Decision Support Systems 10 (1993) 137–160
North-Holland

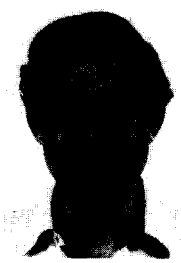
Analogical reasoning and case-based learning in model management systems *

Ting-Peng Liang

National Sun Yat-sen University, Taiwan and Purdue University, West Lafayette IN, USA

Developing computer-based model management systems has been a focus of recent research in decision support. In this paper, we explore the use of analogical reasoning and case-based learning in model management. Analogical reasoning and case-based learning are techniques found useful in human problem solving. They can help model builders apply their modeling experience to construct new models and improve their modeling knowledge through learning. This paper presents a feasible approach to incorporate case-based analogical reasoning in model management systems. First, the role of analogical reasoning and case-based learning in model management is described. A scheme for representing case features is presented. Then, problem and model similarities are discussed at conceptual, structural, and functional levels. This is followed by a description of the process for analogical model formulation, which includes feature mapping, transformation, and evaluation. Finally, examples are illustrated and case-based learning is discussed.

Keywords: Analogical reasoning; Machine learning; Model management systems; Case-based reasoning.



Ting-Peng Liang is an Associate Professor of Information Systems at the Krannert Graduate School of Management of Purdue University. Prior to joining Purdue University, he was on the faculty of the University of Illinois at Urbana-Champaign. He received his MBA from National Sun Yat-sen University (Taiwan, ROC) and Ph.D. in Information Systems from The Wharton School of the University of Pennsylvania. His primary research interests include model management systems, applied artificial intelligence, decision support and expert systems. He has published in a number of academic journals and served on the editorial board for a few journals.

* This research was supported by an XL grant from Purdue Research Foundation and a Research Board Award from the University of Illinois at Urbana-Champaign.

Correspondence to: T.-P. Liang, Krannert Graduate School of Management, Purdue University, West Lafayette, IN 47907, USA. Tel.: (317) 494-4422.

1. Introduction

Due to the increased complexity of today's decision making, quantitative models have played an important role. Developing model management systems (MMS) that facilitate the construction and utilization of these models also becomes a research focus in decision support systems (DSS). Previous work in model management has investigated several important issues concerning model representation, integration, and formulation. For example, Blanning [4–7] studied how relational and entity-relationship models can be used to represent and manipulate models. Dolk and Konsynski [12] developed a frame-based model abstraction technique. Klein [18] and Liang [21,22,24] studied mechanisms for constructing larger models by integrating smaller ones. Geoffrion [14,15] developed a structured modeling language for model specification. Krishnan [20] investigated the application of predicate logic to automated model construction. Liang and Konsynski [28] explored the use of analogical reasoning in model management. Ma et al. [29] and Murphy and Stohr [31] explored the formulation of linear programs using a graph-based approach. Muhanna and Pick [30] adapted the systems concept to model management. Shaw et al. [36] discussed the application of machine learning techniques to the acquisition of model manipulation knowledge.

Among the issues, automatic modeling and system learning are two high-level functions essential to an intelligent MMS [23,27,36]. Automatic modeling allows models to be built automatically from the problem specification provided by the user. The learning capabilities help model builders acquire and maintain modeling knowledge and use previous experience to construct new models.

Different automatic reasoning and learning methods can be applied to model management. In this paper, we study the use of analogical and

case-based reasoning. The concept of case-based reasoning is straightforward. In addition to the abstract knowledge such as rules typically maintained in expert systems, a case-based system also maintains a case base that stores case solved previously. When a new problem is encountered, the system retrieves cases similar to the new problem from its case base, selects the most similar one, and then converts the old solution to a new one for the new problem [34]. This technique has been applied to many domains including criminal sentencing [2], legal reasoning [1], labor negotiation [41], dispute resolution [37], mechanical design [32], and recipe generation [16]. It is appealing because humans often make decisions in similar ways. For example, the recent Gulf Crisis is frequently compared to the cases of Vietnam and Hitler.

