ANALELE UNIVERSITĂŢII
"EFTIMIE MURGU" REŞIŢA
ANUL XII, NR. 1, 2005, ISSN 1453-7394

Marius Groza, Constantin Bărbulescu, Gabriel Limbean

Construction and Assembling Optimization of Power Equipment

The main purpose of this paper is to elaborate a calculation program in Pascal language, using Delphi environment. This calculation program is designed to solve the power engineering optimization problems using the critical path method. For illustrating the use of the algorithm and the calculation program we propose an application from power engineering: a 400 kV electrical overhead line section realization. This paper is structured in 4 parts. In the first part of the paper we present the application as a problem of critic path. In the second part of the paper, we determine the critic path in a program graph and time reserves. In the third part of the paper we present a representative numerical application. In the fourth part of the paper it is described the calculation program.

1. Presenting the problem

We consider that a plant specialized in constructing and assembling of power equipment has to realize a 400 kV electrical overhead line section. We know the component operations that need to be executed in order to realize the final objective, their succession and conditioning, the duration and cost of each operation, the necessary workforce and the equipment. We are required to determine the total duration of executing the line, as well as the optimization from the cost, duration, necessary workforce and equipment point of view.

The problem may be generalized for any "activity program", by program we understand the set of operations that lead to the accomplishing of an objective. The example stated makes reference to a production program; the methods presented being also valid for all the development, respectively research programs.

The classical solution for these problems is found via the schedule graph (the Gantt method), which presents a series of shortcomings: it does not highlight clearly the interdependencies between operations, it does not explain the temporal coincidences, it does not indicate the alternatives of declaring the various

operations, it does not have a rigorous mathematical foundation, it does not allow any studies of optimization.

All the shortcomings are cleared when using the representation of the program via a graph, where the values of the arcs are the durations of the component operations, and applying the critical path method (CPM), which mainly consists in *determining a path of a maximum value* between two peaks of that graph. Using CPM offers a great number of advantages: it offers a clear image of the evolution in time of the program, it allows the decreasing of the total duration of realization for the program without condensing the component operation, it highlights the operations that directly determine the duration of realization of the program, as well as those that allow the redistribution of the resources and the reduction of cost; it offers the alternative of rapid evaluation of the consequences of certain delays in realizing certain operations (without totally rebuilding the graph and the calculations); it can be easily implemented on a computer, having a solid mathematical base.

Applying CPM and optimizing the program is achieved via the following steps:

a) setting the list of the component operations, of their characteristics and representing the program via graphs;
b) setting the graphs in order;
c) determining the critical path (CP) and its value;
d) calculating the time stocks related to the realization of the component operations;
e) optimizing the program as from the length, the cost and the necessary resources points of view.

2. Determining the critical path in a program graph. Related problems

2.1. Defining the critical path

After representing the program of activities in a program graph and allocating values to the arcs (the durations of realization of the elementary operations involved), we raise the problem of determining the total duration of realization of the program. Obviously, this duration cannot be lower than the sum of the total duration of realization of the operations that compose the most unfavorable path from the initial event E_1 to the final event E_n. This path (or these paths) is called *the critical path* (CP).

The CP in a program graph is path of maximum value between the peak corresponding to the initial event and the one corresponding to the final event. The events and the operations corresponding to the peaks, respectively the arcs, that compose the CP are called critical event and critical operations, and the other events and operations are called non-critical.

The CP and its value are determined using algorithms of obtaining the paths with maximum value between two peaks of a graph: the Ford algorithm an the Bellman-Kalaba algorithm. Both algorithms are based on the optimality principle of Bellman, which is the foundation of the dynamic programming, the CP being undoubtedly composed of elementary paths (of smaller length) of maximum value.

