A FEW TRANSFORMATIONS AND COMMENTS CONCERNING A PROGRAMME, USED IN MANAGEMENT OF TIME FOR DIFFERENT PROJECTS

Rădescu Nicolae, Univ."Spiru Haret"-Craiova
Rădescu Octavian Dan, doctorand
Universitatea
Liberă Internațională Moldova-
Chişinău
nicolaeradescu@yahoo.com

Keywords: management of time projects, decomposition graph, graph programm

Abstract: In the study of time management it is settled a graphe of a work programme and starting from the order relations existent between different operations, then, the way of the critical path enabled the, lets say approximative deduction of the realization date of the works' ensemble, fluctuation intervals, etc. Actually the way of the critical path is not by itself a method of order. It is applied only for the already settled graphe programme, meaning in the moment when the order of things has been done. It's role actually is to interpretate the graph from a temporal point of view and to give all the indications needed concerning the realization data of different events of the programme, or to enable the testing of different graphs of the same ensemble of works.Next, using a descomposition of a graph, notion that belongs to us and for which we give the determinating algorithm, we will put in a linear way all the connections between the acitivities included in the programme, which offer a good partial analysis on pieces and then the global analysis of the programme given, using the corresponding programme graph.

Within "current" subjects not without interest, we meet more frequently the issue of project order, by which we will tangentially make an observation, or better said, a filling that is as many other more theoretical (that in the virtue of inertia it has reduced chances for practice, in fact not only this one) and namely related to the activities roads' default critical path graph of a programme defined by time order.

We intend to remind that a program is a lot of operations that lead to a goal (project, large ensemble, etc.), so that it is known for each operation its duration (fixed or random) and order relations on this (history required).

A graph of a programme or order is the graph (X, U) associated with this program defined by time order, in which the set is the operations of the U arches, duration of these operations or the time being alleged known (even random), we attach each arc of graph value greater or equal to zero for each defined each by time operators. In other words we define a function v: $U \rightarrow R_+$, which is known in literature as the value function; v(u), the u arc value. The X set of graph E_i, called events, can be interpreted as indicating partial achievement of objectives.

Setting the graph for each operation requires knowledge of the operations which precede immediately. The graph translates this way the order relations in the set of operations; it is a representation of what some call order (other authors call order the program itself). For reasons easy to understand, assume that the graph of a program is without circuits.

Representing a program using a graph can be made in another way and say: we will not be in a vertex event, but a determined operation, and we use an arc to represent the order in which the two operations Q_{ij} and Q_{jk} follow one another so that it is possible to give a reprieve from starting his Q_{ij} and that of Q_{jk}. In this representation it is convenient naming tasks and operations considered before restrictions arches that are affected by the interruption or delay.

Considering a graph G = (X, U) or G = (X, F) where $X=\{x_1,x_2, ... ,x_n\}$ noted many graph vertex with $|X| \geq 2$ (at least two vertices), U being the multitude of arcs (x_i, x_j) and the multi-value graph F $X \rightarrow X$; in addition, we suppose that the graph G has no loops, circuits and $F^{-1}x_1=\Phi$ (in x_1 enter are no arcs) and $Fx_n=\Phi$ (from x_n exit no arcs).

We call the decomposition graph G vertexes between x_1 and x_n, graph G 'obtained from G so that x_1 and x_n vertexs belong to the graph G' and in addition to any vertex t of the graph G' to have fulfilled the condition: $| |F^{-1}t| \leq 1, t \neq x_n$ (ie on top of t G 'to one arc).

To determine the graph G ', proceed:

1. If there is a vertex x_i in the graph G so that $Fx_i=\Phi$ and $F^{-1}x_i=\Phi$, x_i isolated point, then the algorithm continues to da 4, leaving the graph G' this vertex unchanged, otherwise, go to 2;

2. x_i being a vertex in the graph G so that $F^{-1}x_i=\{x_{j_1}, x_{j_2},..., x_{j_p}\}$, and $Fx_i=\{x_{k_1}, x_{k_2},..., x_{k_r}\}$ meaning that in the vertex x_i at least two arcs and $|Fx_i|>0$ the image vertex x_i is not vidă.