In a case-based system, learning can occur in the processes of memorizing new cases, classifying existing cases, and generalizing knowledge from cases. Analogical reasoning is an inference mechanism that allows case similarities to be identified and new solutions to be developed. In human modeling processes, analogical thinking plays a key role in helping modelers construct relationships between variables and reduce the need for repetitive trials. [1] For example, after learning how to construct a linear program for a product mix problem that maximizes profits under given resource constraints, the experience can usually be adapted to develop models for solving similar problems such as a process selection problem that selects production processes to minimize the cost [25]. A recent empirical study also showed that human modelers relied on stereotypes to facilitate the construction of new models. Once the problem type was found, the modeling became easier [38].

Analogical reasoning and case-based learning can be used in many different ways to enhance MMS. In model construction, analogical reasoning capabilities can be a useful aid for model builders to retrieve existing models having similar features. In model execution, case-based reasoning can provide useful information for identifying and selecting proper solvers. In model utilization, case-based reasoning can explain model outputs to decision makers using analogies they can understand. After model utilization, case-based learning allows knowledge gained from analogical reasoning and model utilization to be memorized and used for future problem solving. Furthermore, the case-based approach allows experience to be cumulated and learning to occur on an incremental basis. This is crucial when we try to build the capability of automatic model construction because the cases stored in the case base can help the system improve itself.

In the remainder of this paper, we present an analogical reasoning mechanism for case-based reasoning and learning in MMS. First, analogical reasoning and its roles in model management are further discussed. Then, problems, models and their representations are presented. Problem and model similarities are discussed at conceptual, structural, and functional levels. Third, the process of analogical modeling, case-based learning, and the operations for transformation are described. Two examples are used to illustrate the analogical reasoning process. One shows the construction of a media selection model based on its similarity to an existing product mix model, whereas the other shows using partial similarity to modify an existing inventory control model to meet new policies. Finally, case-based learning and future research issues are discussed.

2. The role of analogical reasoning and learning

Analogical reasoning is a basic problem solving technique. An analogy is often defined as "a problem of the form A is to B as C is to D ($A:B::C:D$), where, in most situations, the last term is omitted and must be filled in, selected from among answer options, or confirmed in a true–false situation." [40, p. 237]. Analogical reasoning is a process by which D is determined by some known properties among A, B, and C. For example, a common analogy politicians use to justify the military action in the Gulf Crisis is [Hitler's invasion: World War II::Hussen's inva-

[1] The importance of analogical thinking has been discussed by many researchers. A detailed discussion on its role in science can be found in [17]. Analogical reasoning, however, is not the only approach used in case-based reasoning. Other techniques include explanation-based and inductive classification methods. For more details, see [10,16,19]. The importance of analogical reasoning in MMS can be found in [28].

sion:?]. A major reason for using analogical reasoning is that it reduces the cognitive load in sophisticated problem solving. In artificial intelligence, analogical reasoning has also been found useful in machine learning [8], computer program construction and debugging [11], and other areas.

An analogical reasoning process usually includes three core transformation operations: Inference, mapping, and application. [2] First, the relationship between A and B is derived at the inference stage. In our previous example, this relationship may be "Hitler's invasion caused World War II". Then, the relationship between C and D is determined by mapping A to C. In the previous analogy, the relationship between Adolf Hitler and World War II is mapped to the Gulf Crisis. Finally, the derived relationship is applied to find D. Based on this analogy, the outcome of Hussein's invasion would have been some sort of large-scale war similar to World War II if it was not stopped.

Analogical reasoning is an inference process that drives case-based systems. When a new problem (C) is encountered, the system retrieves from memory analogous problems (A) and then modifies the old solutions (B) to develop new ones (D). Two features make the case-based approach suitable for model management. First, a model base usually consists of a number of existing models, which provide a convenient basis for analogical construction of new models. Second, the relationship between existing models and the problems they are capable of solving is usually known.

Analogical reasoning can be integrated into many MMS functions [28]. In this paper, we focus our discussions on model formulation. When a new problem is to be modeled, the MMS must first identify old problems with known models that are similar to the new one. This involves access to the case base and using reasoning knowledge to determine problem similarity. Here a case in the case base is a combination of a problem and its quantitative model. Once similar problems are found, knowledge must be applied to evaluate the appropriateness of analogies and choose the most appropriate one if alternatives exist. After the evaluation, the new model can be constructed by finding matching components between the problems and converting the old model to fit the new problem.

