On Structured Modeling: A Model Management Perspective

Ting-peng Liang

College of Commerce and Business Administration
Bureau of Economic and Business Research
University of Illinois, Urbana-Champaign

BEBR

FACULTY WORKING PAPER NO. 1385

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

August 1987

On Structured Modeling: A Model Management Perspective

Ting-peng Liang, Assistant Professor
Department of Accountancy

Abstract

Recently a framework of structured modeling has been presented by Geoffrion (1987). This note expands his frameowrk by addressing two issues: cognitive considerations and automatic modeling. First, from a model management perspective, it may be unnecessary to decompose every model into its elemental structure. In addition, the decomposition process should not affect a model's cognitive meaning to the user. Second, in order to support automatic modeling, a higher level model abstraction is needed. This abstraction provides an interface through which algorithms and heuristics in graph theory can be applied to automate a modeling process.

1. Introduction

In a recent article, Geoffrion (1987) presents a framework of structured modeling to provide a computer-based environment for conceiving, representing, and manipulating a wide variety of models. The framework uses a hierarchically organized, partitioned, and attributed acyclic graph to represent the semantic as well as mathematical structure of models. In the framework, models are represented at three different levels of abstraction: elemental structure, generic structure, and modular structure. The author argues that the structured modeling system provides a kernel of a model management system.

This note expands Geoffrion's structured modeling framework and discusses two issues crucial to the application of the framework to model management. First, in many situations, decomposing a model into its elemental level may be unnecessary. This is particularly important when we have a large number of models in the model base and a limited amount of computing resource. Therefore, cognitive factors must be considered in determining the bottom line for model decomposition. Second, from the perspective of decision support, it is important for a model management system to have automatic modeling capabilities that automatically integrate several existing models to provide ad hoc support. In order to achieve this goal, a higher-level model abstraction built on top of the structured modeling framework is needed. Only at this level of abstraction, algorithms and heuristics in graph theory and artificial intelligence can be applied to manipulating models. In the remainder of this article,

factors affacting the bottom line for model decomposition will be first discussed. Then, a graph-based abstraction appropriate for automatic modeling follows.

2. Cognitive Considerations in Model Decomposition

One of the reasons for model management is to reduce redundancy in a conventional modeling environment. Therefore, large models are decomposed into functionally dependent, lower-level submodels for storage and model sharing. If at least one input of model A is among the outputs of model B, then model A is called functionally dependent on model B. It is similar to the calling sequence defined in Geoffrion (1987). The lowest level models that actually stored in executable forms in a model base are called basic models. One important issue in this decomposition process is to determine when the decomposition should be terminated, that is, to determine basic models. In the structured modeling system, every model is decomposed into its elemental structure. This certainly provides much insight into a model. From a model management perspective, however, it may be inefficient in terms of system execution and also unnecessary in many cases.

Since the primary purpose of modeling is to improve human decision making, one general guideline for decomposition would be to support user cognitive models corresponding to the mathematical models to be decomposed (Liang and Jones, 1987). Due to many inherent cognitive limitations, human beings tend to store and retrieve knowledge in chunks (Simon, 1981). Therefore, the decomposition must comply with the way users orgainze the

knowledge about a model. In other words, the decomposition of a model should not affect the cognitive meaning of the model to the user.

An inappropriate decomposition may damage the cognitive meaning of a model. The well-known experiment conducted by De Groot (1966) is an example indicating that, to chess masters, knowledge about a game is stored in patterns, rather than a "television scan" of every position. These patterns are basic models; further decomposing them into pieces will lose the information the patterns contain.

Taking this fact into consideration, we can re-examine the feedmix model and the multi-item EOQ model presented in Geoffrion (1987). In brief, from the model management perspective, the feedmix model should be considered as a whole, whereas the multi-item EOQ model can be further decomposed.

Further decomposition of the feedmix model into separate components, such as total cost (objective function) and minimum daily requirements (constraints), will lose the cognitive meaning of the model, because the model works only when both present. If we solve these two components separately by their calling sequence and then combine the two solutions, the final solution may not be the same as the one obtained from the original feedmix model. The major reason here is that the LP solver can be activated only when both present. In other words, although the total cost and minimum daily requirements look decomposable in the genus graph (Figure 3 in Geoffrion 1987), they are actually not, just like pieces in a chess pattern.

The multi-item EOQ model, however, can be further decomposed into five smaller models without losing its original contents. Figure 1 shows the relationships among the five submodels: total cost, item cost, setup cost, carrying cost, and frequency model. A sequential execution of these models will generate a solution the same as what obtained from the original model. In this case it is decomposable.

