T. NICOLAE

About Bellman-Kalaba's Algorithm for Determining the Optimum Path in a Graph

The matricial organizing of computations for determining the path of optimum value in a graph allows this important problem in the theory of graphs to be solved rapidly and with only a small volume of computations.

Bellman-Kalaba's algorithm constitutes one of the methods used to find the path of (minimum or maximum) optimum value between two vertices of a graph. Generally, we are interested in the path of optimum value between the entrance and exit of the graph.

The principle on which Bellman-Kalaba's algorithm is based, is: the path of optimum value is formed of partial paths of optimum value. Since the finding of the optimum is carried out by an iterative process according to certain formulae of computation, it requires, from the practical standpoint, a large volume of computations. Manual computations require long, concentrated attention. It will be seen hereafter that these disadvantages are obviated by matricial organizing of the computations.

Here is, in short, the contents of Bellman-Kalaba's algorithm for the case of the minimum:

Let be a graph $G = (X, \Gamma)$ without any loops and having a finite number of vertices in the set X. A numerical value c_{ij} (which may represent the cost or the time of transportation along the arc (X_iX_j) or other magnitudes of an economic character) is associated to any arc $(X_iX_j) \in U$, where U is the set of arcs in the graph. The entrance of the graph being noted by X_0, and its exit by X_n, hence, the graph having $(n + 1)$ vertices, the square matrix (c_{ij}) is formed, in which the elements represent the values associated to the arcs. On the main diagonal we shall have 0, since the graph has no loops, hence $c_{ii} = 0$. For two vertices (X_p, X_q) for which there is no connecting arc, we write $c_{pq} = \infty$.

The aim is to find a path $\mu = [X_0, X_{i1}, X_{i2}, ..., X_{ik}, X_n]$ such that the sum $c_{0i_1} + c_{i_1i_2} + ... c_{i_k}$ be a minimum.

To this end it is sufficient to solve the system of equations:

$V_i = min\,(V_j + C_{ij})$; $i = 0, 1 ... n - 1$; $i \neq j$

$V = 0$ where V_i is the value of the optimum path from vertex i to the terminal vertex.

The following formulae are then applied:

$$V_i^{(0)} = c_{in} \; i = 0, 1, 2, ..., n-1 \qquad (1)$$
$$V_n^{(0)} = 0.$$

This first iteration leads to the writing of n equalities after the previous writing of the matrix.

$$\left.\begin{array}{l} V_i^{(1)} = \min\limits_{i \neq j} (V_j^{(0)} + c_{ij}) \\ V_n^{(1)} = 0 \quad i \neq j \end{array}\right| \begin{array}{l} i = 0, 1, ..., n-1 \\ j = 0, 1, ..., n \end{array} \qquad (2)$$

This second iteration requires the determination of the minimum value, after $(n-1) \times n$ sums have been written.

$$\left.\begin{array}{l} V_i^{(k)} = \min (V_j^{(k-1)} + c_{ij}) \\ V_n^{(k)} = 0 \quad j \neq i \end{array}\right| \begin{array}{l} i = 0, 1, ..., n-1 \\ j = 0, 1, ..., n \\ k = 1, 2, 3, ... \end{array} \qquad (3)$$

The formulae *(3)* lead to the application of the successive iterations after the first two, until iteration k, for which we find $V_i^{(k)} = V_i^{(k-1)}$ for any i, when the algorithm comes to an end. In this phase $V_0^{(k)}$ represents the optimum value of the path between X_0 and X. If, for a graph with $(n+1)$ vertices, k iterations are made until the path of optimum value is found, this means that $(n-1) \times n \times k$ sums have been computed, from which k minimum values have been retained. For instance, for a graph with 9 vertices for which 5 iterations are made, the total number of operations is $8 \times 9 \times 5 = 360$; hence, rather a large number for such a graph.

In the matricial organizing of computations, for a graph like the one in the following example, the number of operations will decrease from 360 to about 40 for a beginner, and for a specialist the number may even be lower.

For the case of a path of maximum value (graph without circuits), the algorithm is modified as follows: the maximum of the sums in the formulae shown is computed instead of the minimum, and for the vertices that are not connected by any arc, $-\infty$ is written in the matrix.

