MINIMIZATION OF DATA FILE GENERATION TIME IN MANAGEMENT INFORMATION SYSTEMS

V. N. Burkov and V. A. Kletin

UDC 658.52.011.56

A technique is described for minimizing data file generation time by optimizing their processing sequence. The problem is stated formally and cases are cited for which efficient procedures and appropriate algorithms can be devised. Application of the proposed technique to the generation of software for a specific management information system is described.

1. Introduction

The operation of management information systems (MIS) is, as a rule, associated with processing of large numbers of data files and generation of output documents both of which consume considerable machine time. Attempts to reduce such time losses are based on the use of data banks and on a careful study of data file arrangement on machine carriers (magnetic tape, disks, drums, etc.), of file organizations (index-sequential, relative, direct, sequential), and of encoding systems [1-3].

The basic elements of MIS data bases are data files intended either for long-term or for temporary information storage. Data files located in the system data bank for long-term storage, centralized updating, and general use are called master files. All other files intended for temporary storage and for application by a specific user are called working files.

Since the contents of various working data files frequently intersect, the above methods of processing-time minimization can be enhanced by devising a data file generation sequence which would take advantage of file interaction for shortening their generation time. In other words, some working files are generated not directly from master files stored in data banks but with the aid of already available and more "accessible" files. This also includes the production of intermediate files which are not needed for MIS operation but help to save time for the generation of working files.

In the following the problem is stated in a formal way, cases for which efficient procedures and appropriate algorithms can be devised are cited, and a program package implementing this approach in the design of a specific MIS is described.

2. Formal Statement of Problem

Let there be given a weighted oriented graph G(X, U), where X is the set of nodes ($|X| = n$) and U is the set of edges ($|U| \leq n^2$). To each edge $(i, j) \in U$ is assigned a weight $t(i, j) \geq 0$. On the set of nodes X of the graph G(X, U) are selected a node $s \in X$ with no edges entering into it and a subset of nodes $Q \in X \setminus \{s\}$ such that any node $q \in Q$ is accessible from s along edges of the graph G(X, U). On G(X, U) we want to isolate a subgraph $G^1(X^1, U^1)$ such that [4-6] $G^1(X^1, U^1)$ is a tree with a root in s, $Q \subseteq X^1$, and

$$\sum_{(i,j)\in U^1} t(i,j) \to \min. \tag{1}$$

We will show that the problem of data file generation formulated above can be reduced to the problem (1). To each data file let us put into correspondence a node of the graph G(X, U). The data filed include those necessary for MIS operation as well as intermediate files not directly needed for MIS operation but used for reducing the time needed for generation of working files. Data files needed for MIS operations are said to be obligatory, i.e., they must be generated. Two nodes are joined by an edge $(i, j) \in U$ if the j-th file can be generated on the basis of the i-th file. To each edge $(i, j) \in U$ is assigned a weight t(i, j) equal to the time needed for generation of the j-th file on the basis of the i-th file. On the set of nodes X of the graph G(X, U) we select a subset of nodes

Moscow. Translated from Avtomatika i Telemekhanika, No. 2, pp. 86-91, February, 1982. Original article submitted December 8, 1980.

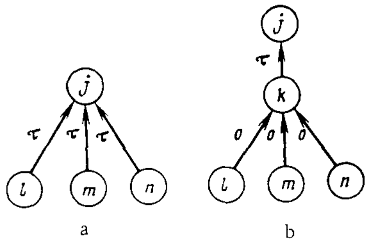

Fig. 1

$Q \subset X$ corresponding to obligatory data files and a node $s \in X \setminus Q$ corresponding to master files. Thus, the problem of generating data files needed for MIS operation in minimum time is reduced to finding a minimal tree with a root in node s on the graph G(X, U) corresponding to the given problem.

It can be shown that the case in which a j-th file is generated on the basis of several other files in time τ can be also be reduced to (1). In such a case a fictitious node k is added to the graph G(X, U) which can be accepted from the same nodes as j in zero time. In the new graph $G^1(X^1, U^1)$ the node j is accessible only from k and $t(k, j) = \tau$ (see Fig. 1a, b).

The following problems can be shown to be reducible to the problem (1): construction of electric transmission lines; construction of gas and oil pipelines; making decisions about the development of deposits; transmission of messages; etc. [5-7].