INSERT FIGURE 1

Therefore, in addition to the graphical representations of the components constituting a model, the structured modeling framework may need to differentiate decomposable genus graphs from non-decomposable ones. In general, two criteria are applicable. The first is solver retrievability. Although solvers are usually not given a major concern in discussing model management, some solvers (such as an LP algorithm) are so strong that they become part of the cognitive model to the user and provide a natural bottom line for decomposition. Any decomposition should not damage the linkage between the model and this kind of powerful solvers.

The second criterion is the nature of application. If a model is decomposable and some of its submodels may be combined with other models to become a new model for solving another problem, then the model should be decomposed into a lower level in order to minimize redundancy in modeling effort and to help model sharing. If such opportunities do not exist, then it may be more efficient to store the whole model as a basic model and not to

decompose it, even if it is decomposable.

3. Automatic Modeling

Given the basic models available in a model base, it becomes possible for a model management system to perform automatic modeling that integrates basic models to formulate larger ones for supporting ad hoc decision making. The automatic modeling process involves three major tasks: (1) identify appropriate basic models, (2) determine their functional dependent relationships (i.e., calling sequences) by matching input attributes and output attributes, and (3) select one model if there exists more than one alternative.

One way to deal with these tasks is to take advantage of the algorithms and heuristics available in graph theory and artificial intelligence literature. In fact, this is one of the major motivations for developing a graph-based framework. In order to take advantage of this knowledge, a higher level model abstraction built on top of the structured modeling framework is needed.

In structured modeling, each model is represented as a collection of data attributes (called schema) without explicitly pointing out inputs and outputs. Actually a model can also be considered as a mapping from a set of input data attributes to a set of output data attributes or an operator that bridges two states: input and output.

If we use a node to represent a set of data attributes and an arc to represent a mapping function from an input node to an

output node, then each basic model can be represented as a combination of two nodes and one arc connecting the two nodes. This significantly simplifies the genus tree and modular tree of a model. For example, the multi-item EOQ model, if considered as a basic model, can be represented as in Figure 2. In a text form, the multi-item EOQ model can be represented as {multi_item_EOQ, [D(n),H,F,Q], [FREQ,SETUP$,CARRY$,ITEM$,TOT$]}, which means it serves as a bridge between [D(n),H,F,Q] and [FREQ,SETUP$,CARRY$,ITEM$,TOT$].

INSERT FIGURE 2

Since problem solving is often described as "a search through a vast maze of possibilities" (Simon, 1981), we can define the automatic modeling process as a process by which a computer-based model management system searches its model base to find proper basic models and then organizes the selected basic models in a way that can effectively convert the initial state of a problem to the desired goal state. By this definition, every basic model is considered an operator and the initial state represents the information available in the problem and the goal state represents the information desired for problem solving. At this level of model abstraction, we can define the three tasks involved in automatic modeling.

1. Identify appropriate basic models

A basic model is considered a candidate component for modeling if the model bridges two states that reduce the difference between the initial state and the goal state of the

decision.

2. Determine functional dependencies

In the case where more than one basic model is identified as candidate component, the following rule can be used to determine functional dependencies:

> If the input state of a model includes a nonempty set of elements that is a subset of the output state of another model but is not a subset of the input state of the latter model, then the former model is functionally dependent on the latter one.

For example, suppose we have a single-item EOQ model, {EOQ, [D(n),H,F], [Q]}, then the multi-item EOQ model is functionally dependent on it because there exists a set of element, [Q], in the input state of the multi-item EOQ model, which is a subset of the output state but not a subset of the input state of the single-item EOQ model.

3.Select one model from alternatives

After identifying basic components for modeling and the functional dependencies among them, a graph, called model graph, can be formulated and the modeling process can be defined as a process by which a path that connects the initial and final states can be found. Depending upon the criteria used, this process can be formulated as either a maximum flow or a shortest path problem.

If every basic model is given a validity measure and the goal of the modeling process is to find a path with the highest overall validity, then the process is a maximum flow problem. If the objective is to optimize the execution efficiency of the formulated ad hoc model and each model is given a measure of

execution time, then the process becomes a shortest path problem. After formulating a maximum folw or shortest path graph, the algorithms and heuristics in graph theory and artificial intelligence can be applied to automate the modeling process.

The following example illustrates how this graph-based abstraction works in an automatic modeling process.