After expounding the matricial organizing of computations we shall deal with a few examples by this method in view of showing concretely the very considerable economy in computations and time, achieved by this method.

Matricial organizing of computations (for the minimum)

Let be the matrix c_{ij} associated to the graph having the vertices X_0, X_1, ..., X_n.

Remark 1: It is no longer necessary to apply the formulae

$$\left.\begin{array}{l} V_i^{(0)} = c_{in} \\ V_n^{(0)} = 0 \end{array}\right| , \; i = 0, 1, ..., n-1 \qquad (1)$$

since the values $V_i^{(0)}$ are yielded precisely by the elements of the last column of matrix (1).

Remark 2: The above column matrix is taken and transposed, and we write the line matrix obtained in continuation to matrix c_{ij}. The formulae *(2)* are now applied as follows: the elements of the transposed matrix are added to the elements in the same position, in the lines of matrix c_{ij}, starting from the line 0 until the line $(n-1)$.

The following operations are performed: 1) a rectangular table is provided with lines for $(n+1)$ columns (as many as the matrix c_{ij} has) and with n lines (that is, one line less than in the matrix c_{ij}); 2) the squares for which

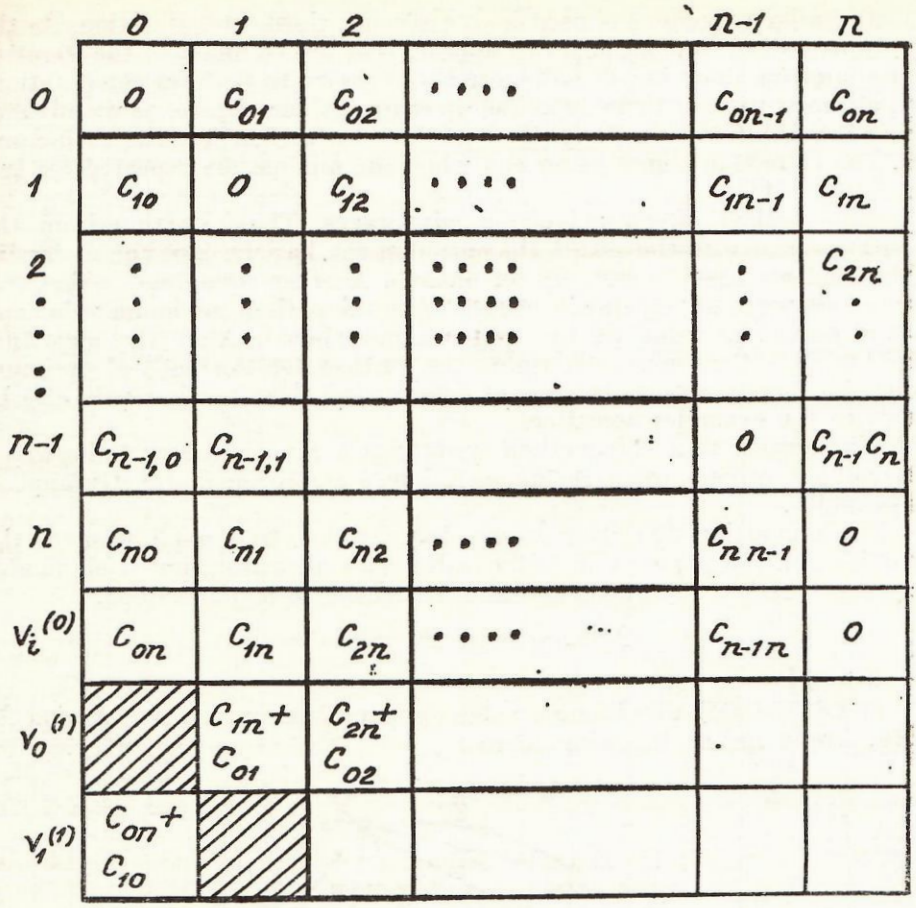

Fig. 1.