3. Solution Algorithms

The problem formulated above is a problem in discrete programming which for a general solution requires the application of various sorting procedures such as branch-and-boundary methods, the Balas algorithm, etc. It is, however, possible to indicate certain conditions which allow the application of efficient procedures that guarantee a global optimal solution and do not require full sorting. A trivial example of such problems is the case when in (1) only one node of $X \setminus \{s\}$ must be accessible from s; obviously, in this case one can use the shortest-path method [8].

If all nodes of the subset $X \setminus \{s\}$ are "obligatory" one can use one of the procedures effective for different types of graphs; a description of these procedures is given below.

Mesh Search. Let G(X, U) in (1) be a mesh, i.e., it has no loops [s] and $Q = X \setminus \{s\}$. The following algorithms [5] can then be used to solve (1).

ALGORITHM 1

1. On the obtained mesh select a node belonging to the first level as counted from the root node.

2. Contract the selected node to the root node.

3. Add the weight of the contracted edge of the former weight and record the index.

4. If the graph is exhausted, i.e., if all edges have been contracted to the root node, the algorithm is completed and the solution is obtained in step 3 of the last iteration; otherwise go to step 5.

5. Test for the presence of parallel edges in the resulting mesh. If such edges are found go to step 6; otherwise return to step 1.

6. Of all parallel edges leave the one whose weight is minimal and eliminate all others. Return to step 1.

ALGORITHM 2

1. On the obtained mesh select a node not considered before. If no such edges are present proceed to step 3; otherwise continue with step 2.

2. On the set of edges entering the node selected in the preceding step retain the edge with a minimal weight and eliminate all other edges. Return to step 1.

3. End of algorithm. The resulting tree is the solution. Proving that the algorithm is finite and the optimal solution is trivial. The first follows from the finiteness of the graph and the second, from the properties of meshes.

If a more general case, viz., $Q = X^1 \subset X\backslash\{s\}$, is considered the Algorithms 1 and 2 are, generally speaking, not valid since they give only the upper bound of the resultant weight of the minimal-tree edges. To compute the lower bound use the following procedure.

ALGORITHM 3

1. Mark all nodes of the subset $X^1 \subset X$ and assign to the variable S a weight zero.

2. If all marked nodes have been contracted to the root node go to step 9, if not continue with the next step.

3. Select an arbitrary node on the set of marked nodes $X\backslash\{s\}$.

4. On the set of edges entering the node selected in step 3 of the last iteration take the edge having a minimal weight Δ.

5. Reduce the weights of all edges entering the selected node by Δ.

6. Increase S by Δ.

7. "Contract" the nodes connected by the edge with zero weight and mark the newly formed node.

8. If contraction of the nodes resulted in parallel edges retain the edge with minimal weight and eliminate all other edges. Return to step 2.

9. End of algorithm. S is equal to the lower bound of the total weight of the minimal tree and the contracted edges correspond to the edges of subset U^1.

Search On Planar Null Graphs. If the starting graph G(X, U) is such that to each edge $(i, j) \in U$ corresponds an edge $(j, i) \in U$ and $t(i, j) \cdot t(j, i) = 0$, the graph is said to be a null graph.

Problems of optimizing the trajectory of particles in force fields can be frequently reduced to solving (1) on such graphs in which the weights of edges t(i, j) represent the loss of energy on moving from point i of the field to point j. Let $U^1 \subset U$ be the subset of edges with nonzero weights and $U'' = U\backslash U^1 \subset U$. It can be shown that the search on graph G(X, U) can be reduced to a search on a graph $G_1(X_1, U_1)$ such that $G_1(X_1, U_1^1)$ is a mesh symmetric to the mesh $G_1(X_1, U'')$ by contracting to a single node all loops the total weight of whose edges is zero. If $G(X_1, U_1)$ is planar the problem dual to (1) is the search of a minimal cut on a biconnected, planar, oriented graph dual to $G_1(X_1, U_1^1)$ [9]. However, as shown in [9], the magnitude of a minimal cut on planar graphs is equal to maximum circulation and is an integral quantity. Hence, if the starting graph is a null graph and satisfies the above conditions the search for a minimal tree can be implemented by the methods of linear programming.