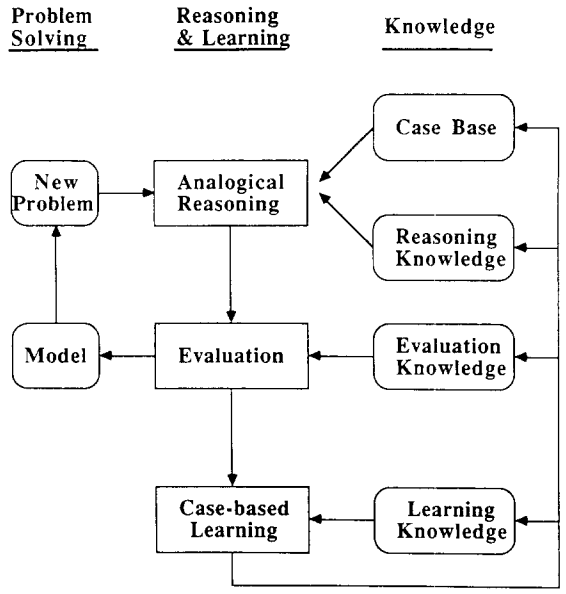


Fig. 1. Role of CBR in model management.

Case-based learning can occur throughout the process of model formulation in two possible forms. First, the new case can be stored to enrich the case base. Second, the experience gained from the model formulation process allows the evaluation and refinement of the knowledge used in reasoning, evaluation, and learning. Fig. 1 illustrates the case-based analogical reasoning and learning processes.

Several issues are keys to using the case-based approach in model management. First, cases must be properly represented and stored for re-use. Second, problem and model similarities must be defined. Finally, operations must be identified to construct new models from existing ones and to learn knowledge from experience.

3. Case representation

Given our definition that a case includes a problem and its model, we need to define problems, models, and their representations in the case base.

[2] Disagreements exist among a large number of theories of analogical reasoning. See [39,40] for a detailed discussion.

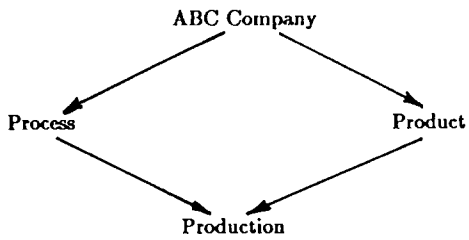
3.1. Problems and feature representation

The basic elements constituting a real-world problem include *features (nominal and numerical) and their relationships.* Nominal features (called "entities") do not have numerical values but can be grouped into sets. Numerical features (called "attributes") have values to indicate properties of certain entities in a problem and can be represented by a set of subattributes. The representation of a problem, therefore, should include at least three aspects: Entities, attributes, and subattributes.

Definition 1. An *entity* is an object or concept of interest to the decision maker. It does not have a numerical value but can be measured by its properties.

Definition 2. An *attribute* is a measurable property of an entity of interest to the decision maker. It can be represented by a set of subattributes.