$i = j$ are hatched since the formulae *(2)* and *(3)* apply for $i \neq j$; 3) each column is totalled and the minimum by lines is taken for each line (in the examples hereafter it will be seen that this computation is easy to carry out). Considerable economy in computations is thus achieved since no totals will be made for the squares where at least one term of the sum is ∞ (and this happens for very many squares).

The omission of these totals is admitted since the minimum can only be found among the totals proceeding from the sum of two numbers, both finite.

The omission of these computations is shown in the table, in the respective squares, either by an X, or leaving them blank. After the first table has been computed the minimums by lines are taken and placed as the transposed was placed, continuing the following table. Since n minimum values were obtained in the rectangular table, the last minimum value $(an + 1 - a)$ will be 0 according to the formulae of the algorithm.

A similar procedure is used in the second, third, etc., iteration. As the iterations follow, minima begin to appear, that are common to the iterative tables, and for those it will no longer be necessary to perform computations but, at most, to copy them from the preceding table; hence, as we advance in our computations, we have the possibility to further decrease their number. The algorithm comes to an end when the minima are repeated for two successive tables.

This method offers undeniable advantages. Thus, starting from the matrix associated to the graph, the optimum can be determined on successive tables that are easy to draw up for anyone. Another advantage is that this method allows to obtain simultaneously both the path of minimum value and that of maximum value (if the graph admits both). After the algorithm comes to an end one must determine the vertices for the path of optimum value. A concrete process in view of determining these vertices will also be shown in the examples hereafter.

We mention that this method suppresses a supplementary table which is usually drawn up when Bellmann-Kalaba's algorithm is directly applied on formulae.

The organizing described above observes the strict application of the formulae and the sequence in Bellman-Kalaba's algorithm, but it eliminates very many useless operations and makes it easier to be understood.

Application of the method *

Examples

1) Let the path of minimum value be found between vertices X_0 and X_8 of the graph in the following figure:

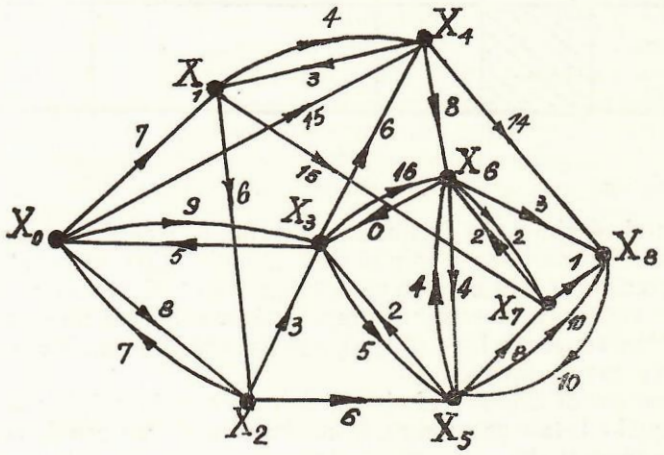

Fig. 2.

* The graph in this example is from A. Kauffmann's work *Méthodes et modèles de la recherche opérationnelle*, vol. II, Ed. Dunod, Paris, 1964, pag. 291—294. (This example has been intentionally chosen to offer the reader the possibility of comparing the method expounded to the one requiring the application of the formulae).

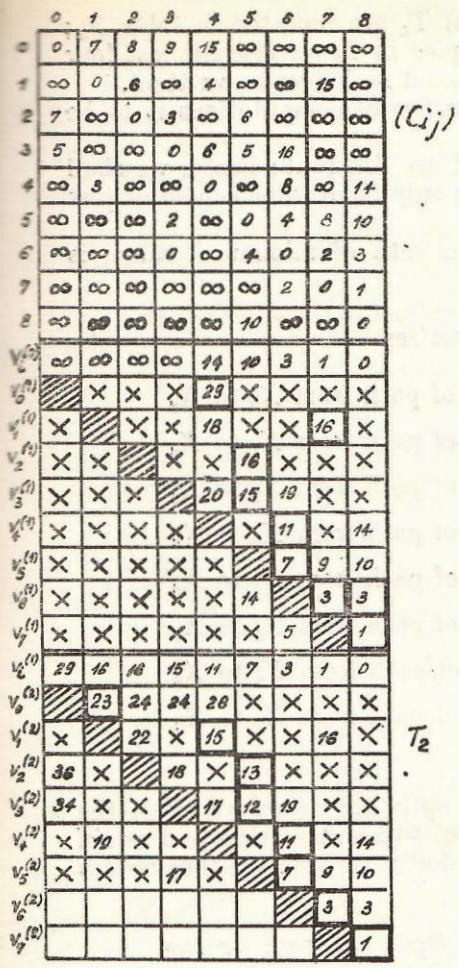

Fig. 3.

