DEVELOPMENT OF A KNOWLEDGE-BASED MODEL MANAGEMENT SYSTEM

TING-PENG LIANG

University of Illinois, Urbana, Illinois

(Received December 1986; revision received September 1987; accepted January 1988)

This paper presents a software architecture and a graph-based framework for developing knowledge-based model management systems. The architecture consists of three major components: a model utilization subsystem, a modeling subsystem, and an inference engine. The core of the system is the inference engine that applies the graph-based framework to drive the processes of model integration and selection. The graph-based framework includes a model representation scheme and reasoning mechanisms. The representation scheme depicts a set of data as a node, a set of functions as an edge, and a basic model as a combination of two nodes and one connecting edge. Based on this scheme, mechanisms for model integration and selection are discussed. These mechanisms enable a model management system to create composite models automatically. A prototype implemented in PROLOG is also presented to demonstrate the graph-based framework.

Decision Support Systems (DSS) are computer-based information systems that support semistructured or unstructured decisions. Due to the complexity of these decisions, using proper models can significantly improve human performance by facilitating understanding about the decision problem, examining more alternatives, or enhancing prediction (Little 1970). Therefore, a model management system (MMS) that supports the development of decision models and their subsequent use has been considered crucial to the success of DSS (Alter 1980; Bonczek, Holsapple and Whinston 1981b; Elam 1980; Keen and Scott Morton 1978; Sprague and Carlson 1982; Stohr and Tanniru 1980).

Early research in MMS considered models as data or subroutines and proposed that an MMS must support model creation, storage, retrieval, execution, and maintenance (Sprague and Watson 1975; Will 1975). Recent research primarily focused on two issues: model base organization and model representation. A model base is a repository of decision models. On the one hand, because the model base and the data base are similar in many aspects, researchers have studied the application of data models, such as the relational model (Codd 1970), to the development of MMS (Blanning 1982, 1983, 1984, 1985; Donovan 1976). On the other hand, because of the knowledge intensive nature of models, some researchers concentrated on adopting knowledge representation techniques developed in artificial intelligence to represent models in the model base. The model representation schemes investigated include SI-nets (Elam, Henderson and Miller 1980), knowledge abstractions (Dolk 1982; Dolk and Konsynski 1984; Konsynski and Dolk 1982), predicate calculus (Bonczek, Holsapple, and Whinston 1980, 1981a), and frame-based systems (Watson 1983). Development of MMS needs both a model base organization for model storage and an appropriate technique for model representation.

In addition to these two issues, however, it is very important for an MMS to have the capabilities of model integration and selection. The capability of model integration allows an MMS to create a complicated model by integrating existing models in the model base. In this case, the models stored in the model base are stand-alone decision models and building blocks for creating new models. The capability of model selection helps the user determine what models are available to produce the requested information and then automatically selects or allows the user to select a model for execution. With these capabilities, an MMS can better support decision makers by formulating ad hoc models to meet unanticipated requirements quickly. An MMS that supports model integration and selection is called a knowledge-based MMS.

This paper introduces an expert systems approach to building such capabilities in MMS, with emphasis on the design of graph-based mechanisms for driving the process of model integration. The remainder of

Subject classification: Computers, artificial intelligence: knowledge-based systems. Information systems: model management for decision support. Networks/graphs: graph-based reasoning.

0030-364X/88/3606-0849 $01.25
© 1988 Operations Research Society of America

this paper is organized as follows. First, an architecture for MMS design is presented. The architecture is composed of three functional modules: a model utilization subsystem, a modeling subsystem, and an inference engine. Then, a graph-based approach to designing the inference engine is presented. It includes a graph-based representation scheme and mechanisms for model integration and selection. The graph-based scheme provides an interface through which algorithms and heuristics developed in artificial intelligence and graph theory can be applied to model management. Finally, TIMMS (The Integrated Model Management System), a prototype implemented in PROLOG, is described to illustrate the graph-based approach. Sample sessions are also presented. Successful implementation of the architecture and mechanisms indicates a promising integration of operations research, artificial intelligence, and DSS research.

1. Architecture for a Knowledge-Based MMS

Since the processes of model integration and selection involve reasoning and judgment, an architecture different from traditional DSS architectures must be used to develop a knowledge-based MMS. Figure 1 illustrates an architecture adapted from expert system