2.2. Methods for determining the critical path

- ***The Ford algorithm***

We consider the program graph, connected and without circuits, $G = G(X,F)$, with n peaks, x_1 and x_n being the initial, respectively the final peak. To each arc $(x_i, x_j) \in A$ we assign a value $v_{ij} = t_{ij}$ (the duration of realization for the involved operation). We assume that the graph G is in order, presenting alternatives according to the Ford algorithm.

In order to determine the critical path, we have to go through the following steps:

a) to each peak x_i we assign a value $\lambda_i = t_i$, initialized on 0 (because we are looking for a positive maximum);

b) on step k, $k = \overline{2,n}$, we determine λ_k via the relation

$$\lambda_i = \underset{i}{Max}(\lambda_i + t_{ik}); \quad (x_i, x_k) \in A \tag{1}$$

λ_i being certainly non-zero, because $i < k$;

c) step b) is repeated n-1 times, until we determine λ_n, which is the value of the critical path;

d) we determine the peak x_{d_l} for which the following relation is observed:

$$\lambda_n - \lambda_{d_l} = t_{d_l n} \tag{2}$$

storing the peak x_{d_l} as the critical path;

e) on step l of the search in reversed order we determine a new peak on the CP, x_{d_l}, as being the one that complies with the following relation

$$\lambda_{d_{l-1}} - \lambda_{d_l} = t_{d_l d_{l-1}} \tag{3}$$

f) step e) is repeated until $x_{d_l} = x_1$, the CP being

$$DC = \{x_1 \quad x_{d_{l-1}} \quad ..... \quad x_{d_2} \quad x_{d_1} \quad x_n\} \tag{4}$$

In a program graph, there can be one or more CP. If there are more than one CP, the condition of maximum from the relation

$$\lambda_i = \underset{i}{Max}(\lambda_i + t_{ik}), (x_i, x_k) \in A \quad (5)$$

for a certain peak f x_k is complied with by more peaks x_i, respectively on one or more steps more peaks $x_{d_{k+1}}$ comply with the relation

$$\lambda_{d_{l-1}} - \lambda_{d_l} = t_{d_l d_{l-1}} \quad (6)$$

Then we separately analyze each solution, resulting more CP.

If we apply the Ford algorithm to a program graph $t_i = \lambda_i$ represents the natural time (expected) of realization for the event E_i: the minimum time that has to run from the beginning of the program until the moment when the E_i event takes place, in order that all the previous operations can be realized in the programmed durations.

- ***The Bellman-Kalaba algorithm***

It is a variant of the previous algorithm, using the techniques of dynamic programming. It is based on the following property, which is the particularization of the Bellman principle: any path of maximum value of r length is composed of elementary paths of $k(k \leq r)$ length and maximum value.

The matrix $[T]$ of the values (operating times) is defined in the following way: t_{ij} represents the duration of realization of the operation between E_i and E_j, if $(E_i E_j) \in A$, $i \neq j$, respectively $-\infty$ if $(E_i E_j) \notin A$, $i \neq j$, whereas for $i = j$ it is null. If the *graph is in a certain* order then all the elements in the *lower* triangle of $[T]$ are $-\infty$.

$$[T] = \begin{cases} t_{ij}; (E_i E_j) \in A, i \neq j; \\ -\infty; (E_i E_j) \notin A, i \neq j \\ 0; i = j \end{cases} \quad (7)$$

Based on the statements above, the steps of the Bellman-Kalaba algorithm are as follows, the algorithm consisting mainly of determining the maximal elementary paths of length 1, 2, 3, ... at most, between a certain peak of the graph and the final peak:

a) on the first step we determine the elements of the auxiliary vector v^1, representing the value of the paths of 1 length from E_i to E_n:

$$v_i^1 = t_{in}, i = \overline{1,n} \quad (8)$$

b) on a certain step k we determine the elements of the auxiliary vector v^k, repre-senting the value of the maximum paths of k length at most, between E_i and E_n:

$$v_i^k = \underset{j}{MAX}(t_{ij} + v_j^{k-1}), i = \overline{1,n} \quad (9)$$

c) the calculation is through when the following relation is complied with.