În this case, enter the graph noted G_i' lots consisting of p vertices $\{x_{i1}^{'}, x_{i2}^{'},..., x_{ip}^{'}\}$, xi which replaced the tip and at the same time we introduce the corresponding graph G_i' crowd of arches

$\{(x_{j_1}, x_{i1}^{'}),(x_{j_2}, x_{i2}^{'}),...,(x_{j_p}, x_{ip}^{'})\}$ replaced in the graph G_i' the arcs of G

$\{(x_{j_1}, x_i),(x_{j_2}, x_i),...,(x_{j_p}, x_i)\}$;

Archs $\{(x_i, x_{k_1}),(x_i, x_{k_2}),...,(x_i, x_{k_r})\}$

and many are replaced by $\{(x_{ih}^{'}, x_{k_1}),(x_{ih}^{'}, x_{k_2}),....,(x_{ih}^{'}, x_{k_r})\}$, h=1,2, ... ,p. The rest of the vertex and Archs graph G remains unchanged G_i'

3. If a vertex x_i in the graph G is such that when $Fx_i=\{x_{k_1}, x_{k_2},..., x_{k_r}\} \neq \Phi$ and$F^{-1}x_i=\Phi$ in graph G_i' consider the vertexs $\{x_i, x_{k_1},..., x_{k_r}\}$ and arcs corresponding $(x_i, x_{k_1}),...,(x_i, x_{k_1})$ of G, and the rest of the vertex and Archs graph G, will still keep in graph G_i'.

4. If there exists a graph G vertex x_i, $x_i \neq x_n$, $F^{-1}x_i=\{x_{j_1},..., x_{j_r}\} \neq \Phi$ and $Fx_i=\Phi$, so when this situation properly, the graph G_i' we consider instead the vertex x_i, many of vertices with many $\{x_{i1}^{'}, x_{i2}^{'},..., x_{ip}^{'}\}$, with $\{(x_{j_1}, x_{i1}^{'})...,(x_{j_p}, x_{ip}^{'})\}$ Archs who replaced crowd $\{(x_{j_1}, x_i),...,(x_{j_p}, x_i)\}$

5. If during the application of the algorithm meet the vertices x_i graph G characterized by the steps 1-4, is built properly, in each case grafu G_i' algorithm and resumes for the graph from step 1.

a) if $I \neq n$, we assign I: = I +1 algorithm and resumes for the new graph from step 1;

b) if I = n, n being the index vertex x 1n with $Fx_n=\Phi$, then the algorithm is stopped; graph G_n' is thus obtained graph G', the graph decomposition of G with the property required.

If we are interested to include paths between the tip and the tip x_1 and vertex x_n partial or paths between the vertex x_1 and the x_i vertex of the initial graph G, then browse arcela sequence graph G'=(X',F') using the scrolling is $F'^{-1}:X' \rightarrow X'$,those arcs whose end we are interested in the final vertex in the arrival route. If the graph G ', we determine the shape of a path:

$(x_1,x_{hp}, \ldots ,x_{kb},x_{ja},x_i)$ then giving up the second show, we can obtain the appropriate initial graph G, a path $(x_1,x_h, \ldots ,x_k,x_j,x_i)$ from vertex x_1 to vertex x_i.

Using graph decomposition program, we can put the record in a linear linkages between all activities included in the program or achieving partial, which provides a good analysis pieces on the partial and global at the same time allow the introduction of new restrictions on u program, or changing the sequence relations between tasks, simply by adding arches, but without changing the program complete graph.