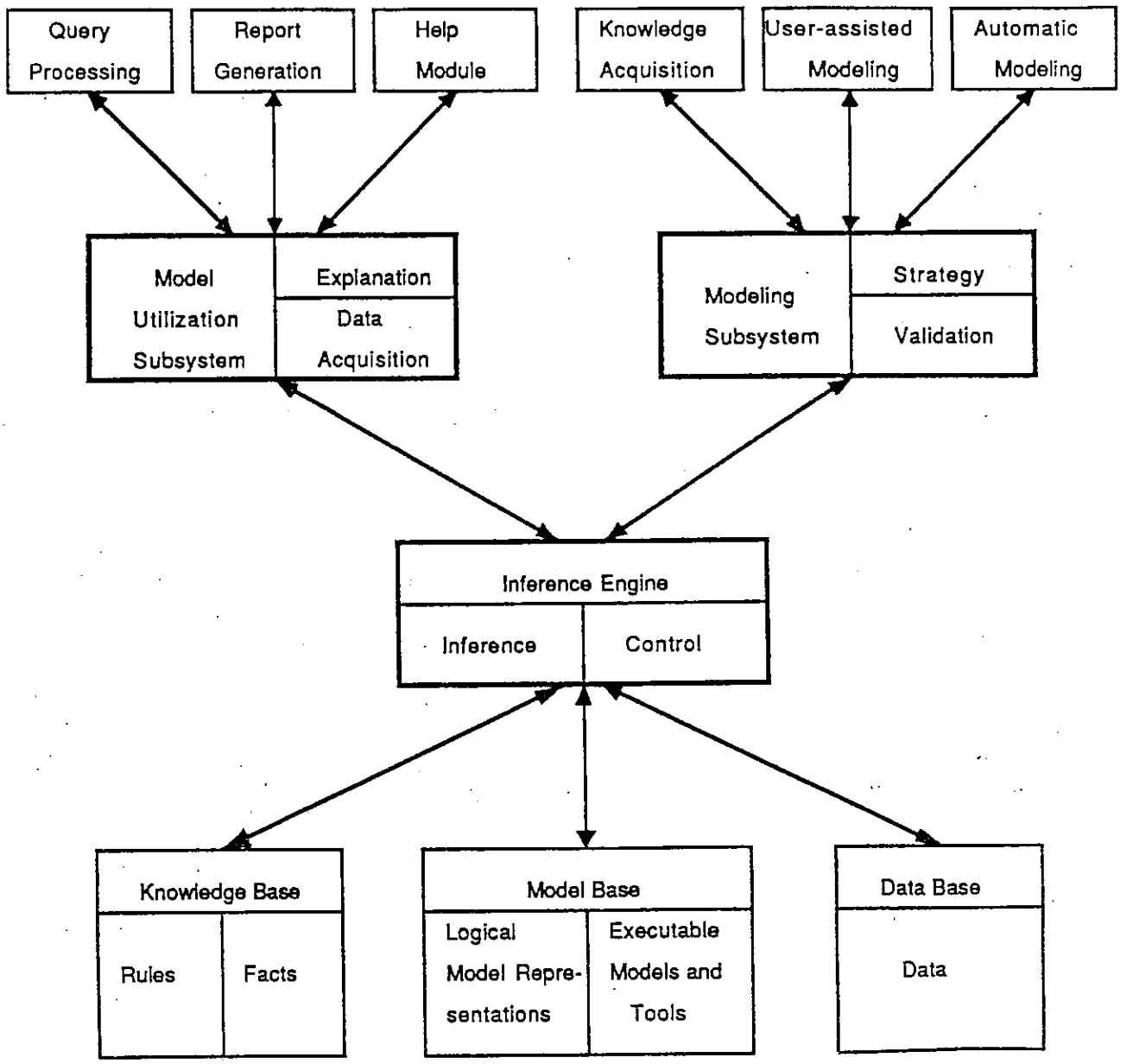


Figure 1. A software architecture for MMS.

research. The architecture includes two major subsystems: modeling and model utilization. The modeling subsystem focuses on improving the productivity of model builders, whereas, the model utilization subsystem concentrates on effective use of models. In addition, an inference engine is required to drive the processes of model integration and selection, and to integrate three basic components: model base, data base, and knowledge base. Basic models are stored in the model base; data pertaining to decision making are stored in the data base; and the knowledge regarding effective use of the models in the model base and the data in the data base is stored in the knowledge base.

To support the effective use of models, the model utilization subsystem should be able to accept user queries, report results to the user, and provide helpful messages in the course of consultation. In other words, it should have three major functions: query processing, report generation, and help. The query processor is the interface between decision makers and the system. It translates a user's query into a set of commands understood by the system. The report generator provides the requested output in a format the user prefers. The help module provides helpful messages such as how the results were generated.

The modeling subsystem is designed to support the model builder, who is responsible for developing useful models. It should have three major functions: knowledge acquisition, user assisted modeling, and automatic modeling. The MMS acquires knowledge of models, such as integrity constraints, through the model acquisition module, whereas, the model builder interacts with the user assisted or automatic modeling modules to create new models or modify existing models.

The inference engine is the heart of a knowledge-based MMS. It performs two major functions: inference and control. The model utilization or modeling subsystem translates a user's request into commands understood by the system. Then, the inference engine executes the commands, controls access to the data base and the model base, retrieves knowledge from the knowledge base, and makes inferences, if necessary. After obtaining the required information or proving that it is not available, the inference engine passes messages back to those two subsystems and then reports the result to the user.

The inference engine has three major inference mechanisms: 1) integrating the data base and the model base, 2) integrating models in the model base, and 3) controlling the execution of a selected model.

The first mechanism controls the access of the data base and the model base. For any query, the MMS first searches the data base. If the information is available in the data base, it will be retrieved. Otherwise, the system will search the model base and see whether there is a set of models available for producing the information. If there is any model available, the mechanism will check the availability of its inputs and then retrieve the inputs to execute the model (Liang 1985).

If no basic model in the model base is available to produce the required information, the second mechanism will take over and try to develop executable composite models. A detailed discussion of a graph-based mechanism will be presented in the next section.

After the model for producing the desired information is chosen, the mechanism for model execution will be activated to schedule the integrated component models and make sure that they are executed in a proper sequence.

2. A Graph-Based Inference Mechanism

The discussion in the previous section indicated that the mechanism for integrating models is at the heart of a knowledge-based MMS. Although research in MMS has increased dramatically in the past decade, little has focused on this important issue. Recently, Geoffrion proposed an approach called structured modeling, which focuses on exploring functional relationships among the modules constituting a model during the modeling process (Geoffrion 1985, 1987). The graph-based approach presented in this section is built on the concept of Geoffrion's work and deals with the following essential issues:

1. representation of models,
2. algorithm for integrating models, and
3. algorithm for selecting models.

The major difference between the proposed approach and structured modeling is that the former considers each model in the model base as a single entity, whereas the latter focuses on the structural relationships within a model. In fact, the former approach reflects a higher level abstraction on which heuristics and algorithms may be developed to automate the modeling process. Detailed discussions on different levels of abstraction can be found in (Liang 1988).

2.1. Graphical Representation of Models

Since problem solving is often described as a search through a vast maze of possibilities (Simon 1981), we can describe the process of human modeling as a

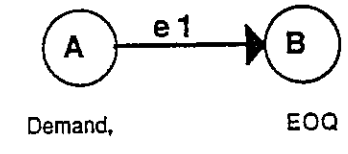
search through a number of possible relationships in order to find a route which can convert the initial state (available information) of a problem to the desired final state (output information). By this concept, models in the model base can be represented by two basic elements: nodes and edges. The modeling process can be formulated as a process that creates a directed graph and selects a path on the graph. The directed graph, called a model graph, represents all possible alternatives for solving the problem and each path in the graph represents a model. These terms can be formally defined as follows.