[Example] Assume that we have the following basic models in our model base:

(1) a single-item EOQ model

{EOQ, [D(n),H,F], [D(n),H,F,Q]},

(2) a multi-item EOQ model

{Multi_item_EOQ, [D(n),H,F,Q], [FREQ,SETUP$,CARRY$, ITEM$,TOT$]}.

(3) a demand forecasting model that uses moving average approach to forecast demand for year n by demands of the past 10 years,

{Moving_avg, [D(i)| i = n-1,...,n-10], [D(n)]},

(4) a demand forecasting model that uses regression approach to forecast demand for year n by demands of the past 10 years,

{Regress, [D(i)| i = n-1,...,n-10], [D(n)]}.

Further assume that the data base already contains data of H, F, n, and D(i), i = n-1, ..., n-10 and the decision maker needs data of D(n), H, F, Q, FREQ, SETUP$, CARRY$, ITEM$, and TOT$. That is,

(1) Initial state = [H,F,D(i)| i = n-1,...,n-10],

(2) Goal state = [D(n),H,F,Q,FREQ,SETUP$,CARRY$,ITEM$,TOT$].

Developing a model to support the decision maker is now equivalent to finding a path that can bridge the difference between the initial state and the desired goal state. In this

case, the difference is [D(n),FREQ,SETUP$,CARRY$,ITEM$,TOT$]. In formulating the model graph, the following heuristics are used:

H.1: If there exists more than one model that eliminates at least part of the difference, then apply the one that eliminates the largest number of elements in the difference list;

H.2: If more than one model is selected by H.1 and a model is functionally dependent on others, then apply the model first.

INSERT FIGURE 3

Based on these two heuristics, Figure 3 shows the difference elimination process. Multi_iten_EOQ is first applied because it can remove four elements from the difference list, then EOQ is used because it is functionally dependent on Regress or Moving_avg. Finally, both Regress and Moving_avg are applied to remove the last element in the difference list because there is no rule to break the tie. The resulting model graph, which includes two paths, is shown in Figure 4.

INSERT FIGURE 4

Selecting one path between the two is easy and does not need any complex algorithm. When the number of alternative paths increases, however, selecting a path in a model graph may become very sophisticated and, therefore, need to use algorithms developed in graph theory. In this example, suppose we want to maximize the validity of the formulated model and already have the validity values of Regress and Moving_avg, which are 0.8 and

0.5 respectively, then this is a maximum flow problem and the the darkened route in Figure 4 will be the one selected for the decision maker. This path means that the decision maker can obtain the desired information by executing Regress, EOQ, and Multi_item_EOQ sequentially.

4. Conclusions

This note has expanded Geoffrion's structured modeling framework to the model management domain. Two issues have been discussed. First, the concept of basic models must be considered in model decomposition processes. Human cognition and solver retrievability usually define a bottom line for decomposition. Therefore, from a model management perpective, it may not be necessary to represent every model in the elemental level. Second, in order to support automatic modeling, a higher level model abstraction is needed. It uses nodes and arcs to represent sets of data attributes and mappings between nodes respectively. Models are defined as bridges that connect two different states and problem solving is considered a process by which a path can be found to eliminate the difference between the initial state and the desired goal state. Building this high-level abstraction on top of structured modeling opens the door to a new area of research in which algorithms and heuristics in graph theory can be applied to automate a modeling process.

References

De Groot, A.D., "Perception and Memory versus Thought: Some Old Ideas and Recent Findings," in B. Kleinmuntz (ed.) Problem Solving, John Wiley, New York, NY, 1966, 19-50.

Geoffrion, A.M., "An Introduction to Structured Modeling," Management Science, 33, 5, (May 1987), 547-588.

Liang, T.P. and Jones, C.V., "Meta-Design Considerations in Developing Model Management Systems," forthcoming in Decision Sciences, 1987.

Simon, H.A., The Sciences of The Artificial, Second Edition, MIT Press, Cambridge, MA, 1981.