$$v_i^k = v_i^{k-1}, i = \overline{1,n} \quad (10)$$

d) CP has the value v_1^k, in the hypothesis that the condition on step c) is complied with;

e) we determine the peak x_{d_1} for which the following relation is observed

$$v_1^k - v_{d_1}^k = t_{1d_1} \quad (11)$$

storing the x_{d_1} peak as the CP;

f) on step l of the search we determine a new peak of the CP, x_{d_l} as being the one that complies with the relation

$$v_{d_{l-1}} - v_{d_l} = t_{d_{l-1}d_l} \quad (12)$$

g) step f) is repeated until $x_{d_l} = x_n$, CP being

$$DC = \{x_1 \quad x_{d_1} \quad x_{d_2} \quad ..... \quad x_{d_{l-1}} \quad x_n\} \quad (13)$$

2.3. Problems related to the method of the critical path

- ***Intervals of fluctuation***

We have determined for each event E_i the natural duration (expected) of realization, noting it with t_i, as the value of the maximal path from the initial event E_1 to the event E_i. We now raise the problem is the duration of realization of a E_k event cannot be increased, without altering the duration t_n of realization of the entire program (the CP value), therefore without disrupting the subsequent operations.

Thus we define the limit duration t_i^* of realization of the E_i event as the latest moment on which the event E_i may take place without disrupting the completion of the program (without altering t_n).

Considering that the program graph is in a certain order, obviously $t_n^* = t_n$, the other limit duration being determined via the relation:

$$t_i^* = \underset{j}{MIN}(t_j^* - t_{ij}), (E_i E_j) \in A, i = \overline{1,n} \quad (14)$$

The fluctuation interval Δt_i is defined as the delay allowed in the realization of the event t_i, without affecting the value of the CP , and it may be calculated via the relation:

$$\Delta t_i = t_i^* - t_i \quad (15)$$

For the critical events $\Delta t_i = 0$, and for those that are not critical $\Delta t_i > 0$.

- ***Operation margins***

Considering $t_1 = 0$, the operation represented by the E_iE_j arc cannot begin before the time t_i from the beginning of the program, respectively it cannot be completed before the time t_i^*, if want to avoid the perturbation of the program. The operation $E_i E_j$ disposes of the time $t_i^* - t_i$, from which it is programmed to use t_{ij}. If $t_{ij} < t_i^* - t_i$ then there is a time stock, which is called the total margin of the operation $E_i E_j$, and it is noted with mt_{ij}, defined by the relation:

$$mt_{ij} = t_i^* - t_i - t_{ij} \quad (16)$$

The total margin of an operation may be used as follows:

- the duration of the operation may be augmented, allowing it to partially or totally consume its margin;
- we preserve the duration of the operation, but we delay its beginning, partially or totally using the margin;
- we fractionate the operation, interposing breaks, which, added, may not exceed the margin.

By using integrally the total margin of an operation, we decrease the margins of the previous operations, as well as those subsequent. Consequently, in practice there are used other two types of margins: the free margin and the safe margin.

The free margin ml_{ij} is defined via the following relation

$$ml_{ij} = t_j - t_i - t_{ij} \quad (17)$$

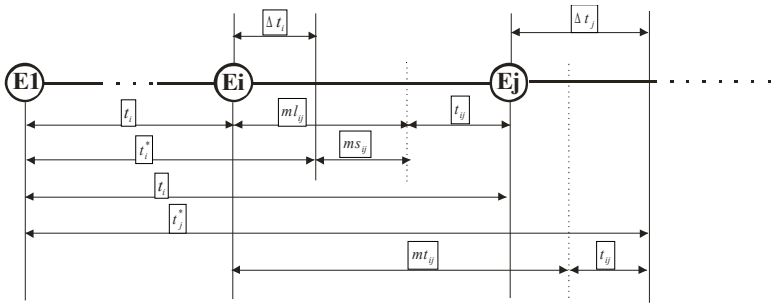


Figure 1.

representing the maximum duration with which it can be delayed, from the natural duration of realization of the event t_i, the beginning of the operation E_iE_j, without disrupting the natural duration of realization of the event E_i, therefore without altering the margins of the subsequent operations.