(a) Entity Graph



(b) Attribute Hierarchy

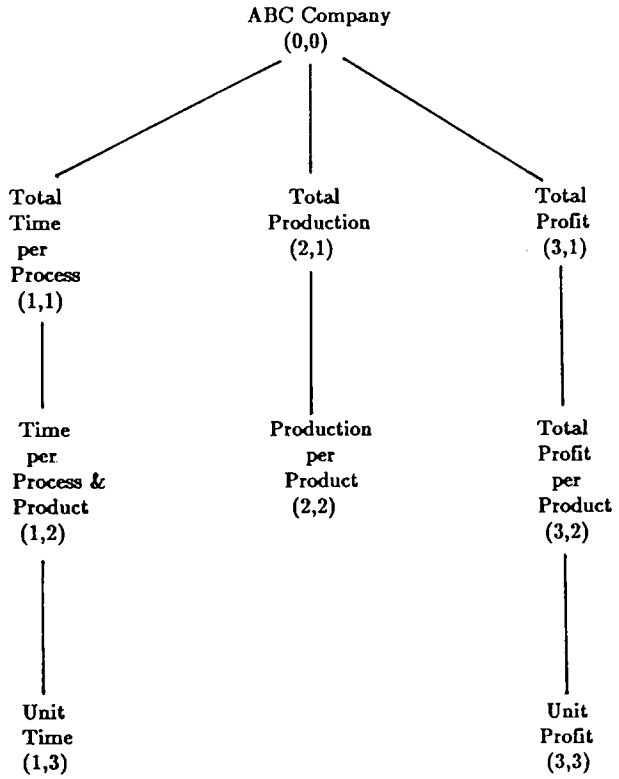


Fig. 2. Conceptual structures of the product mix model.

Definition 3. A *subattribute* is a concept associated with an attribute. It may be related to the value or the measurement of the attribute.

Similar entities can be grouped into *entity sets* and an *entity graph.* Similar attributes can be grouped into *attribute classes* and an *attribute hierarchy*. Each problem contains an entity graph and an attribute hierarchy.

Definition 4. An *entity set* contains a finite number of entities having similar attributes.

Definition 5. An *entity graph* is a directed graph showing the association of entity sets in a problem. Problem identification is the root of the graph.

Definition 6. An *attribute class* contains a finite number of attributes having similar subattributes.

Definition 7. An *attribute hierarchy* is a tree of attribute classes to indicate attribute affiliations. Problem identification is the root of the tree.

Definition 8. A *problem* is represented as a combination of an entity graph and an attribute hierarchy whose elements are properly instantiated by entity and attribute instances of interest to the decision maker.

We use the product mix problem that maximizes the total profit under certain production constraints as an example (see appendix A). The entities of the problem include dot matrix and laser printers (which can be grouped into a set "product"), inspection and assembly lines (which can be grouped into a set "process"), and the linkage between products and processes (which can be grouped into a set "production"). These entities and entity sets can then be organized into an entity graph as shown in fig. 2(a).

The problem includes many attributes that can be grouped into the following attribute classes: Unit profit, total profit by product, total profit, unit inspection and assembly times, total inspection and assembly times, production quantities by product, and overall production quantity. Each of the attributes can be represented by at least three subattributes: Domain, value status, and unit. The domain indicates the range of legal values. The value status shows the current value of the attribute. The unit shows the measurement unit of the attribute. In addition, we would like to include its affiliated entity and its position in the attribute hierarchy. These require the addition of two subattributes: Instance_ of and entity. For example, the unit profit and the production quantity of dot matrix printers can be represented as follows:

unit_profit(dot_matrix)

instance_of:	unit_profit
unit:	dollar
domain:	Real
value status:	100

quantity(dot_matrix)

instance_of:	production_per_product
unit:	unit
domain:	integer
value status:	unknown

The attribute classes can be organized in an attribute hierarchy as shown in fig. 2(b). The tree starts with a root (which is the problem identification) and grows by adding more and more attribute classes to its branches. There are two issues associated with the tree construction: (1) how to differentiate branches; and (2) how to grow leaves. In this research, we use the primary unit of an attribute to build branches. The primary unit of an attribute with a single unit is that unit. For an attribute whose unit is a combination of more than one subunit, the numerator is its primary unit. For example, the primary unit of "pound" is "pound" and that of "dollar/month" is "dollar".

The leaves grow from attribute classes representing more aggregrated concepts to those representing more detailed ones. In other words, attribute classes are organized with a decreasing level of aggregation. For example, the total profit of dot matrix printers is considered more aggregated than the unit profit of a dot matrix printer. Total inspection time is more aggregrated than the inspection time spent on a product.

3.2. Model and model representation

A quantitative model is a mathematical representation of reality that can help a decision maker answer questions. In order for a decision maker to answer questions of interest, a model must

incorporate all possible states of the problem. In other words, a *model* is a set of mathematical statements from which all states in the state space of a problem can be derived. The derivability and models can be defined as follows.

