APPLICATION OF CHAIN SEARCH IN ONE ALGORITHM FOR SOLVING ON A COMPUTER A CERTAIN CLASS OF PLANNING-ECONOMIC PROBLEMS

MATHEMATICS

1967

SovietRxiv

Abstract

Full Text

UDC 519.95

MATHEMATICS

Corresponding Member of the Academy of Sciences of the USSR L. A. LYUSTERNIK, I. E. MAIZLIN

APPLICATION OF CHAIN SEARCH IN ONE ALGORITHM FOR SOLVING ON A COMPUTER A CERTAIN CLASS OF PLANNING-ECONOMIC PROBLEMS

Modern systems of production planning and management accumulate and process large arrays of economic information. In algorithms for processing this information on a computer, the foremost task is the organization of an economical search, which, being the most widespread procedure, determines the total time for solving problems and the efficiency of the control system.

In the present paper, closely connected with ([1], [2]), an algorithm is described for solving one class of planning-economic problems, in the implementation of which on a computer a chain information search is applied ([3]).

Before defining the class under study, let us formulate three typical problems of this class.

Problem 1. Compilation of specifications.

To take account of complex relations among different articles within a given production process, one has to solve the following problem. There are complex articles of n types $x_1, x_2, \dots, x_n$. Suppose that x_i $(i = 1, 2, \dots, n)$ is the code of the i-th article and $1 \leq x_i \leq n$. Article x_j contains l_{ij} articles x_i $(i, j = 1, 2, \dots, n)$. Denote

$$m = \max_{1 \leq j \leq n} m_j,$$

where m_j is the number of nonzero numbers in the sequence $l_{1j}, l_{2j}, \dots, l_{nj}$. In practice, usually $m \ll n$, and it is convenient to specify the initial information by triples (x_i, x_j, l_{ij}), arranged in an arbitrary order.

It is required, for $1 \leq \alpha \leq n$, $1 \leq \beta \leq n$, to determine y_α^β—the number of articles x_α contained in one article x_β.

This problem reduces to the following. Let G be a directed multigraph without circuits ([4]) with n vertices $x_1, x_2, \dots, x_n$; l_{ij} is the number of arcs running from vertex x_i to vertex x_j $(i, j = 1, 2, \dots, n)$. Then the quantity y_α^β defined above

$(\alpha, \beta = 1, 2, \ldots, n)$ is equal to the number of all possible paths leading from vertex x_α to vertex x_β.

Problem 2. Determination of the completion time of development under network planning and management.

The mathematical model of a system of network planning and management is a directed graph G without circuits, whose arcs correspond to jobs, while the vertices denote events consisting in the completion of all jobs corresponding to the arcs entering that vertex.

Let the events (vertices of the graph) have numbers $x_1, x_2, \ldots, x_n$ $(1 \leq x_i \leq n)$. The initial information for the time model of the NPM system is given in the form of an unordered collection of triples of numbers (x_i, x_j, l_{ij}) corresponding to each job, where x_i is the number of the preceding event, x_j is the number of the following event, and l_{ij} is the duration of the given job. Let x_α be the initial event (the beginning of development), and x_β the final event (the end of development), i.e., in the graph G there are no arcs of the form (x_i, x_α) or (x_β, x_j). Denote

$$m = \max_{1 \leq j \leq n} m_j,$$

where m_j is the number of jobs immediately preceding event x_j. Usually $m \ll n$.

It is required to determine the duration of the entire development, which

is equal to the length of the longest path leading from vertex x_α to vertex x_β.

Problem 3. Determination of the length of the shortest path between two nodes of a transportation network.

With one-way or predetermined direction of movement along each segment of a route, by specifying the network analogously to the preceding problem, one can pose the problem of finding the length of the shortest path from node x_α to node x_β.

We now give a general formulation of the problem. Let $G = (X, \Gamma)$ be a directed graph without circuits, with n vertices $x_1, x_2, \ldots, x_n$, numbered in an arbitrary order, and N arcs (x_i, x_j), where $x_j = \Gamma x_i$. Suppose there is a unique initial vertex x_α and a unique terminal vertex x_β, i.e., in G there are no arcs of the form (x_i, x_α) or (x_β, x_j). On the arcs $(x_i, x_j) \in G$ a function $f(x_i, x_j)$ is given. Two operations $(*)$ and $(+)$ are defined on pairs of numbers,

$$z = x * y, \qquad z = x + y.$$

It is required to determine at each vertex x_j $(j = 1, 2, \ldots, n)$ of the graph G the values of the function $g(x_j)$, equal to 0 at x_α and satisfying, for $j = 1, 2, \ldots, n;\ j \neq \alpha$, the recurrence relations

$$g(x_j) = [g(x_{i_1}) + f(x_{i_1}, x_j)] * [g(x_{i_2}) + f(x_{i_2}, x_j)] * \ldots$$

$$... * [g(x_{i_p}) + f(x_{i_p}, x_j)],$$

where $x_{i_1}, x_{i_2}, ..., x_{i_p}$ is the set of precisely those vertices of G from which arcs leading to x_j emanate.

Problem 1 is reduced to this problem by replacing the multigraph by a simple graph of analogous structure, and we set $f(x_i, x_j) = l_{ij}$ and $g(x_\alpha) = 1$. The operation $(+)$ is multiplication, and $(*)$ is addition. The desired y_α^β is equal to $g(x_\beta)$.

In Problems 2 and 3 we also set $f(x_i, x_j) = l_{ij}$, but the operation $(+)$ is ordinary addition, while $(*)$ is the selection of the maximum of two quantities in Problem 2 and the minimum in Problem 3. The desired quantities are equal to $g(x_\beta)$.