Definition 1. Node—. A node, N, represents a set of data attributes. It could be the inputs or the outputs of a set of models.

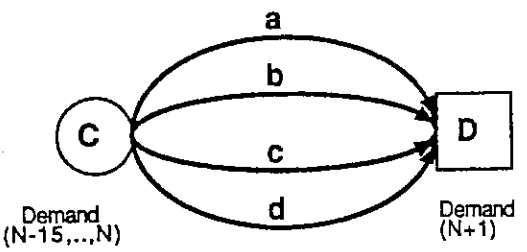
Example

In Figure 2a, node A represents a set of data including the demand, holding cost, and ordering cost. Node B represents the computed economic order quantity (EOQ).

Nodes represent states. We also need edges to indicate transformation of states.

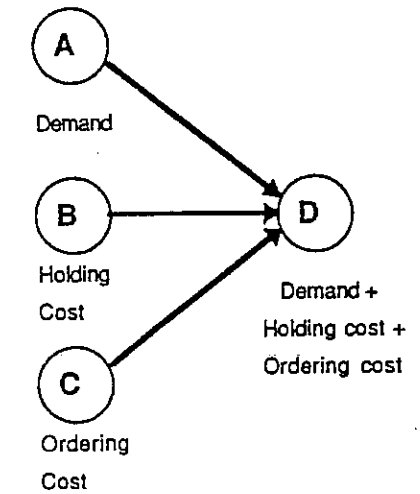


(a) Graphical Representation of the EOQ Model

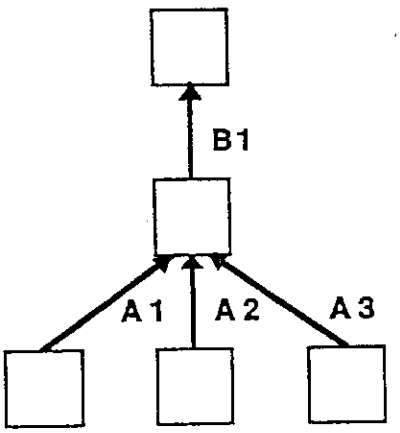


Where: a: Moving average
b: Exponential smoothing
c: Regression
d: Box-Jenkins

(b) A One-stage Modeling Process



(c) An Example of AND Nodes



Where: A1: Moving average
A2: Regression
A3: Demand function
B1: EOQ

(d) A Two-stage Modeling process

Figure 2. Graph-based representations.

Definition 2. Edge. An edge, e, represents a set of functions that convert a set of input data (the starting node of the edge) to their associated output (the ending node).

Example

The edge $e1$ in Figure 2a represents the function which computes EOQ from the demand, holding cost, and ordering cost.

Definition 3. Connectivity. Two nodes are connected if there exists at least one edge that converts the data in one node to that in another.

Example

Nodes A and B in Figure 2a are connected because edge $e1$ converts the demand, holding cost, and ordering cost in node A to the EOQ in node B.

In practical applications, both nodes and edges should be nonempty sets. Two connected nodes and one edge connecting them constitute a basic model, the smallest unit in the model base.

Definition 4. Basic Model. A basic model, M_b, is a combination of two nodes and an edge connecting the two nodes. The starting node of the edge represents the inputs of the basic model, and the ending node of the edge represents the outputs of the basic model. Hence a basic model can be represented as a triple, $\langle N1, e, N2 \rangle$.

Example

The combination of $\langle A, e1, B \rangle$ in Figure 2a is a basic model.

Each basic model in the model base is a stand-alone model, but it is also a basic element for automatic modeling. Since there is usually more than one way to convert a set of inputs to a set of outputs, the edge between two nodes may not be unique. That is, a model base may have more than one model for solving a particular problem. For example, if one wants to forecast demand for the next year based on the demand data in the last 16 years, one may use the moving average, exponential smoothing, regression, or the Box-Jenkins approach, as illustrated in Figure 2b. In other words, four basic demand forecasting models in the model base, $\langle C, a, D \rangle$, $\langle C, b, D \rangle$, $\langle C, C, D \rangle$, and $\langle C, d, D \rangle$, are available for forecasting the future demand.

In addition to the case where more than one model is available to produce a set of required outputs, it is possible that a set of basic models, in combination, produce the required outputs, but each individual model produces a subset of the required outputs. In order to differentiate these two situations, we need to define two types of nodes: AND nodes and OR nodes.

Definition 5. AND Node. An AND node, N_a, is a node that is the ending node of more than one basic model. Each model produces a subset of the required output, but the combination of these models produces the whole set of the required outputs. An AND node is true only if all edges ending at the node are true.

Example

Node D in Figure 2c is an AND node because the model $\langle A, a, D \rangle$ produces the demand information, the model $\langle B, b, D \rangle$ produces the holding cost, and the model $\langle C, c, D \rangle$ produces the ordering cost. Therefore, the three models, in combination, produce the information contained in node D, but each model produces only a subset of the information. In this paper, an AND node is represented as a circle.

Definition 6. OR Node. An OR node, N_r is a node that is the ending node of more than one basic model; each model produces the entire set of required information. An OR node is true if at least one edge ending at the node is true. In this paper, an OR node is represented as a square.

Example

Node D in Figure 2b is an OR node because there are four models ending at node D and each can produce the forecasted demand.

In the human modeling process, an AND node represents a union point where more than one set of output data is combined to formulate the required output; and an OR node represents a decision point where one or more models are selected among those available models.

Because all the four forecasting models represented in Figure 2b produce the same set of outputs, and no output of a model becomes an input of another model in the graph, it can be called a one-stage graph. However, not all modeling problems are as simple as this example. Rather, many problems may need integration of various kinds of models. By "integration," we mean that two or more models are combined to become a composite model in which the output of a model is fed into another model. For example, Figure 2d illustrates a two-stage graph. It represents an integration of the EOQ and demand forecasting models. In the figure, the output of the demand forecasting models (A1, A2, or A3) are fed into the EOQ model

(B1). Because the model base has one model for EOQ computation and three models for demand forecasting, there are three paths (1 * 3), i.e., three different models, for producing the desired information. Formal definitions of path, integrability and a composite model are as follows. Because there are two different types of nodes, the definition of a path is different from the traditional concept.