Definition 9. A state ϵ in the state space $\mathscr{E}$ is *derivable* from a set of mathematical statements M, if the functions and relations of M are true for the features and values in ϵ.

Definition 10. M is a *model* of $\mathscr{E}$, if for all $\epsilon \in \mathscr{E}$, ϵ is derivable from M.

One necessary condition for a model is that its statements must cover the whole set of $\mathscr{E}$. Our definition reveals two points. First, models and decision goals are separated. The major advantage of this division is that, given the same model, different goals may be achieved in different circumstances. Second, models and data are separated. This allows sensitivity analysis, a critical step in decision making, to be defined as an operation that applied the same model to different attribute values. Both concepts are consistent with previous work including Geoffrion's structured modeling [14] and Zeigler's work on simulation modeling [42].

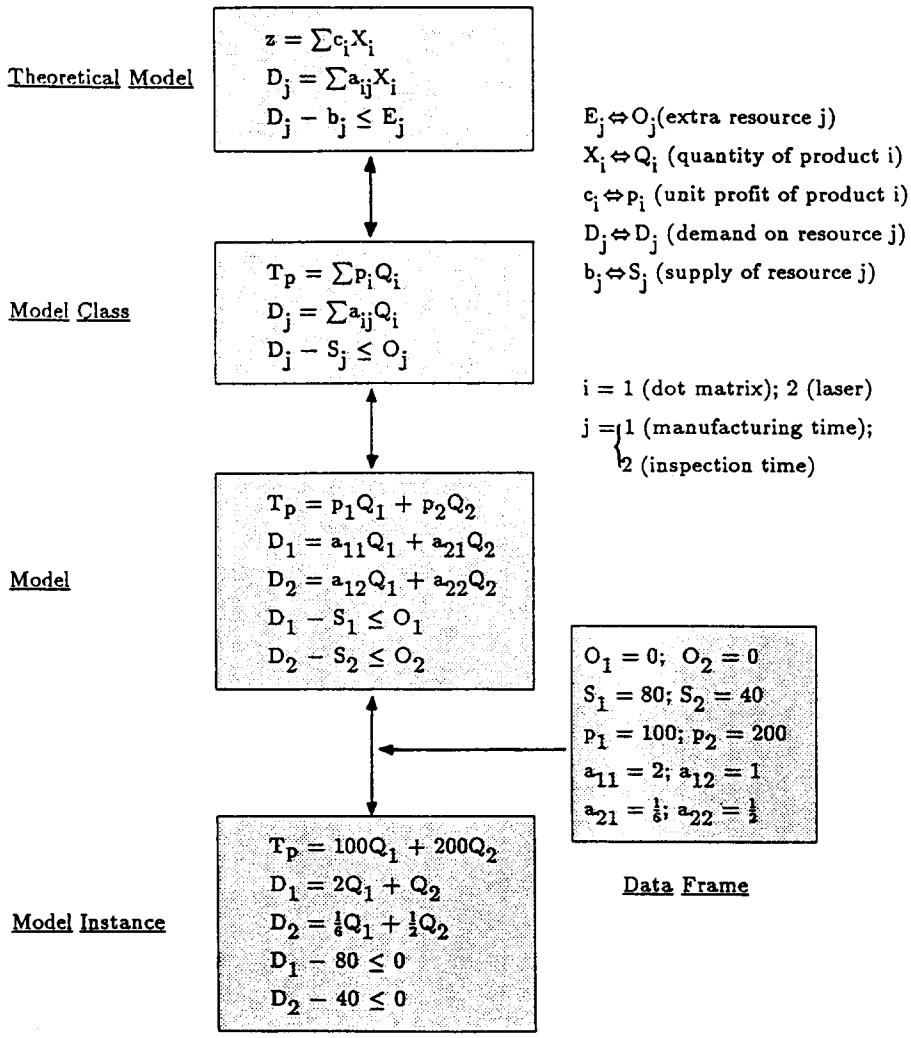
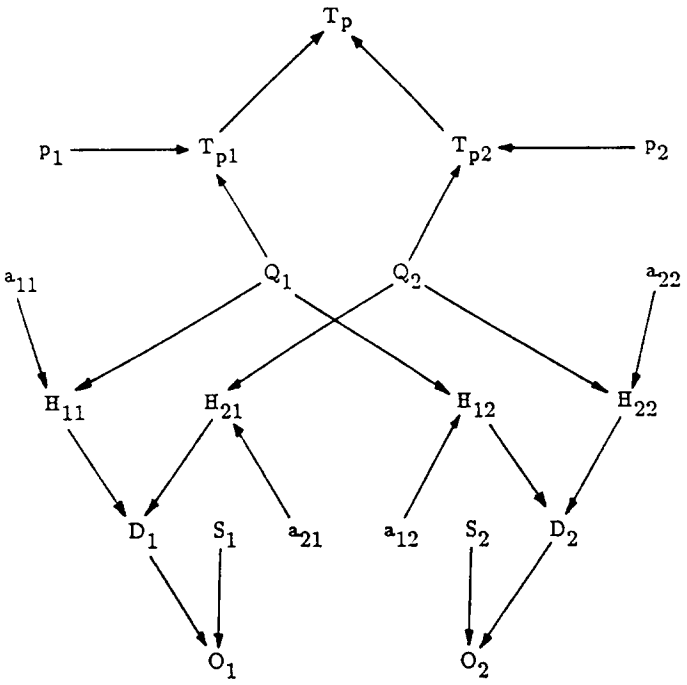