Whether fig.1,

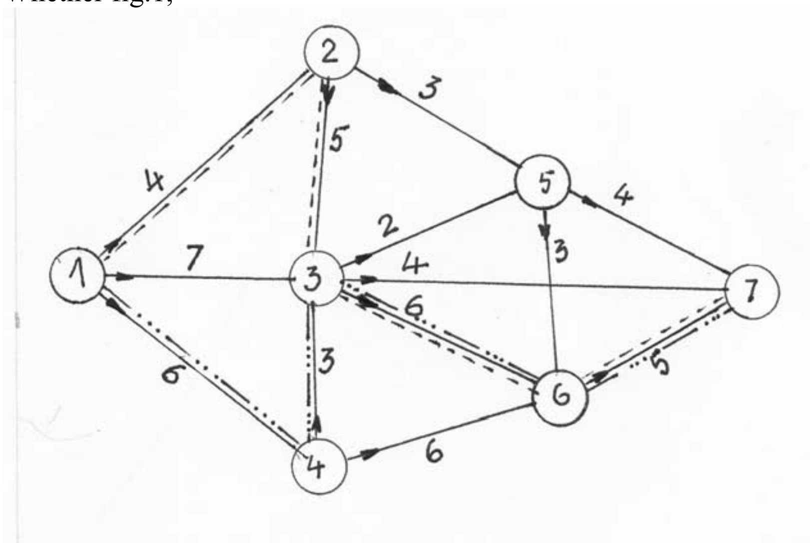

Fig.1

graph G of the past are dependencies between different activities and value function v: U → R_+ (or even defining v: U → Z_+) duration of time defined for each arc a value that is entered on the arc . In Fig.2, graph G_3' represent the partial conduct of the program graph G corresponding vertex x_3 using algorithm presented, with the remark that the values arcs introduced in, would be defined according to the arcs introduced so :

$$v(x_{j_1},x_{i1}^{'})=v(x_{j_1},x_i),v(x_{j_2},x_{i2}^{'})=v(x_{j_2},x_i),\ldots,v(x_{j_p},x_{ip}^{'})=v(x_{j_p},x_i)$$

$$v(x_{ih}^{'},x_{k_1})=v(x_i,x_{k_1}),v(x_{ih}^{'},x_{k_2})=v(x_i,x_{k_2}),\ldots,v(x_{ih}^{'},x_{k_r})=v(x_i.x_{k_r})$$

In the fig.2 $\Rightarrow$ the paths of graph G_3':