Search on Graphs with Bicomponents

If $Q = X\backslash\{s\}$ but G(X, U) is an oriented graph with biconnected components, one can use a more complex algorithm devised for the search of maximal trees on arbitrary graphs [7]. The solution of problem (1) is in this case preceded by a simple procedure: Select a number $M = \max t(i, j)$ and assign a weight $r(i, j) = M - t(i, j)$ to each edge $(i, j) \in U$. The problem is then solved by the algorithm described below.

ALGORITHM 4

1. For each node $j \in X\backslash\{s\}$ on the obtained graph select an edge (i, j) such that $r(i, j) = \max_k r(k, j)$ and mark it. If the selected edges form no bicomponents go to step 4; otherwise continue with step 2.

2. Select any loop formed of marked edges and contract it to one of the nodes not belonging to it.

3. If the resulting graph has no biconnected components return to step 1; otherwise return to step 2.

4. If the loops of the graph have been contracted go to step 5; if no such loops are present go to step 9.

5. If the number of nodes of the obtained graph is $|X|$, go to step 9; otherwise go to step 6.

6. Replace one of the contracted loops V_q into the graph denoting the set of nodes belonging to V_q by $X(V_q)$ and the set of edges by $U(V_q)$.

7. To each edge $(i, j) \in U\backslash U(V_q)$ such that $i \in X\backslash X(V_q)$, $j \in X(V_q)$, assign a weight $r^1(i, j) = r(i, j) + \min_{(k,l) \in U(V_q)} r(k, l) - r(m, j)$, where $(m, j) \in U(V_q)$.

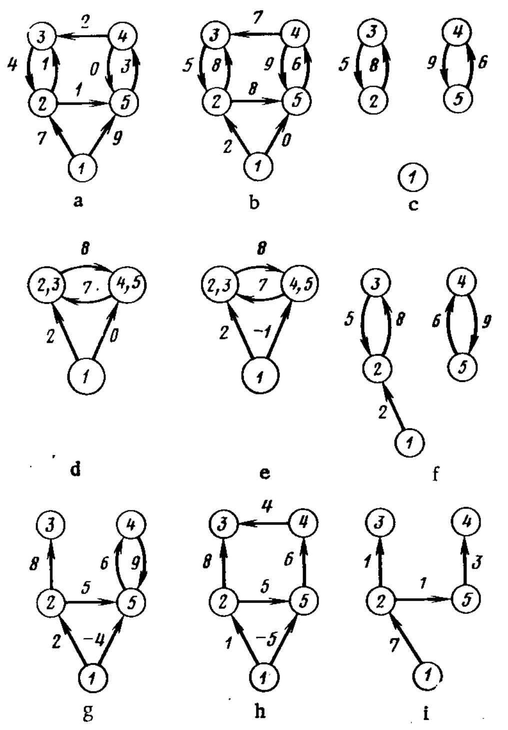

Fig. 2

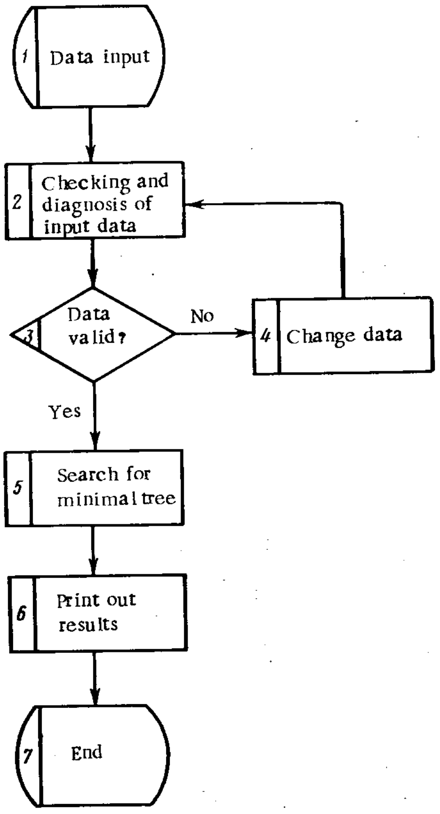

Fig. 3

8. On the subset of edges discussed in the foregoing step select the edge with maximum weight and mark it at the same time unmarking the edge entering into the same node which has been marked before.