We proceed to the description of the algorithm for determining the values of the function $g(x_i)$ $(i = 1, 2, ..., n;\ i \neq \alpha)$.

Let the information about the graph be given by a set of N triples of numbers $a[p]$, $b[p]$, and $y[p]$ $(p = 1, 2, ..., N)$, where $a[p]$ is the number of the vertex preceding the given arc, $b[p]$ is the number of the following vertex, and $y[p]$ is the value of the function f on the given arc. In addition, there are working arrays with values $c[i]$, $d[i]$, and $z[i]$ $(i = 1, 2, ..., n)$, and before the start of the algorithm $c[i] = d[i] = z[i] = 0^*$. Recall that the numbering of the vertices is arbitrary, the triples of numbers in the initial information are arranged in arbitrary order, and the index p has been introduced by us for convenience in describing the algorithm in a language close to machine language. We introduce working parameters γ, δ, δ_0, δ_1 and shall use the assignment sign $(:=)$ to distinguish the new value of a quantity (on the left-hand side) from the old value, which enters into the right-hand side.

During the operation of the algorithm, a chain search will be carried out in order to determine the numbers of triples with a specified value of $a[p]$. The number of operations for finding all triples for which $a[p] = k$ will not exceed $c^{(1)}m$ for each k $(1 \leq k \leq n)$, where $c^{(i)}$ here $(i = 1)$ and below is a constant depending on the type of machine, and $m = \max_{1 \leq j \leq n} m_j$ was defined above.

For the implementation of the chain search (3), preliminary processing of the arrays $a[p]$ and $c[p]$ is performed. Successively for $p = N, N-1, ..., 2, 1$ we carry out the following procedure: set $\delta = a[p]$, $a[p] =$

---

* In Problem 1 we set $z[\alpha] = 1$.

$= c[\delta]$, and $c[\delta] = p$. Having initially set $d[p] \equiv 0$, successively for $p = 1, 2, ..., N$ we determine $\delta = b[p]$ and increase $d[\delta]$ by one.

The main algorithm is divided into stages. Each stage corresponds to a definite vertex, which we shall call the "stage" vertex. At the first stage the stage

vertex is the initial vertex x_α. The order in which the stage vertices follow is determined in the course of the algorithm with the aid of the array $d[p]$ and the working parameter δ_0, initially equal to α. Let us note, incidentally, that this order will determine the "proper" numbering of the vertices of the graph, under which, for all arcs, the number of the preceding vertex will be smaller than the number of the following vertex.

Let us describe one stage of the algorithm. Let the stage vertex be the vertex with number γ (at the first stage $\gamma = \alpha$). We determine $\delta_1 = c[\gamma]$. Next we set $\delta = b[\delta_1]$ and then

$$z[\delta] := z[\delta] * \{z[\gamma] + y[\delta_1]\}.$$

We decrease $d[\delta]$ by one and, if $d[\delta]$ has become equal to zero, set $d[\delta_0] = \delta$, and then $\delta_0 = \delta$. Next we determine $\delta_1 := a[\delta_1]$. If $\delta_1 = 0$, then the stage is completed. If $\delta_1 \neq 0$, then we set $\delta = b[\delta_1]$ and carry out with $z[\delta]$ and $d[\delta]$ the same operations as for the preceding value of δ at this stage.

Next we again determine $\delta_1 := a[\delta_1]$, compare δ_1 with zero and, if $\delta_1 \neq 0$, determine $\delta = b[\delta_1]$, etc. Finally, at some step it will turn out that $\delta_1 = 0$. At this point the given stage is completed. We determine the number γ of the stage vertex for the next stage by the formula $\gamma := d[\gamma]$ and pass to this stage if $\gamma \neq 0$ and $\gamma \neq \beta$. If $\gamma = 0$, this indicates the presence in the graph of circuits or of an initial vertex different from x_α.

The equality $\gamma = \beta$ is a sign that the algorithm has finished. By this time $z[p]$ $(p = 1, 2, \ldots, n)$ are equal to $g(x_p)$. For problem 1 we then obtain n quantities $y^p_\alpha = g(x_p)$, $p = 1, 2, \ldots, n$. In problem 2, $g(x_p)$ is equal to the early time of occurrence of the event x_p $(p = 1, 2, \ldots, n)$.

Let us estimate the number of operations in implementing the algorithm on a computer. The algorithm consists of $n - 1$ stages. Inside each stage there is a cycle of repeated operations, and the number of repetitions is determined by the number of the step at which $\delta_1 = 0$ has occurred. But there can be no more than m such steps, i.e. the total number of operations does not exceed $c^{(2)}mn$. In practice, as noted above, $m \ll n$. The memory size in implementing the algorithm is equal to $2N \log_2 n + 2n \log_2 n$ binary digits and, in addition, it is necessary to store $N + n$ numbers, the range of whose values is determined by the quantities l_{ij} and $g(x_i)$.

Received
25 VIII 1967

REFERENCES

1. I. E. Maizlin, DAN, **159**, No. 4 (1964).

2. L. A. Lyusternik, I. E. Maizlin, UMN, **21**, No. 2 (128) (1966).

3. L. R. Johnson, Comm. ACM, **4**, No. 5 (1961).

4. C. Berge, *The Theory of Graphs and Its Applications*, IL, 1962.

Note: Figure translations are in progress. See original paper for figures.

Source: Math-Net.Ru and CyberLeninka. Machine translation. Verify with the original.