The free margin may be interpreted as that part of the total margin which can be used to augment of delay the operation E_iE_j without altering the fluctuation interval of the terminal event E_j:

$$mt_{ij} - m_{ij} = t_i^* - t_i = \Delta t_j \geq 0 \qquad (18)$$

From the previous relations, it results:

$$mt_{ij} \geq ml_{ij} \geq ms_{ij} \qquad (19)$$

ml and mt being unquestionably non-zero.

Obviously, all the operations that are part of the CP have non-zero margins, because they do not allow any delay without altering the program total duration of realization.

3. Numerical application

For the example stated at the beginning of this paper, the list of the main operations is given in the following table (table 1), where we have also filled in the duration of each operation (in days) and the order relations between operation has been highlighted. We have considered only on expansion panel (the entire LEA will be handled analogously).

❑ *The Ford algorithm*

The results obtained with the Ford algorithm are shown in the next graphic (figure 2), using the following conventions of noting and representation:

- the peaks of the graph are put in a certain order;

- next to each arc is the value $v_{ij} = t_{ij}$ (the duration for the corresponding event);
- next to each arc there are three digits, the first one stands for the natural duration of the event involved;
- on each step the arc $x_i x_j$ (for which the condition of maximum is observed) has been marked with a dotted line.

Table 1.

Nr.	Operation	Symbol	Duration	Directly conditioned operations
1	Transporting precast foundations	TF	4	SG
2	Transporting ballast for the concrete	TB	6	SG
3	Digging the holes for the foundation	SG	12	BF, MF
4	Transporting the concrete and concreting the foundations	BF	8	RS, MP
5	Mounting the precast foundation	MF	16	RS, MP
6	Transporting the poles	TS	8	AS
7	Assembling the poles	AS	17	RS
8	Raising the poles	RS	14	BC, MC
9	Transporting the conductors	TC	2	MC
10	Mounting the conductors	MC	15	VS
11	Mounting the ground socket	MP	5	-
12	Concreting the caps	BC	8	-
13	Dyeing the poles	VS	7	-

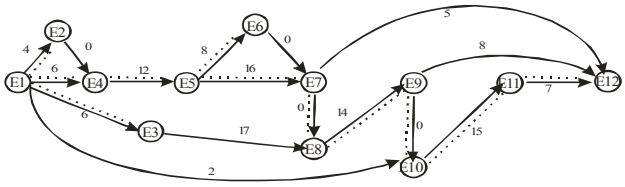


Figure 2.

On the first step we initialize for all events $t_i = 0$. On steps 2 and 3, we obtain

$t_2 = t_1 + t_{12} = 0 + 4 = 4$; $t_3 = t_1 + t_{13} = 0 + 8 = 8$

the arcs $E_1 - E_2$, respectively $E_1 - E_3$, being marked in a dotted line on the graph.

On step 4 we determine t_4: $t_4 = Max\{(t_1 + t_{14}) \quad (t_2 + t_{24})\} = 6$

the maximum condition being observed for the arc $E_1 - E_4$, which is therefore marked with a dotted line.

Proceeding analogously, we determine the natural duration for each event, in the end resulting $t_{12} = 70$ days, which is the value of CP.

In order to determine the CP, we look for a path from E_{12} to E_1, which must contain only marked arcs, resulting

$E_1 - E_4 - E_5 - E_7 - E_8 - E_9 - E_{10} - E_{11} - E_{12}$

as the one and only solution for the problem.