Definition 7. Path. A Path, P, is a finite sequence of edges of the form that

1. these edges are connected,
2. at each OR node, only one edge that enters the node is true,
3. at each AND node, all edges that enter the node are true.

Definition 8. Integrability. Two basic models are integrable if the input of one model and the output of the other share common data attributes.

Definition 9. Composite Model. A composite model, M_c, is a model that integrates a set of basic models.

According to the definitions previously described, the concept of a model graph and the modeling process can be defined.

Definition 10. Model Graph. A model graph, G, is a graph that represents all possible models, including basic models and composite models, for producing the requested information. Each path in a model graph represents a model. A model graph must be acyclic.

Example

Figure 2d is a model graph that represents models for computing EOQ. The model graph is composed of three composite models. For example, path A1–B1 is the model which forecasts demand by using the moving average technique (edge A1) and then computes the EOQ by using the EOQ model (edge B1).

Definition 11. Modeling Process. A modeling process is a process that includes two phases: the formulation of a model graph and the selection of one or more paths in the formulated model graph.

The modeling process is a logical process that formulates a model graph capturing all possible paths for producing the requested outputs and makes selection in the graph. Because a model graph clearly represents the relationships among relevant basic models, it becomes much easier for the system to provide advice regarding model integration and selection. Based on the model graph, an MMS may either perform model integration and selection automatically (the automatic modeling mode) or provide advice about model integration to the user and then allow the user to create composite models (the user assisted modeling mode). For implementation purposes each model graph must be acyclic. Otherwise, the modeling process may be infinite.

Each path in the graph implies an appropriate model, but it does not guarantee that the model will generate a feasible solution. For example, if a model base contains a capital budgeting model that uses the integer programming technique to determine the best combination of projects for investment, the model graph only indicates the existence of this model, but it will not be able to tell the user whether the model can produce a feasible solution until the model is actually executed.

After formulating a model graph and choosing a path in the graph, the MMS also needs a process for executing the selected path. In graphical terms, the model execution process can be defined as follows.

Definition 12. Execution Process. The model execution process is a process that activates a path and then executes the models constituting the path in an appropriate sequence in order to generate the output.

2.2. Implementation of the Graph-Based Representation

Concerning the implementation of a model representation method, a model can be portrayed by the following five categories of information: 1) outputs of the model, 2) inputs required to produce the output, 3) computational procedures used in the model, 4) integrity constraints of the model, and 5) validity of the model.

In other words, a basic model can be represented by a set of five relations; each represents a unique characteristic of the model, as follows.

INPUT (Modelname, Inputs)
OUTPUT (Modelname, Outputs)
OPERATION (Modelname, Functions)
INTEGRITY (Modelname, Constraints)
VALIDITY (Modelname, Evaluation).

These relations should be read as "the inputs of ⟨modelname⟩ include ⟨input1, input2, ..⟩," "the outputs of ⟨modelname⟩ include ⟨output1, output2, ..⟩," and so forth. The first four relations are important to the formulation of a model graph, and the fifth relation (validity relation) is important to the selection of models.

Corresponding to the graph-based representation, the input and output relations are nodes for formulating a model graph. The operation relation is represented as an edge. It specifies computational functions used in a model and is part of the interface between the logical integration of models indicated in a model graph and the actual execution of the selected model. A basic model, identified by a unique name, is a combination of one operation relation (an edge) and its associated input and output relations (two nodes).

The integrity relation of a model specifies constraints that must be satisfied before the model can be considered applicable to a specific problem. For example, the least squares linear regression technique requires that the number of cases be larger than the number of independent variables plus two. Unless this constraint is satisfied, the sales forecasting model using the regression approach should not be considered in formulating the model graph.

The validity relation indicates a measure of the fitness of a model to a particular problem. Because the validity of a model can only be assessed after it has been implemented, the validity value in the relation usually represents the historical validity of the model in a specific context. In other words, it represents a kind of subjective confidence in the model, based on a predefined model evaluation function or previous experience in that specific context. For example, in the case of forecasting future sales, our experience indicates that the accuracy of the moving average technique is poor for identifying the turning point in a trend (Chambers, Mullick and Smith 1971); the model should have a low validity value when it is considered for forecasting the turning point.

Figure 3 is a sample representation of the EOQ model. The integrity constraint of the model indicates that both the holding cost and the ordering cost must be constants in the period. If the constraints are satisfied, the validity of the model is 0.8 on a 0.0 to 1.0 scale.

2.3. Mechanism for Model Formulation

Formulation of a model graph involves an extensive search in the data base and the model base. Many heuristics have been developed for creating and traversing a search tree (see Busacker and Saaty 1965; Carre 1979; Rich 1983 or other books on graph theory or artificial intelligence for a review). These include depth-first search, breadth-first search, and best-first search. For creating a model graph, the depth-first search and the best-first search strategies are better than the breadth-first search strategy because they support both the optimizing and the satisficing modeling strategy. An MMS generates a satisfactory model in the satisficing strategy but selects the model with the highest validity in the optimizing strategy. In this section, a depth-first search algorithm for creating model graphs is presented.

```
OUTPUT(EOQModel, [Economic order quantity])
INPUT(EOQModel, [Demand, Ordering cost, Holding cost])
OPERATION(EOQModel, [EOQ subroutines])
INTEGRITY(EOQModel, [Constant(Ordering cost,
                              Holding cost)])
VALIDITY(EOQModel, [0.8])
```

Figure 3. Representation of the EOQ model.

The basic idea of the depth-first search is to pick up an alternative at every node arbitrarily and work forward from that alternative. Other alternatives at the same level are completely ignored as long as there is any hope of reaching the destination using the original choice. If the original choice is proved impossible to lead to a solution, then go back one level to work on another alternative.