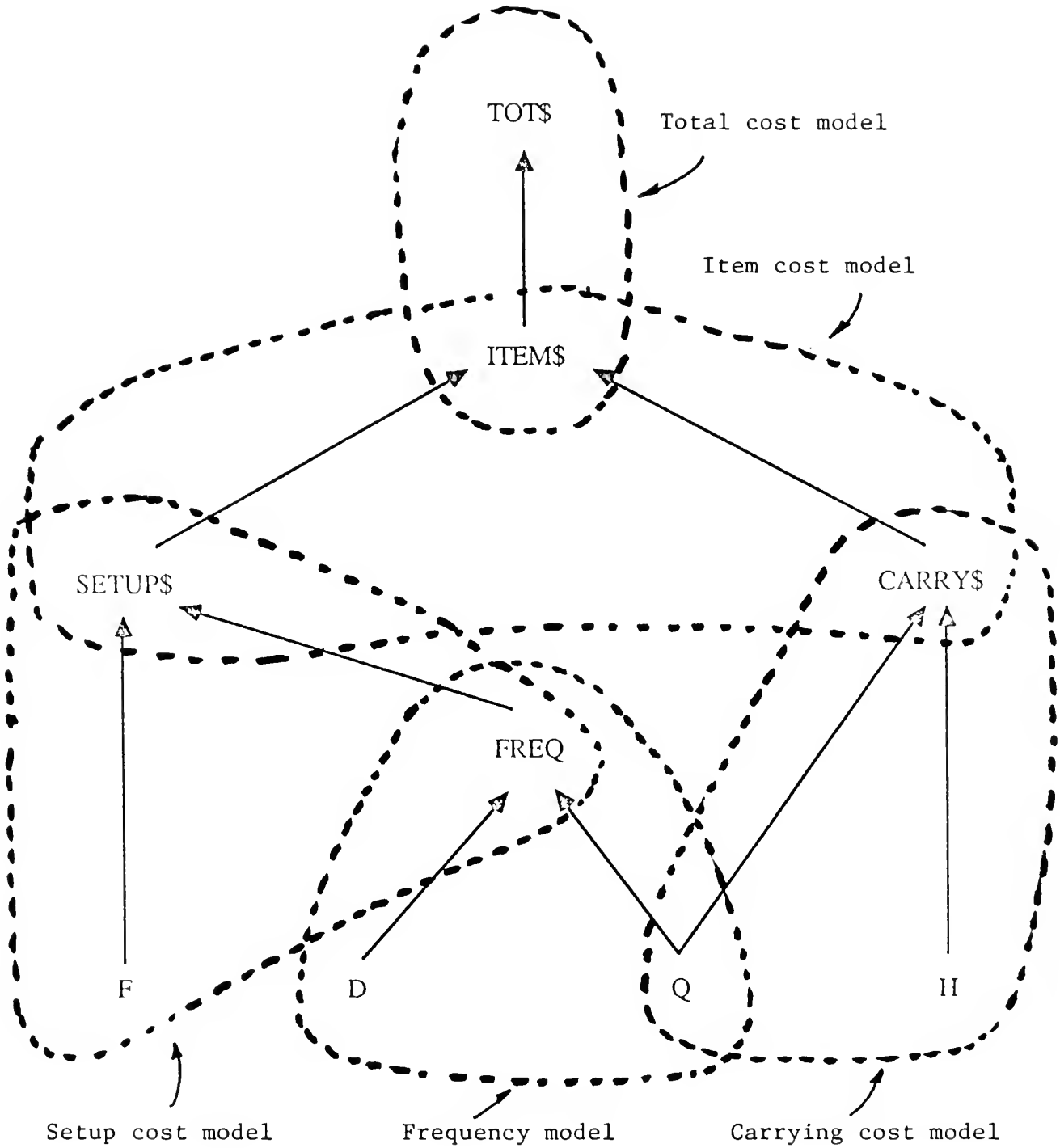

Figure 1. Decomposition of Multi-item EOQ Model

[D(n),H,F,Q]

[FREQ,SETUP$,carry$,
item$,TOT$]