- ***The Bellman-Kalaba algorithm***

The matrix $[T]$ is shown in the next table, being written according to the operating times. According to the presented relations, the row-vector V^1 is identical to the last row of the $[T]$ matrix, it elements v_i^1 standing for the value of the roads which have the 1 length from the x_i peak to the x_n peak (from the E_i event to the E_n event)

$[T] =$

	1	2	3	4	5	6	7	8	9	10	11	12	
1	0	4	8	6	-∞	-∞	-∞	-∞	-∞	2	-∞	-∞	1
2	-∞	0	-∞	0	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	2
3	-∞	-∞	0	-∞	-∞	-∞	-∞	17	-∞	-∞	-∞	-∞	3
4	-∞	-∞	-∞	0	12	-∞	-∞	-∞	-∞	-∞	-∞	-∞	4
5	-∞	-∞	-∞	-∞	0	8	16	-∞	-∞	-∞	-∞	-∞	5
6	-∞	-∞	-∞	-∞	-∞	0	0	-∞	-∞	-∞	-∞	-∞	6
7	-∞	-∞	-∞	-∞	-∞	-∞	0	-∞	-∞	-∞	-∞	5	7
8	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	14	-∞	-∞	-∞	8
9	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	0	-∞	-∞	8	9
10	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	0	15	-∞	10
11	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	0	7	11
12	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	0	12

$[V] =$

	V^1	V^2	V^3	V^4	V^5	V^6	V^7	V^8	V^9
1	-∞	-∞	24	47	47	61	61	70	70
2	-∞	-∞	-∞	33	33	50	50	64	64
3	-∞	-∞	39	39	53	53	53	53	53
4	-∞	-∞	33	33	50	50	64	64	64
5	-∞	21	21	38	38	52	52	52	52
6	-∞	5	5	22	22	36	36	36	36
7	5	5	22	22	36	36	36	36	36
8	-∞	22	22	36	36	36	36	36	36
9	8	8	22	22	22	22	22	22	22
10	-∞	22	22	22	22	22	22	22	22
11	7	7	7	7	7	7	7	7	7
12	0	0	0	0	0	0	0	0	0

The elements of the row-vector V^2 are determined for $k = 2$, being the value of the roads which have a maximum length 2 from the x_i peak to the x_n peak. Analogously we determine V^3, V^4, ..., V^9.

The condition for the ending is observed by the elements of the vectors V^9 and V^8, the value of the CP being 70.

Next we will look for the peaks CP: considering that E_1 is part of the CP, we highlight the next peak, obtaining: $v_1^9 - v_4^9 = t_{14}$; $70 - 64 = 6$. Therefore the E_4 event, respectively the 1-4 arc, is part of the CP. $v_4^9 - v_5^9 = t_{45}$; $64 - 52 = 6$

Proceeding analogously, in the end we will find the requested CP , the result being the same as that obtained via the Ford algorithm (table 2).

Table 2.

Nr.	Operation	t_i	t_i^*	t_j	t_j^*	t_{ij}	mt_{ij}	ml_{ij}	ms_{ij}
1	$E_1 - E_2$	0	0	4	6	4	2	0	0
2	$E_1 - E_3$	0	0	8	17	8	9	0	0
3	$E_2 - E_4$	4	6	6	6	0	2	2	0
4	$E_1 - E_{10}$	0	0	48	48	2	46	46	46
5	$E_3 - E_8$	8	17	34	34	17	9	9	0
6	$E_5 - E_6$	18	18	26	34	8	8	0	0
7	$E_7 - E_{12}$	34	34	70	70	5	31	31	31
8	$E_9 - E_{12}$	48	48	70	70	8	14	14	14

4. CRITIC calculation program

4.1. Overview

The CRITIC calculation program has been elaborated in Turbo Pascal language, using the Delphi environment. It finds the CP via a program graph and it solves a series of related problems.