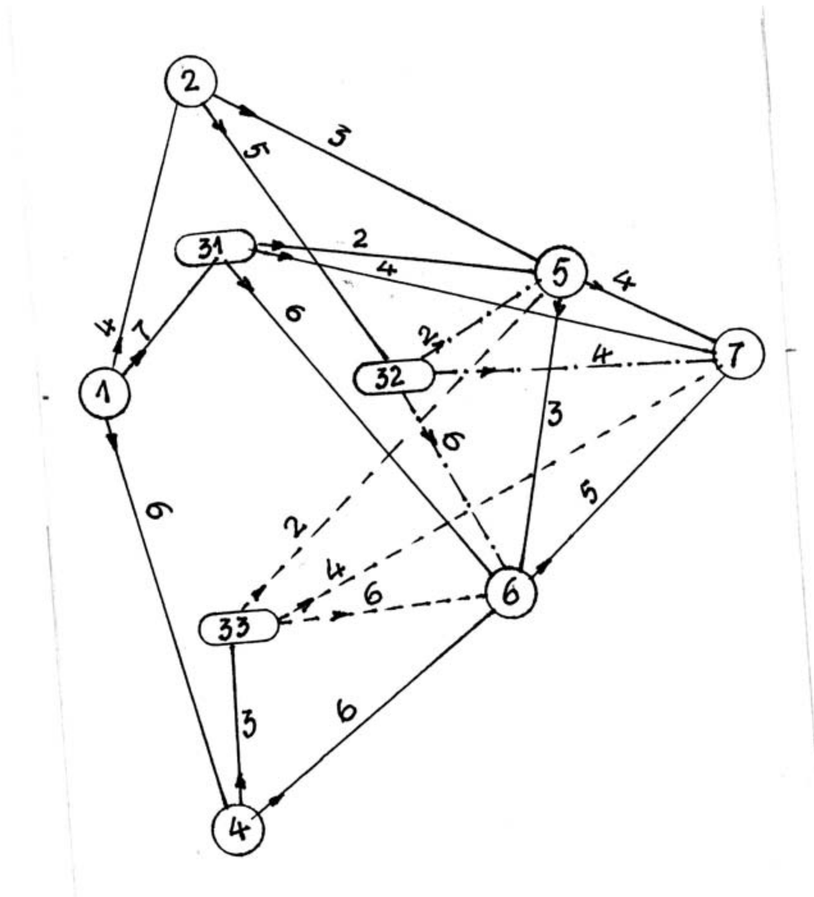

Fig.2

(1,2,5,7), (1,2,(32),5,7), (!,2,5,6,7), (1,2,(32),7), (1,2,(32),6,7)-critical path,

(1,4,(33),5,6,7), (1,4,(33),6,7)-critical path,....

The pats in the graph G:

(1,2,5,7), (1,2,3,5,7), (1,2,5,6,7), (1,2,3,7),... and (1,2,3,6,7), (1,4,3,6,7)- the critical paths.

If the program graph G' program decomposition graph G (which "apparently" looks more complicated than the program graph G) is useful in terms of correlations between the activities included in the program with the possibility of introducing new restrictions in the program or eliminate the consequences of other restrictions others remain confident that we can say that practice alone can give the evidence presented.

Then having the program graph G' associated with the program graph, we can easily calculate if a program graph G with more edges and arcs (as in practice) for example, during the first (start) of an S, while the later admissible (ST) for starting a business and on that basis peroade time that can be deferred or delayed activity, without disrupting other activities.

Graph G', , G decomposition graph program, they enjoy that $|F^{-1}t| \leq 1$(ie the top of the G' t enter one arc), for any vertex t, t $\neq x_n$ the graph, we can apply specialized computer programs, using diagrams where a vertex is the end (or beginning) of a single real activities.

In Figure 3,

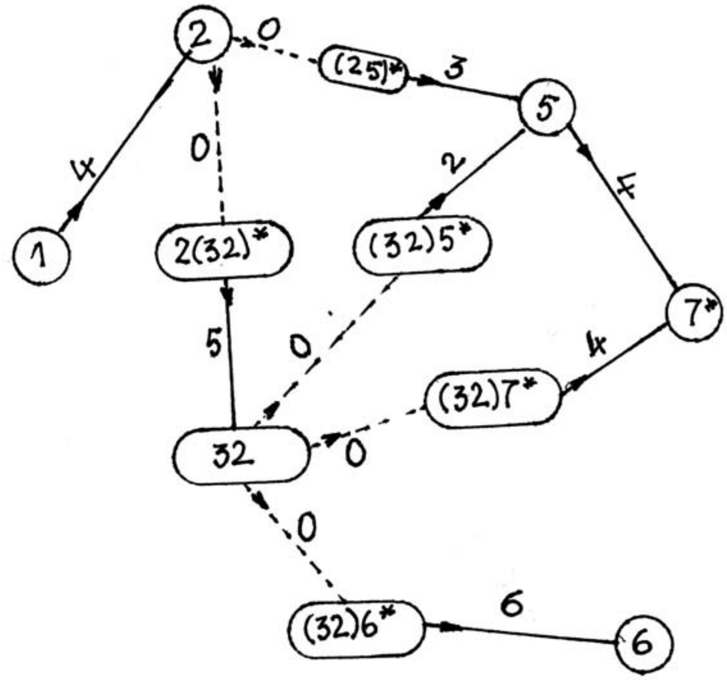

Fig.3

the graph G' were introduced fictitious vertexs: (25)*. 2(32)*, (32)5*, (32)7*, (32)6* corresponding to the vertex x, |Fx| ≥ 2, activities fictitious vare for u, v(u)=0, obtaining apparently satisfied provided that each vertex represents the beginning of a single activity. Each path of the graph G' transformed thus: (1,2,(25)*,5,7), (1,2,(2(32))*,(32),7), (1,2,(2(32))*,(32),5,7) obtain a path in the original program graph G as I mentioned, apart from a vertex any dummy inserted: (1,2,4,7), (1,2,3,/), (1,2,3,5,7).

