parallel Numerics'95, sorrento, Italy.

[6] Kozielski, S., Szczerbiński, Z.: (1993) *Komputery równoległe.* WNT, Warszawa.