The following aspects related to the calculation program may be noted:

a) The program solves all aspects related to the creation and actualization of the data base related to a certain application, as well as the loading and saving of the files that contain the data bases;

b) A large number of check-ups for the compatibility and the accurateness of the elements in the data base, with warning or error messages if necessary;

c) The CP may be found via the Ford algorithm and the Bellman-Kalaba algorithm;

d) After finding the critical path the time stocks are calculated: those related to events (the fluctuation intervals) and those related to operations (operation margins);

e) The problems also having a didactic side, the visualization of the results may be done according to the user's wish: from the sole display of the final results to the visualization in a work window of all the intermediate results for each iteration, with the option of browsing them;

f) The option of saving or listing the result files is available.

4.2. Delphi® environment

Introduced as Object Pascal 7.0, the firm that produced it was not able to give up the codename Delphi, due to its market popularization.

Delphi introduces several concepts unknown in the Pascal language, such as: classes, properties, interfaces and objects. The classes derive from the old objects Pascal, the method concept setter-getter being introduces in an original manner.

Delphi is part of the RAD (Rapid Application Development) applications package, being a visual environment, via which we can easily build Windows interfaces.

The programming style is named OOP (Object Orientated Programming), known in the world of Borland developers as PME (Properties-Methods-Events).

There is a great similarity between the Delphi environment and Turbo Vision, both operating new objects and having a new way of treating events and objects. Due to the fact that Windows is a system based on messages, the Pascal should be modified so as to respond to messages, in order to comply with the present requirements. Delphi introduces the concept of notification procedure, which is actually a method of a class, this method is indirectly called by the system, its filling being actually a perpetual waiting after a user's action.

At this time, worldwide, for the standard of home user, the operating system is minimum Windows 98, and the industrial standard is Windows NT4. A strong tendency of migra-tion to the NT engines is obvious, now existing the possibility for a simple user and a firm to use the same operating system (Windows XP) for different problems in order to achieve maximum efficiency. This operating system is promoted in all fields where a computer is required.

The integrated developed environment (IDE) Delphi has been chosen since Pascal is an easy language, which allows to introduce easily and in a graduate manner all the Windows programming problems, without losing much of the specifics of the applications.

4.3. Using the program

CRITIC program, elaborated in **Delphi**® environment, version 5, is a classical application for **Windows**®, offering a facile interface for the students, during the practical works for the ***Techniques of optimization in the power industry*** course, as well as for all those interested in such applications.

When it is launched, the program window will have the following aspect (figure 3):

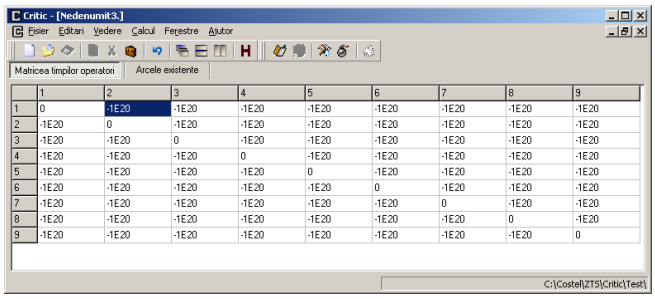


Figure 3.

A new file (empty) is loaded, and it contains a matrix which represents a graph with n nodes and which is to be filled in by the user with all the necessary data. This is the matrix of the operating times. The program also allows in this window the option to see the matrix of the existing arcs between the events involved.

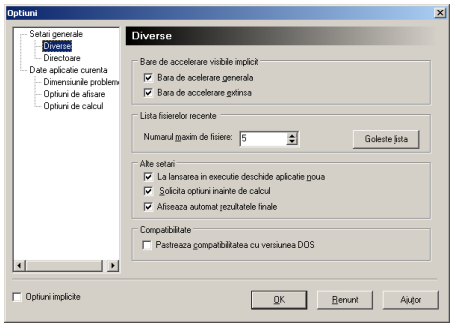


Figure 4.