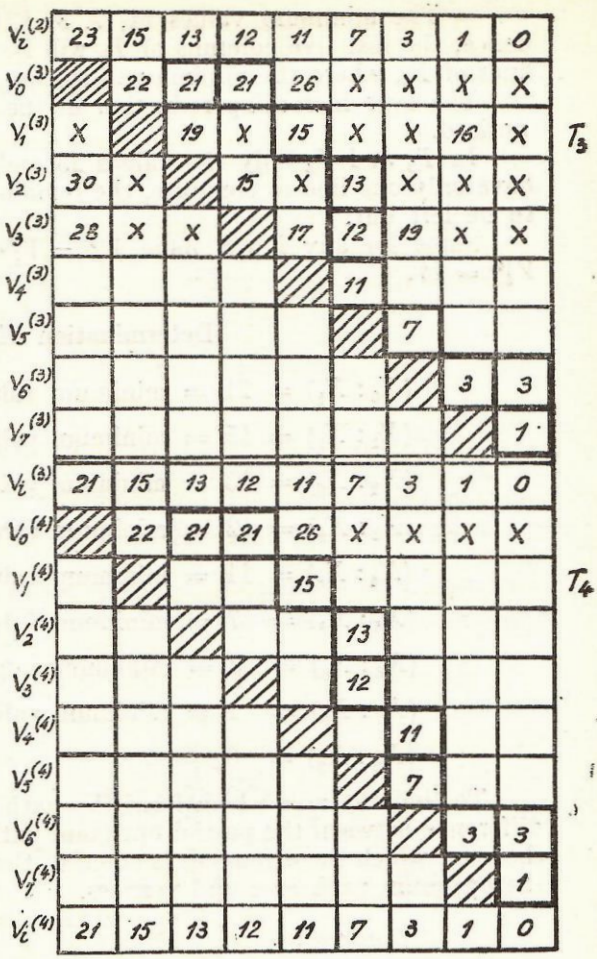

Fig. 4.

Explanation of computations

— we write the matrix associated to the graph (0 on the main diagonal and ∞ for the case when there is no arc connecting the respective vertices);

— we transpose the last column;

— we compute the first table. We hatch the squares for which $i = j$ and write an X where the sum contains ∞. We now see that in the first table 15 sums were effected instead of 72 that are necessary when applying the formulae directly.

— We take the minimum on each line and form the first line of the following table where we now proceed similarly.

Remark. In table T_2. The last three minimum values 3, 1, 0 are the same as in the first iterations; hence, we copy the last three columns (or wo no longer write the last two lines of T_2 where the minima will subsist).

— The minimum values 11, 7, 3, 1, 0 of T_2 are repeated in table T_3; hence, the last five columns of T_2 will be copied in T_3 (or else the last four lines of T_3, where the minima will subsist, should no longer be written).

The ever increasing reduction in the number of computations is to be noticed.

In T_3 and T_4 only the values 23 and 21 are different; hence, we shall have only one line to compute, the remaining only have to be copied or just to be left out.

Since for any i we have $V_i^{(4)} = V_i^{(3)}$ the path of minimum value is $V_0^{(4)} = 21$.