Suppose a user has placed a query and the requested information is not directly available in the data base; the procedures for applying the depth-first search to formulate a model graph are as follows.

Step 1. Search OUTPUT relation in the model base to see whether there is a model that produces the output.

Step 2. If no model is found, then stop searching and report that no model is available in the model base. The system may ask the user to develop a new model.

Step 3. If a model is available, then search the INPUT relation of the model to find the input data required for execution.

Step 4. Repeat the following process until all inputs are obtained or one input is proved unavailable.

- 4.1. Pick up an input, check whether it is an output of its preceding model (check for acyclicity).
  - a. If it is true, then drop this model and go to Step 3.
  - b. If it is not true or the model does not have any preceding model, then skip this procedure.
- 4.2. Search the data base for availability.
  - a. If the input is available in the data base, then retrieve its value and go to Step 4.1.
  - b. If the input is not available in the data base, then go to Step 4.3.

4.3. Search the OUTPUT relation of the model to see whether it can be produced by a model in the model base.
   a. If no model is available, then go to Step 4.4.
   b. If a model is found, then go to Step 4.5.
4.4. Prompt the user for the input.
   a. If it is provided by the user, then obtain its value and go to Step 4.1.
   b. Otherwise, drop the model.
4.5. Search the INPUT relation of the model to find input data required for execution. Repeat Step 4 until all input data have been obtained or one input is proved unavailable.

Step 5. If all input data are available, then check integrity constraints.

5.1. If any integrity constraint is not satisfied, then drop the model.
5.2. If all constraints are satisfied, then add the model to the model graph.

Step 6. Check whether there is another model for producing the desired information.

6.1. If there is another model, then go to Step 3.
6.2. Otherwise, stop the process and then provide advice based on the formulated model graph.

Figure 4 illustrates the process for formulating a model graph. The circled numbers in the figure are corresponding steps. A proof of the generality of this mechanism is presented in Appendix 1.

The procedures of the best-first search are basically the same as that of the depth-first search, except that the former employs an evaluation function to evaluate the potential of all possible paths before further investigation and gives higher priority to better paths in order to make sure that models with higher validities will be examined earlier. There are certainly other possible approaches for building model graphs. They will not be discussed here, however, because they may be derived from the procedures described before.

In this mechanism, if the operation that picks up an input of a model and searches for the availability of the specific input is considered a basic operation in the model base and represented as an edge, then the formulated model graph will be an alternate AND/OR tree.

Definition 13. Tree. A tree, T, is a graph containing one or more nodes such that

1. there is a specially designed node called a root,
2. the remaining nodes are partitioned into n ($n \geq 0$) disjoint sets $T1, \ldots, Tn$ where each of these sets is also a tree. $T1, \ldots, Tn$ are called the subtrees of the root.

Definition 14. AND/OR Tree. An AND/OR tree is a tree that includes both AND nodes and OR nodes.

Definition 15. Alternate AND/OR Tree. An alternate AND/OR tree is a tree in which the AND node and OR node appear at alternate levels. In other words, if nodes at level m are AND nodes, then the nodes at level $m + 1$ must be OR nodes.

Example

Figure 5 illustrates an alternate AND/OR graph. It is the model graph formulated by the above algorithm for providing advice about the EOQ and demand forecasting problem described in the first section.

Proposition 1. *The model graph formulated in the above algorithm is an alternate AND/OR tree.*

A proof of this proposition is given in Appendix 2.

2.4 Strategies for Model Selection

Given the mechanism for formulating model graphs, there are two different strategies for providing advice: optimizing and satisficing. The optimizing strategy requires that an MMS formulate a complete model graph and then evaluate all paths in the graph to find the best alternative. If validities of all models in the model graph are available, then the optimizing strategy is simply to maximize the validity of the selected path. This can be formulated as a maximum validity flow problem subject to the constraints of modeling time, modeling costs, and other considerations. In this case, algorithms may be applied to find the path with the highest validity (i.e., the best composite model). Although the optimizing strategy guarantees that, given the criteria, the formulated model is the best available, the combinatorial explosion in the model graph formulation process sometimes makes it unrealistic for a system to develop a complete model graph, and hence, forces a system to adopt the satisficing strategy. For example, in a model graph with b branches at each node and d levels of depth, a depth-first iterative deepening algorithm would take $O(b^d)$ time to find the optimum solution (Korf 1987).

The satisficing strategy, on the other hand, requires that each path be evaluated immediately after it is found and accepted if it is satisfactory. Therefore, formulation of a complete model graph may not be necessary. In this strategy, the MMS follows the same process for formulating a model graph except that every path is evaluated at the time it is formulated. If a satisfactory path has been found, the process for formulating the model graph is terminated and the