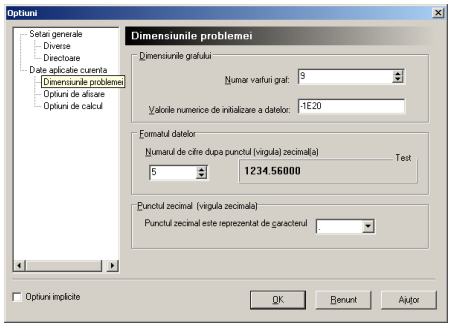


Figure 5.

Before initialising the process of filling-in the elements of this matrix, the user must access the menu *Vedere – Optiuni*, or o click on the according button on the toolbar. This command will lead to another window, in which the user may specify the number of peaks of the graph, as well as other figures (the number of exact digits, the initial value of the data etc.).

After introducing the information necessary in order to fill in the matrix, the user has the option to save that file under a certain name, for a subsequent use.

The window of the application for the matrix that contains the user data, and the table with the existing arcs will have the following aspect:

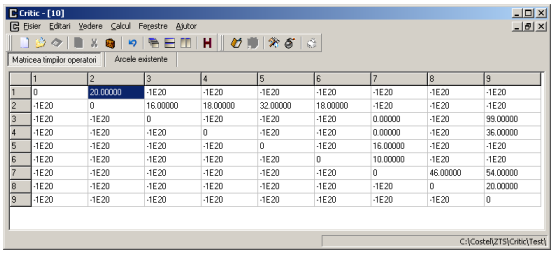
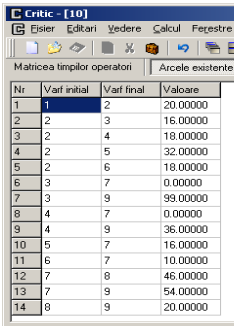


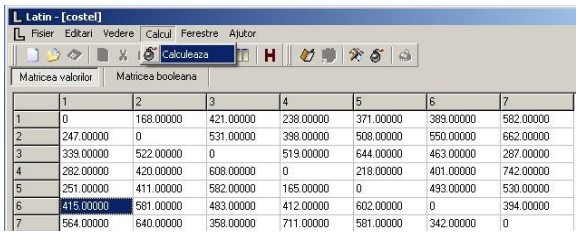
Figure 6.



Nr	Varf initial	Varf final	Valoare
1	1	2	20.00000
2	2	3	16.00000
3	2	4	18.00000
4	2	5	32.00000
5	2	6	18.00000
6	3	7	0.00000
7	3	9	99.00000
8	4	7	0.00000
9	4	9	36.00000
10	5	7	16.00000
11	6	7	10.00000
12	7	8	46.00000
13	7	9	54.00000
14	8	9	20.00000

Figure 7.

After having gone through the process of filling-in the data matrix, the user accesses the menu *Calcul – Calculeaza*.



	1	2	3	4	5	6	7
1	0	168.00000	421.00000	238.00000	371.00000	389.00000	582.00000
2	247.00000	0	531.00000	398.00000	508.00000	550.00000	662.00000
3	339.00000	522.00000	0	519.00000	644.00000	463.00000	287.00000
4	282.00000	420.00000	608.00000	0	218.00000	401.00000	742.00000
5	251.00000	411.00000	582.00000	165.00000	0	493.00000	530.00000
6	415.00000	581.00000	483.00000	412.00000	602.00000	0	394.00000
7	564.00000	640.00000	358.00000	711.00000	581.00000	342.00000	0

Figure 8.

After selecting this option, the "Optiuni" window (seen earlier) will appear again and it will mainly allow to configure 3 parameters: the dimensions of the problem (tackled above); the display options: whether to print the initial data or not, the intermediary results, whether to clear or not the screen before displaying the results; the calculation options: stipulating the algorithm to be used to determine the critical path, whether to calculate or not the fluctuation intervals, or whether to calculate or not the margins.