Determination of vertices

$(X_0 : X_8) = 21 =$ minimum value of path from X_0 to X_8

$(X_1 : X_8) = 15 =$ minimum value of path from X_1 to X_8

$(X_2 : X_8) = 13 =$ minimum value of path from X_2 to X_8

$(X_3 : X_8) = 12 =$ minimum value of path from X_3 to X_8

$(X_4 : X_8) = 11 =$ minimum value of path from X_4 to X_8

$(X_5 : X_8) = 7 =$ minimum value of path from X_5 to X_8

$(X_6 : X_8) = 3 =$ minimum value of path from X_6 to X_8

$(X_7 : X_8) = 1 =$ minimum value of path from X_7 to X_8

$(X_8 : X_8) = 0$

To find the arcs which form the path of optimum value, we effect the difference between the partial optimum paths of adjacent vertices and retain those for which the difference is equal to the value of the arc. We thus obtain the optimum path arcs and vertices.

	Concrete:	*Optimum path vertices*
I.	$21 - 13 = 8 = l(X_0; X_2); (X_2)$	$X_0 \quad X_0$
	$21 - 12 = 9 = l(X_0; X_3); (X_3)$	$X_2 \quad X_3$
		$X_5 \quad X_5$
		$X_6 \quad X_6$
II.	$13 - 7 = 6 = l(X_2; X_5); (X_5)$	$X_7 \; X_8 \; X_7 \; X_8$
	$12 - 7 = 5 = l(X_3; X_5); (X_5)$	$X_8 \quad X_8$
III.	$7 - 3 = 4 = l(X_5; X_6); (X_6)$	
IV.	$3 - 2 = 1 = l(X_6; X_7); (X_7)$	
	$3 - 0 = 3 = l(X_0; X_0); (X_8)$	

We notice that four paths of optimum value have been determined:

$$\mu_1 = (X_0, X_2, X_5, X_6, X_7, X_8) = 21$$

$$\mu_2 = (X_0, X_2, X_5, X_6, X_8) = 21$$

$$\mu_3 = (X_0, X_3, X_5, X_6, X_7, X_8) = 21$$

$$\mu_4 = (X_0, X_3, X_5, X_6, X_8) = 21$$

*

Note: In the example treated, one finite minimum was found on each line. There are cases when no finite sum appears in the first few lines. In that case the very value ∞ is taken as a minimum and the iterations are continued until finite minima appear, which will obviously happen if the graph admits a path of optimum value.

2) Let the path of minimum value, as well as the path of maximum value between the vertices X_0 and X_6 of the circuitless graph in the following figure be found:

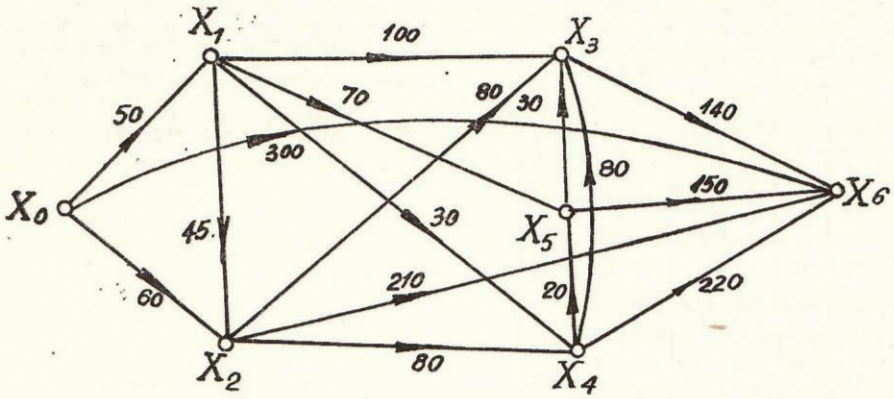

Fig. 5.

Both problems will be solved simultaneously by writing in each square first the values for the problem of minimum, and under them the values for the problem of maximum. In the matrix we note $\pm\infty$ in the squares corresponding to two vertices between which there is no arc ($+\infty$ for the minimum; $-\infty$ for the maximum.)

In the iterative tables we note X whenever $+\infty$ or $-\infty$ occur when summing up.