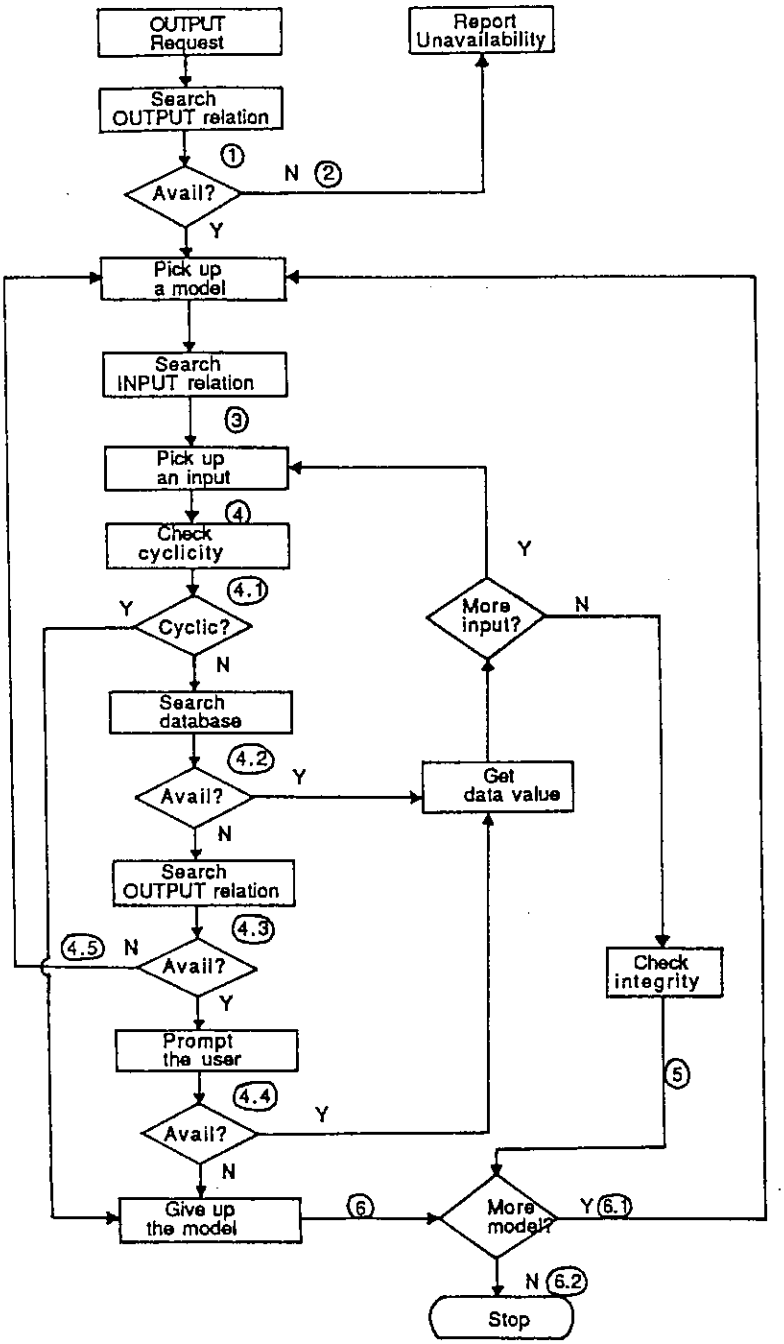


Figure 4. Process for formulating a model graph.

path is chosen to produce the desired information. Figure 6 briefly illustrates the modeling process for implementing the satisficing strategy.

A major issue in implementing a satisficing strategy is the development of model evaluation functions. If more than one model is applicable to a specific problem, the MMS needs their validity values to make the selection automatically. For implementation considerations, we need a quantitative measure of validity to facilitate model selection in a model graph. Therefore, a model evaluation function that determines the validity of a model based on a set of predetermined criteria is required. In general, the following three issues should be considered in developing model evaluation functions.

1. What are proper criteria for determining validity value? There are at least five possible criteria:

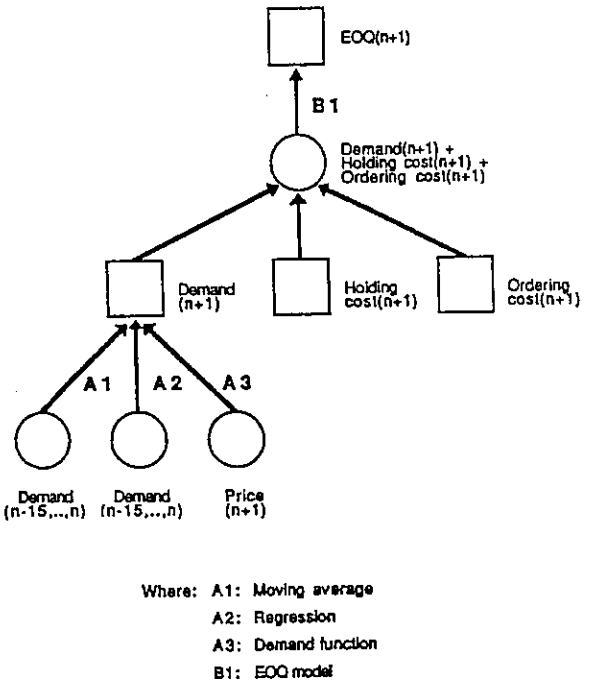


Figure 5. An alternate AND/OR graph.

1) accuracy of the model, 2) the user's preference for the model, 3) distance from producing the desired information, 4) number of models integrated, and 5) total cost.

2. How can several validity values be combined to get the overall evaluation of a composite model?
3. When should a model be evaluated?

A detailed discussion on these issues can be found in Liang and Jones (1988). No matter whether the satisficing or optimizing strategy is chosen, heuristics may be used to reduce the complexity of the process. The following is a sample heuristic for reducing complexity. More discussion on search heuristics can be found in Pearl (1984).

Step 1. Determine the validity of each edge (a model) in the model graph. The system determines the validity of each selected member model by searching the VALIDITY relation of the model or executing the evaluation function if appropriate.

Step 2. Simplify the problem by removing dominated alternatives. If more than one edge is connecting two nodes, i.e., more than one model is available to convert a set of inputs to its associated outputs, then select the one with the highest validity and ignore the rest.

Step 3. Calculate validities for all possible paths from the initial state to the final state. If the validity value represents the applicability of a particular component model to the decision problem, then the overall validity of a path may be calculated by multiplying the validities of its member edges. There are certainly other validity calculi. For example, if the validity value measures the computational cost for each component model, then the overall validity of the path may be the sum of the validities of its member edges.

Step 4. Select the path with the highest validity. If a complete model graph is formulated, the selection may be constrained by some other nontechnical constraints, such as machine memory, modeling time and so forth. In this case, it may need an integer program to determine which path is the best one. Because of the screening procedures conducted in Steps 1–3, the new formulation should be more efficient than the original. In the case where the satisficing strategy is adopted, the complete model graph does not exist and the selection process is to compare the validity of a path with a predetermined cutoff value. If the validity is higher than the cutoff value, the path will be selected. Otherwise, the modeling process continues.