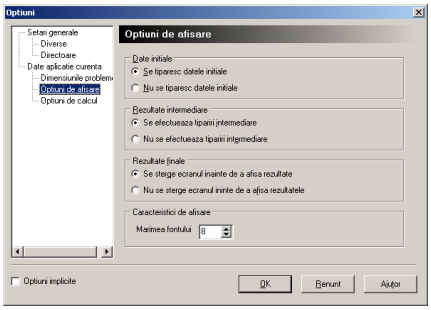


Figure 9.

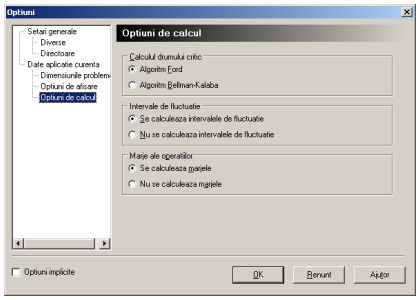


Figure 10.

By clicking the OK button, the calculation process is initiated. In the window which appears upon completion, the results of the calculation process are presented, according to the user's wishes.

The application allows the export of the initial data, respectively of the results as text or rtf files (figure 11), files that may be later opened via programs in the **Office**® package.

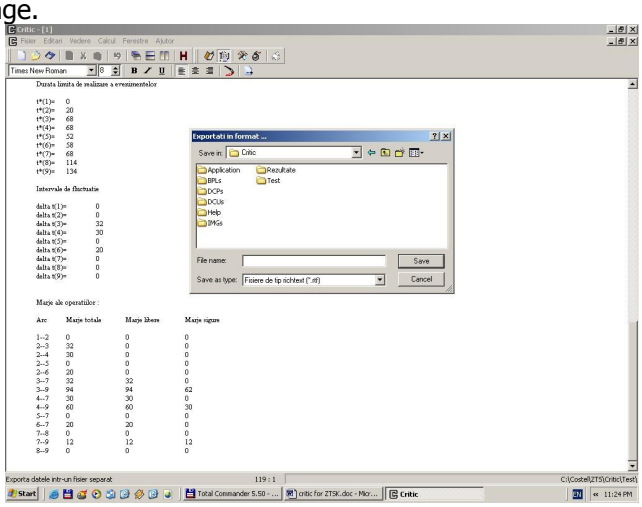


Figure 11.

References

[1] Kilyeni Şt., *Tehnici de optimizare în ingineria energetică*. Editura Orizonturi Universitare, Timişoara, 2004

[2] Kilyeni Şt., *Metode numerice. Algoritme, programe, aplicaţii în energetică, Ed. a 3-a.* Editura Orizonturi Universitare, Timişoara, 2004

[3] Kilyeni Şt., Groza M., Limbean G., *Tehnici de optimizare în ingineria energetică. Lucrări practice.* Editura Orizonturi Universitare, Timişoara, 2003

[4] Momoh J.A., *Electric Power System Applications of Optimization.* Marcel Dekker Inc., New York, 2000

[5] *Borland Delphi 3. Object Pascal Language Guide.* Borland International, 1997

[6] *Borland Delphi 3. Visual Component Library Reference.* Borland International, 1997

Addresses:

- Asist. Drd. Ing. Marius Groza, "Politehnica" University Timişoara, Electrical and Power Engineering Faculty, Bd. V. Pârvan Nr. 2, Timişoara, Romania, e-mail marius.groza@et.upt.ro

- Constantin Bărbulescu, student IVth year, "Politehnica" University Timişoara, Electrical and Power Engineering Faculty, Bd. V. Pârvan Nr. 2, Timişoara, Romania, e-mail costelbarbulescu@k.ro
- Asist. Drd. Ing. Gabriel Limbean, "Politehnica" University Timişoara, Electrical and Power Engineering Faculty, Bd. V. Pârvan Nr. 2, Timişoara, Romania, e-mail gabi.limbean@et.upt.ro