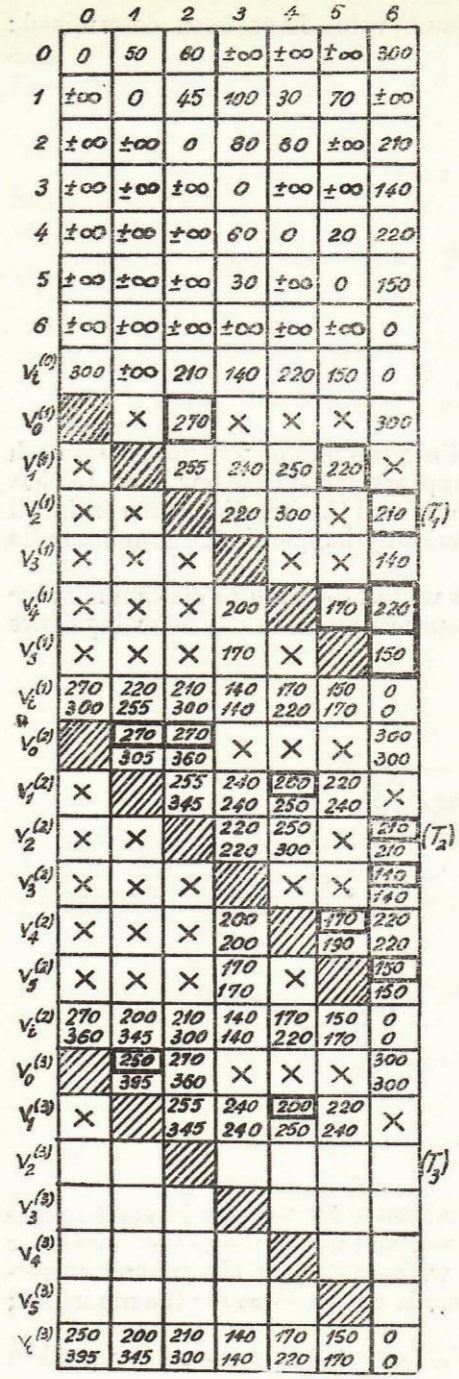

Fig. 6.

Explanations. In T_1 we retained the minimum sum 270 and the maximum sum 300 in the first line.

We proceeded likewise in the other lines.

In the fourth line the minimum value = the maximum value = 140.

Optimum paths obtained:

— minimum $\mu_1 = 250 =$

$(X_0, X_1, X_4, X_5, X_6)$

— maximum $\mu_2 = 395 =$

$(X_0, X_1, X_2, X_4, X_6)$

AU SUJET DE L'UTILISATION DE L'ALGORITHME DE BELLMAN-KALABA EN VUE DE DETERMINER LA VOIE OPTIMUM DANS UN GRAPHE

La détermination de la voie de valeur optimum (maximum ou minimum) dans un graphe, à l'aide de l'algorithme de Bellman-Kalaba s'effectue par un procédé itératif qui nécessite un volume de calculs trés élevé ainsi qu'une attention soutenue.

On recommande dans cet article une organisation matricielle des calculs, permettant une réduction considérable du volume des opérations. L'organisation suggérée présente également l'avantage qu'elle est d'une assimilation et d'une application faciles.

Deux exemples sont traités dans cet article, en vue d'illustrer la méthode qu'on propose. Dans le premier, on détermine la voie de valeur minimum dans un graphe et, dans le second, autant la voie de valeur minimum que celle de valeur maximum (simultanément), ce qui d'ailleurs constitue encore un avantage de la méthode suggérée.

ОБ АЛГОРИТМЕ БЕЛЛМАН-КАЛАБА ДЛЯ ОПРЕДЕЛЕНИЯ ОПТИМАЛЬНОГО ПУТИ В ГРАФЕ

Определение пути оптимальной стоимости (максимальной или минимальной) в графе при помощи алгоритма Беллман-Калаба делается по итеративному методу, требующему очень большого количества расчетов и напряженного внимания.

В работе рекомендуется матричная организация расчетов, следствием которой является значительное сокращение количества операций. Также, преимуществом предложенной организации является легкость ее применения и усвоения.

В статье приводятся два примера иллюстрирующих этот метод. В первом примере определяется путь минимальной стоимости в графе, а во второй — путь минимальной стоимости и максимальсной стоимости (одновременно), что является другим важным преимуществом предложенного метода.