Figure 2. A Higher-level Representation
of Multi-item EOQ Model

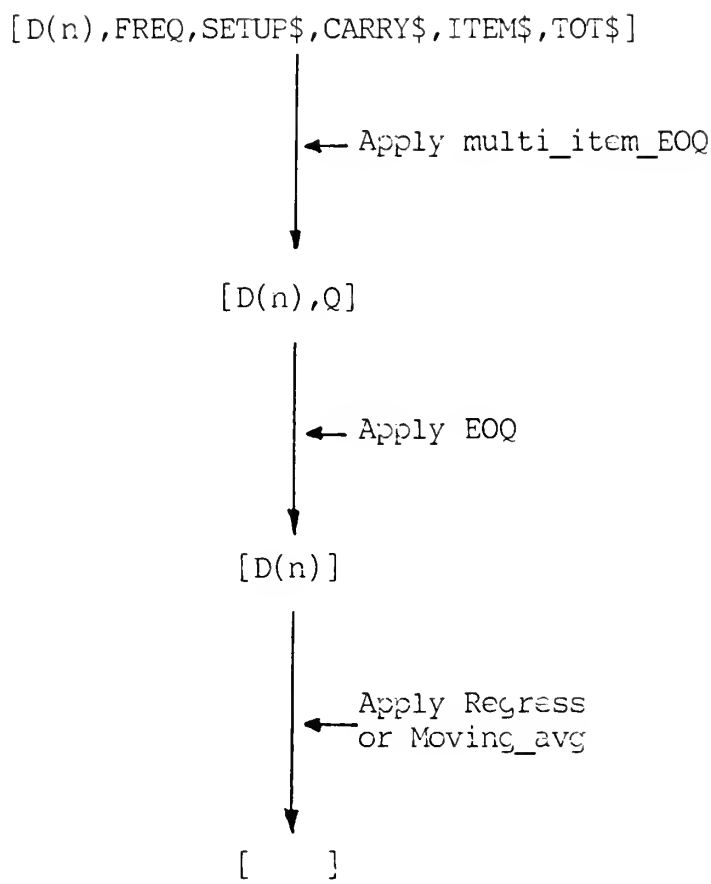

Figure 3. Process of Difference Elimination

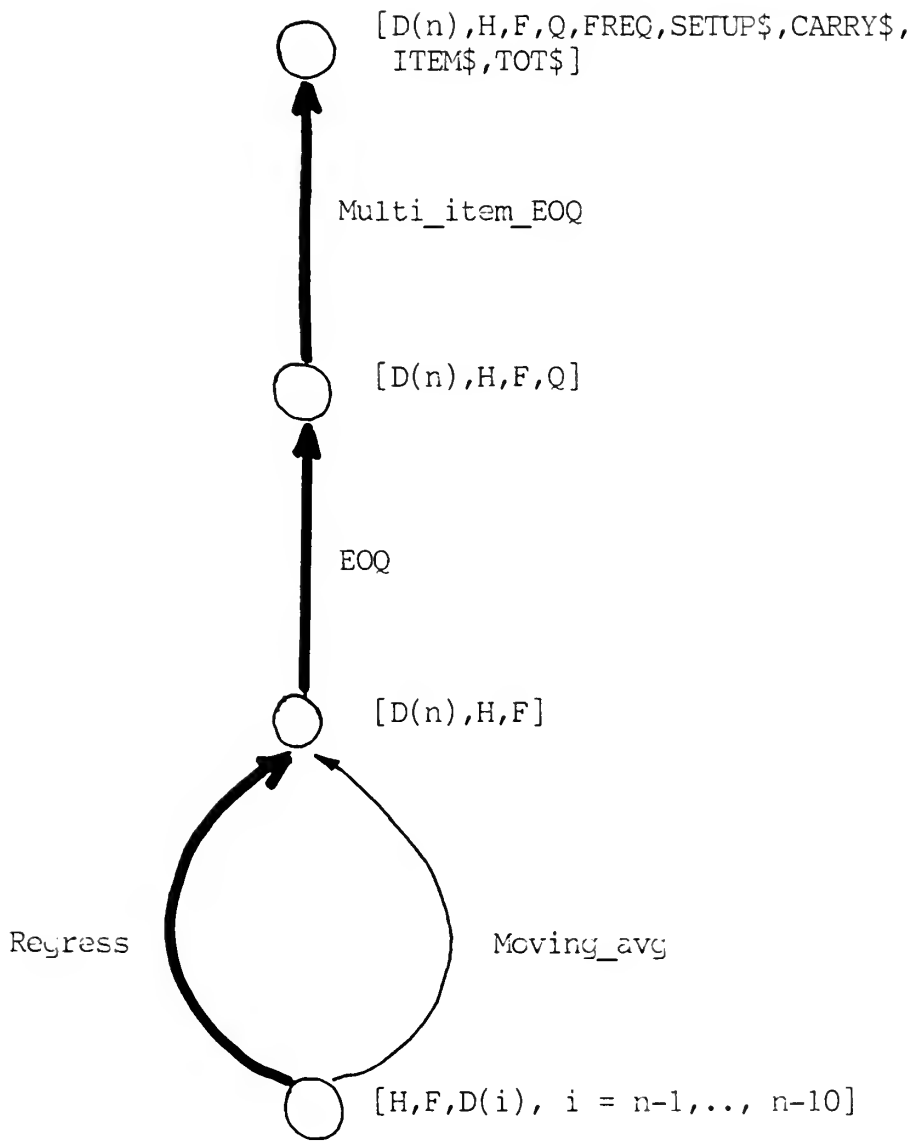

Figure 4. Resulting Model Graph