KNOWLEDGE-BASED TECHNOLOGY FOR SOLVING COMPLEX SYSTEMS OF INTERRELATED PROBLEMS

V. I. Gritsenko, V. A. Bogemskii, and T. M. Kucherenko

UDC 681.322

The solution of a system of problems is modeled as a search among alternative paths on a semantic network. We consider representation of procedural knowledge and modeling of decision processes by production rules and Petri nets.

Planning and control problems handled by large industrial MIS are usually highly complex multivariable and multi-alternative problems which involve a variety of criteria and coordination among many individuals and divisions in the process of planning and managerial decision making. Moreover, the dynamically changing operating conditions of the controlled systems introduce unpredictable disturbances, which require on-line correction or complete recalculation of the plans.

Apart from being complex, the solution technologies for these problems are basically deterministic. This is attributable [1] to the stability of the functional-hierarchical structure of MIS, where we usually know in advance (or can establish) the order of interaction between the component links in the process of decision making and coordination. The solution of a system of problems on the whole is a multistep process in which the choice of the next step depends on the initial state and also on the results obtained in the previous steps, i.e., the process may evolve along alternative paths depending on these factors. The number of alternatives may be fairly large, and the choice of the best alternative is not always obvious: it is determined by an expert evaluation of the situation.

Considerable attention is being devoted to the development of intelligent technologies for the solution of complex problems [1-3]. This is the topic of our paper, which examines some issues of control of the solution of complex systems of interrelated problems, using knowledge about the specific features of decision making in a particular situation.

Figure 1 shows a fragment of the solution technology for a system of train repair scheduling problems [4], which includes the following stages. The train repair plan is computed using time-to-repair standards. Then the user interactively enters an instruction to coordinate the computed repair plan with the freight plan, or to compute directly the derived demand for spare parts, or to print out the repair plan. If coordination is requested, then the freight plan must be computed in addition to the repair plan and, depending on the analysis of the coordination results, the system may continue automatically (without user intervention) to compute the demand for spare parts and then print out the plan; alternatively, it may be necessary to correct (interactively) the repair standards and to recalculate the repair plan from scratch.

In real life, of course, this scheme is much more complex, but in principle the fragment shown in Fig. 1 reflects the main types of interrelationships between the separate blocks, specifically the presence of several compulsory or alternative inputs in each block and several alternative outputs from a block. To each block is associated a program module which realizes some computational algorithm, requests data, or executes an interactive procedure. Depending on the output of this module, and also on the output of previously executed modules, the next step of the computational procedure is selected.

The flowchart shown in Fig. 1 may be represented as a semantic network (Fig. 2), in which nodes correspond to program modules from the set \( M = \{ M_1, M_2, \ldots, M_n \} \) (double lines in the network identify interactive procedures) and arcs represent the succession relation. A semantic network is closed and deterministic, because all the situations are enumerated and the succession relations are uniquely defined.* The single-valued succession relations transform the network into a directed graph, and the search procedure for an alternative solution path on this graph reduces to the identification of a directed subgraph that describes the problem-solving process.

*Strictly speaking, a semantic network remains open in the sense that when a new problem-solving technology appears, the network may be augmented with new nodes and arcs representing this technology.

Each program module $M_i$ ($i = 1, \ldots, n$) generates its own termination code $r_i \in R_i$, whose value is determined by interpreting the output of the computation or search procedure or in response to an interactively entered request from the user. In the simplest case, $R_i = \{0, 1, 2\}$, which indicates respectively that the module was not executed ($r_i = 0$), the execution ended with a satisfactory result ($r_i = 1$), or the execution terminated abnormally ($r_i = 2$). In fact, the spectrum of execution outcomes may be much broader, and for modules implementing interactive procedures a different value $r_i$ may be associated to each user response, i.e., in general $R_i = \{1, 2, \ldots, n_i\}$. The collection of these values is a vector $R = \{r_1, r_2, \ldots, r_n\}$ which describes the state of the network as a whole.

A semantic network with arcs labeled by the values $r_i$ defining alternative succession relations is in essence a model of knowledge about the solution technology of complex systems of interrelated problems under various conditions. Using this knowledge, we can create an intelligent program superstructure over the system of functional program modules which will perform an automatic search of a solution path for a given problem depending on the initial conditions and the results of each step, connecting the user to the search process when and if necessary. We will consider two variants of organization of this superstructure.

**USING A PRODUCTION SYSTEM FOR KNOWLEDGE REPRESENTATION AND PROCESSING**

The knowledge base includes three components:
- the set $\Pi = \{P_1, P_2, \ldots, P_m\}$ of production rules that determine the choice of the next step in the problem-solving process;
- the current state vector $R$ of the semantic network;
- the relation $A(M, R, T)$ which enumerates for each module the set $R_i$ of its termination codes and the corresponding message texts $T_i = \{t_1, t_2, \ldots, t_{n_i}\}$.

The interpreter program, following the principles described in [1], sequentially selects and executes rules form the set $\Pi$. Each $P_j$ implies analysis of the values of some subset of elements of the vector $R$ and selection of the next program module $M_i$ for execution, i.e.,

$$P_j : r_{xj} \land r_{zj} \land \ldots \land r_{nj} \rightarrow M_i.$$ 

The execution of $M_i$ takes the network to a new state and transforms the vector $R$ by inserting a new value $r_i$. Thus, the problem-solving process induces an evolution of the network, successively activating its nodes. The new network state in each stage is determined as $R_{k+1} = P_k R_k$, where $R_k$ is the state in stage $k$ and $P_k$ is the rule applied in that stage.