9. The set of marked edges in the resulting graph gives the desired tree. End.

Example 1. Let us find the minimal tree with a root at x_1 on the weighted oriented graph G(X, U) shown in Fig. 2. The graph for finding the maximal graph and obtained from the starting graph is shown in Fig. 2b; the sequence of transformations implementing Algorithm 4 and the final graph forming the minimal tree are shown in Fig. 2c-h, and in Fig. 2i, respectively.

As noted before, in the general case one has to use for solving (1) one of the many sorting procedures whose effectivity is analyzed in [10]. This analysis served as a basis for the development of a program package described below.

4. DELTA Program Package

General Description. The purpose of the program package is to solve problem (1) and is preceded by entry, checking, and diagnosis of input data. The output results have the form of a contiguity matrix of the nodes of the obtained tree and the record equal to the sum of weights of edges belonging to the tree. Input data serve as the contiguity matrix of the initial graph nodes and a list of "obligatory" nodes of the final tree.

The package is based on the DELTA 3 program which searches for an optimal solution of (1) using a branch-and-boundary method [10].

The estimate $\Lambda(\pi)$ of the partial plan π containing the subset of nodes $Q \subset X$ of the initial graph G(X, U) is computed from

$$\Delta(\pi) = \sum_{x_j \in Q} \min_{x_i \in X_1^1} t(i,j) + \sum_{x_k \in X} \min_{x_l \in X \setminus Q} t(k,l),$$

where $X_1^1 \subset Q$ and the nodes of subset X_1^1 in π continue to x_j.

The block scheme of the package operating algorithm is shown in Fig. 3. The input data input, output, test, and diagnostic programs are implemented in COBOL and the optimal solution search program in FORTRAN IV.

The DELTA program package is designed to work on series ES computers fitted with an OS ES operating system, and a main memory of 100 kbytes. One limitation of the package is that all input data must be stored in the main memory. The DELTA package has been included in the software for the first line MIS. The principal features of the problems solved are given below.

Analysis of Package Efficiency in Solving Specific Problems. The subsystems of the first line of MIS include technico-economic planning, technological preparation of production, marketing, sales, and operational control which encompass the entire production control cycle from issuing a list of orders of preparing a set of documents for shipped products and exchange information by means of interconnecting files. The initial graph G(X, U), which represents the entire set of files and links between them has $|X| = 41$ nodes and $|U| = 89$ edges. Finding a locally optimal solution took 4 h by a group of experts and 2 h by the DELTA package, the programmed implementation providing a daily gain of computing time of 1.5-3 h as compared with the solutions proposed by experts; the overall economical gain amounts to 50,000 rubles/yr.

LITERATURE CITED

1. A. G. Mamikonov, A. N. Piskunov, and A. D. Tsvirkun, Models and Methods for Planning Automatic Control System Software [in Russian], Statistika, Moscow (1978).
2. V. N. Burkov and V. B. Sokolov, "Optimal arrangement of data files in magnetic tape memory for bilateral search," Avtomat. Telemekh., No. 4, 107 (1969).
3. R. V. Sokolov, Economical and Informational Simulation of Information Processing in Automatic Production Control System [in Russian], Leningrad State Univ. (1980).
4. A. A. Zykov, Theory of Finite Graphs [in Russian], Vol. 1, Nauka, Moscow (1969).
5. V. O. Groppen, "The minimum mesh skeleton problem," in: Modern Control Problems [in Russian], Nauka, Moscow (1974), pp. 167-168.
6. I. V. Romanovskii, Algorithms for Solving Optimization Problems [in Russian], Nauka, Moscow (1977).
7. E. Mipieka, Optimization Algorithms for Networks and Graphs, Marcel Dekker (1978).
8. V. N. Burkov, I. A. Gorgidze, and S. E. Lovetskii, Applied Problems in Graph Theory [in Russian], Metsniereba, Tbilisi (1974).
9. V. O. Groppen, "The connection between problems of maximum circulation and minimal cut in a strongly connected graph and the problem of inhomogeneous flow," in: Electronic Engineering, Vol. 9, Automatic Control Systems [in Russian], Moscow (1972), pp. 117-124.
10. V. A. Kletin, "Analysis of the efficiency of combinatorial algorithms for finding optimal trees in weighted graphs," Avtomat. Telemekh., No. 11, 134 (1979).