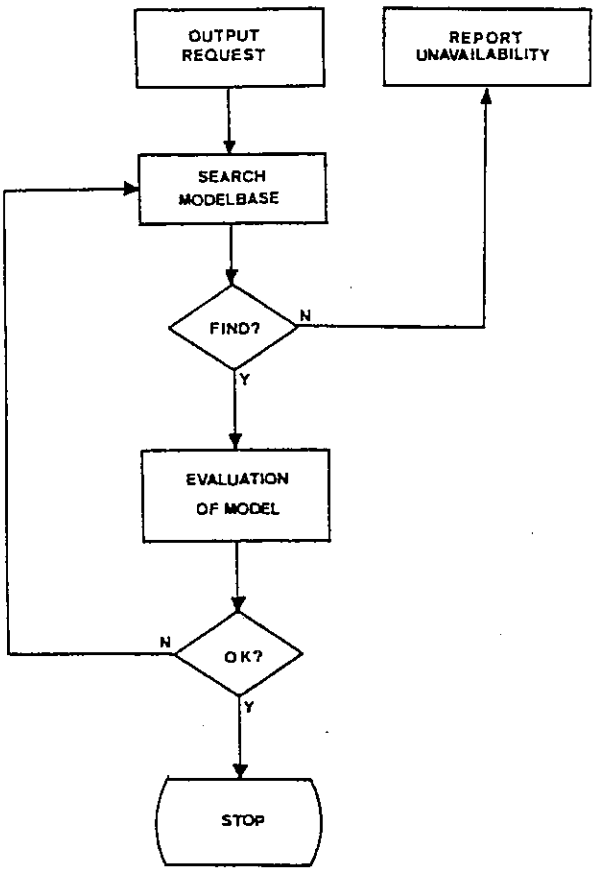


Figure 6. Process of the satisficing strategy.

3. Implementation of the Framework

The graph-based framework described in the previous section provides a basis on which capabilities of model integration and selection can be built in a knowledge-based MMS. In this section, a prototype implemented in PROLOG, called TIMMS (The Integrated Model Management System), is presented to demonstrate the feasibility of the framework. The system supports the following functions.

1. Retrieval of data in the data base.
2. Retrieval and execution of models in the model base.
3. Formulation and execution of composite models.

3.1. Architecture of TIMMS

The architecture of the system is quite similar to that illustrated in Figure 1. It has two major subsystems: model utilization and modeling. A graph-based inference engine drives the integration among models and between the model and the data bases.

3.2. Model Utilization Subsystem

The model utilization subsystem has three major functions: query processing, report generation, and help. TIMMS provides an interactive query language, TQL (The Query Language), in which a user can access both the data base and the model base without the need to identify where the information is stored beforehand.

TQL is a SEQUEL-like language. The system maintains a data dictionary to facilitate understanding of terms used by users. In using TQL, a user first specifies the category of the required output, such as "sales." Then, the system will retrieve the associated attributes from the data dictionary (for "sales" these might be "product" and "year"), prompt for their values and process the query. For example, the query for getting the information, "sales for toys for 1987," is as follows:

SELECT: *sales*
WHERE
  PRODUCT = *toy*
  YEAR = *1987.*

In addition, TIMMS has a report generator for producing formatted information, and a help module to provide information if requested.

3.3. Modeling Subsystem and the Inference Engine

TIMMS [illegible] the graph-based framework as its PROLOG, a logic-based programming language, the theoretical foundation for inference is predicate logic. The graph-based inference mechanism is on top of the predicate logic.

In order to implement the alternate AND/OR tree, TIMMS uses a recursive structure named "path." A technical description of the path structure is given in Appendix 3. By implementing the path structure, the system supports the formulation of model graphs and a simple satisficing strategy that the system will provide advice based on the first alternative available. If the user does not like the first piece of advice and asks for more, the system will then provide the next alternative, if available, to the user. This process can go on until no alternative is available.

Figure 7 is a sample consultation session for integrating the EOQ and demand forecasting models to produce the EOQ for product "*a*" for 1987. Analysis of a more complicated production scheduling model composed of 18 basic models is available in Liang (1986).

In this example, the user specifies the desired information, the system first searches the data base and finds that it is not available in the data base. Then, the system searches the model base, formulates a model graph as previously illustrated in Figure 5, and then informs the user that the integration of the EOQ model and a demand forecasting model will be able to generate the desired information. The user may accept that advice and execute the composite model, as shown in the session, or request more advice. The path structure for this query is illustrated in Appendix 3.

4. Concluding Remarks

One of the most important but difficult research issues in developing model management systems is how to develop model integration and selection capabilities. With these capabilities, a knowledge-based MMS can integrate basic models in the model base to formulate ad hoc decision models. This paper presents a graph-based approach to building such capabilities. Development of this approach and the successful implementation of the prototype presented in this article indicate the feasibility of automating the modeling process.

To build a real-world application, however, follow-up research needs to be conducted. For example, the model evaluation function has a significant impact

Table I
Path Structure

```
[path(eoq(a, 1987, X),
   [path(h_cost(a, 5), [database], []),
    path(o_cost(a, 20), [database], []),
    path(demand(a, 1987, Y),
      [path(demand(a, 1986, 158), [database], []),
       path(demand(a, 1985, 145), [database], []),
       path(demand(a, 1984, 132), [database]), [])],
    M4)],
M1),    /* First Path */
path(eoq(a, 1987, X),
  [path(h_cost(a, 5), [database], []),
   path(o_cost(a, 20), [database], []),
   path(demand(demand(a, 1987, Y),
     [path(demand(a, 1986, 158), [database], []),
      path(demand(a, 1985, 145), [database], []),
      path(demand(a, 1984, 132), [database], []),
      path(demand(a, 1983, 123), [database], [])],
    M3],
M1),    /* Second Path */
path(eoq(a, 1985, X),
  [path(h_cost(a, 5), [database], []),
   path(o_cost(a, 20), [database], []),
   path(demand(a, 1987, Y),
     [path(price(a, 10), [user], [])],
   M2)],
M1)].    /* Third Path */
```

that the price is retrieved from the data base; whereas the "[]" (nothing in the square bracket) means that no model execution is required since no model is included in the input path. The path structure illustrated in Table I represents the model graph shown previously in Figure 5.