Without going into details, it is better when meeting an operation composed by elementary operations to try and secure it is possible to simplify the representation. Decomposition of a programme can be sure to mark these composed operations, and the following example illustrates how to do this simplification operation.

Considering an operation composed of three elementary operation fig.4,

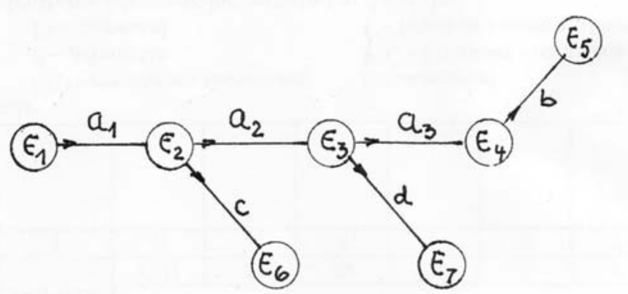

Fig.4

so that the c operation is after a_1, d is after a_2 and b is after a_3. In fig.5,

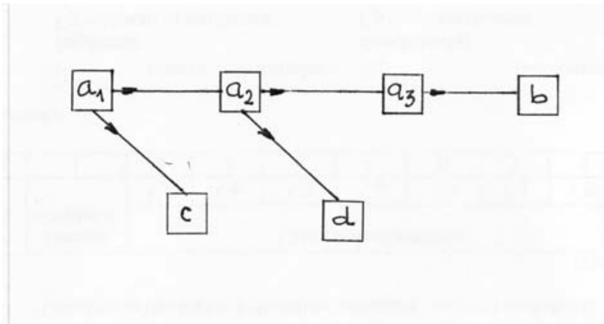

Fig.5

we show the representation by schedules and restrictions.

Thus, it is possible to simplify the represintion (fig.6); the new restrictions are easy to calculate.

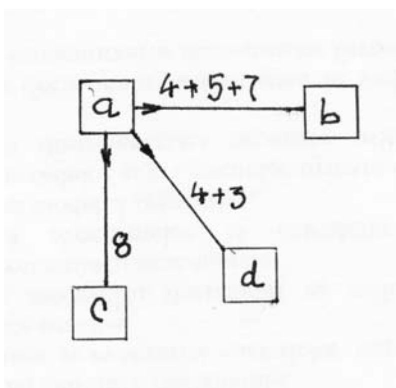

Fig.6

REFERENCES

1. **Berge C.,** Théorie des graphes et ses applications, Dunod, Paris,1967
2. **Buşe L., Siminică M., Marcu N.,** Analiza economico-financiară, Editura Scrisul Românesc, Craiova, 2003
3. **Constantinescu D., Nistorescu T.,** Managementul proiectelor, Editura SITECH, Craiova, 2008
4. **Iliescu V., Gherghinescu O.,** Managementul proiectelor, Editura didactică şi pedagogică R.A, Bucureşti, 2005
5. **Kaufman A., Desbazeille G.,** La méthode du chemin critique, Dunod, Paris, 1969
6. **Rădescu N., Rădescu Eugenia**, Probleme de teoria grafurilor, Editura Scrisul Românesc, Craiova,1982
7. **Rădescu N., Rădescu E.,** Determinarea unui flux de valoare minimă într-o rețea de transport redusă folosind o desfacere a rețelei, Lucr.V Simpozion "Modelarea cibernetică a proceselor de producție", 14-16 iunie1984,Vol.I, Bucureşti