This structure includes three paths that produce EOQ for product "*a*" for 1987: integrating models M4 and M1, integrating models M3 and M1, and integrating models M2 and M1. In the path structure, "[user]" means that the information is provided by the user.

Acknowledgment

The author thanks James C. Emery, Paul Kleindorfer, Christopher V. Jones, Steven O. Kimbrough, and an anonymous associate editor for their helpful comments.

References

ALTER, S. 1980. *Decision Support Systems: Current Practices and Continuing Challenges.* Addison-Wesley, Reading, Mass.

BLANNING, R. W. 1982. A Relational Framework for Model Management in Decision Support Systems. *DSS-82 Transactions* 16–28.

BLANNING, R. W. 1983. Issues in the Design of Relational Model Management Systems. *AFIPS Conference Proceedings* 395–401.

BLANNING, R. W. 1984. Conversing With Management Information Systems in Natural Language. *Commun. ACM* 27, 201–207.

BLANNING, R. W. 1985. A Relational Framework for Joint Implementation in Model Management Systems. *Decision Support Syst.* **1**, 69–82.

BONCZEK, R. H., C. W. HOLSAPPLE, AND A. B. WHINSTON. 1980. The Evolving Roles of Models in the Decision Support Systems. *Decision Sci.* **11**, 337–356.

BONCZEK, R. H., C. W. HOLSAPPLE AND A. B. WHINSTON. 1981a. The Evolution from MIS to DSS: Extension of Data Management to Model Management. In *Decision Support Systems*, M. J. Gintzberg and E. A. Stohr (eds.). North-Holland, Amsterdam.

BONCZEK, R. H., C. W. HOLSAPPLE AND A. B. WHINSTON. 1981b. *Foundations of Decision Support Systems.* Academic Press, New York.

BUSACKER, R. G., AND T. L. SAATY. 1965. *Finite Graphs and Networks: An Introduction with Applications.* McGraw-Hill, New York.

CARRE, B. 1979. *Graphs and Networks.* Oxford University Press, Oxford, UK.

CHAMBERS, J. S., S. K. MULLICK AND D. D. SMITH. 1971. How to Choose the Right Forecasting Techniques. *Harvard Business Review* **49**, 55–64.

CODD, E. F. 1970. A Relational Model of Data for Large Shared Data BAnks. *Commun. ACM* **13**, 377–387.

DOLK, D. R. 1982. The Use of Abstractions in Model Management. Ph.D. dissertation, University of Arizona.

DOLK, D. R., AND B. R. KONSYNSKI. 1984. Knowledge Representation for Model Management Systems. *IEEE Transact. Software Eng.* **SE-10**, 619–628.

DONOVAN, J. J., AND S. E. MADNICK. 1977. Institutional and Ad Hoc DSS and Their Effective Use. *Data Base* **8**, 79–88.

ELAM, J. J. 1980. Model Management Systems: A Framework for Development. In *Proceedings of the 1980 SEAIDS* 35–38.

ELAM, J. J., J. C. HENDERSON, AND L. W. MILLER. 1980. Model Management Systems: An Approach to Decision Support in Complex Organization. In *Proceedings of the First International Conference on Information Systems.* 98–110.

GEOFFRION, A. M. 1985. Structured Modeling. Monograph. Graduate School of Management, University of California, Los Angeles.

GEOFFRION, A. M. 1987. Introduction to Structured Modeling. *Mgmt. Sci.* **33**, 547–588.

JONES, C. V. 1984. *Graph-Based Models.* Ph.D. dissertation, Cornell University.

KEEN, P. G. W., AND M. S. SCOTT MORTON. 1978. *Decision Support Systems: An Organizational Perspective.* Addison-Wesley, Reading, Mass.

KONSYNSKI, B. R., AND D. R. DOLK. 1982. Knowledge Abstractions in Model Management. *DSS-82 Transactions.* 187–202.

KORF, R. E. 1987. Planning as Search: A Quantitative Approach. *Artif. Intelligence* **33,** 65–88.

LIANG, T. P. 1985. Integrating Model Management With Data Management in Decision Support Systems. *Decision Support Syst.* **1,** 221–232.

LIANG, T. P. 1986. Toward the Development of a Knowledge-based Model Management System. Ph.D. dissertation, University of Pennsylvania.

LIANG, T. P. 1988. Reasoning in Model Management Systems. *Proceedings of the 21st Hawaii International Conference on System Sciences.* Hawaii. **III,** 461–470.

LIANG, T. P., AND C. V. JONES. 1988. Meta-design Considerations in Developing Model Management Systems. *Decision Sci.* **19:1,** 72–92.

LITTLE, J. D. C. 1970. Models and Managers: the Concept of a Decision Calculus. *Mgmt. Sci.* **16,** B466–B485.

PEARL, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, Reading, Mass.

RICH, E. 1983. *Artificial Intelligence.* McGraw-Hill, New York.

SIMON, H. A. 1981. *The Science of the Artificial.* MIT Press. Cambridge, Mass.

SPRAGUE, R. H., AND E. D. CARLSON. 1982. *Building Effective Decision Support Systems.* Prentice-Hall, Englewood Cliffs, N.J.

SPRAGUE, R. H., AND H. J. WATSON. 1975. Model Management in MIS. In *Proceedings of the 7th National AIDS Meeting.* 213–215.

STOHR, E. A., AND M. R. TANNIRU. 1980. A Database for Operations Research Models. *Int. J. Policy Anal. Inform. Syst.* **4,** 105–121.

WATSON, G. W. 1983. *Knowledge Base Management for Model Management.* Master's thesis, Naval Post Graduate School.

WILL, H. J. 1975. Model Management Systems. In *Information Systems and Organization Structure,* pp. 467–482, E. Grochla and N. Szyperski (eds.). Walter de Gruyter, Berlin.