Fig. 3. Four levels of model generalization: An example.



Notation: T_p = Total profit; T_{p1} = Total profit of dot-matrix (DM) printers; T_{p2} = Total profit of laser (LS) printers; p_1 = Unit profit of DM; p_2 = Unit profit of LS; Q_1 = Production of DM; Q_2 = Production of LS; H_{11} = Total assembly time for DM; H_{12} = Total inspection time for DM; H_{21} = Total assembly time for LS; H_{22} = Total inspection time for LS; a_{11} = Unit assembly time for DM; a_{12} = Unit inspection time for DM; a_{21} = Unit assembly time for LS; a_{22} = Unit inspection time for LS; D_1 = Total demand on assembly time; D_2 = Total demand on inspection time; S_1 = Total available assembly time; S_2 = Total available inspection time; O_1 = Assembly overtime; O_2 = Inspection overtime.

Fig. 4. Elemental graph of the product-mix model.

Following the above definitions, the product mix model is presented as follows: [3]

$T_p = T_{p_1} + T_{p_2}$ {Total profit},
$T_{p_1} = p_1 Q_1$ {Profit from dot-matrix printers},
$T_{p_2} = p_2 Q_2$ {Profit from laser printers},
$H_{11} = a_{11} Q_1$ {Assembly time for dot matrix},
$H_{21} = a_{21} Q_2$ {Assembly time for laser},
$H_{12} = a_{12} Q_1$ {Inspection time for dot matrix},
$H_{22} = a_{22} Q_2$ {Inspection time for laser},
$D_1 = H_{11} + H_{21}$ {Demand on Assembly time},
$D_2 = H_{12} + H_{22}$ {Demand on inspecting time},
$O_1 \geqslant D_1 - S_1$ {Assembly overtime},
$O_2 \geqslant D_2 - S_2$ {Inspection overtime}.

A model may have several different levels of generalization. First, specific constants can be replaced by parameters. Second, parameters can be generalized to remove isomorphic structures. Third, certain semantic meanings can be removed to make a model purely mathematical. The first operation generalizes from a model instance (i.e., combining a model and its parameter values in a data frame) to the model. The second operation generalizes from a model to a model class. By a *model class*, we mean all models with similar concepts and structures, e.g., all product-mix

[3] The model shown here is an elaborated version of a linear program with two variables and two constraints. It can be easily simplified to the standard format. Please see fig. 4 for notations.

models. The third operation generalizes from a model class to a *theoretical model* (also called a *basic model*) in which no semantic meanings are associated with the variables or parameters. The model becomes a set of purely mathematical formulas to be manipulated by mathematical operations. The theoretical model is useful in exploring the fundamental properties of a model class. Fig. 3 shows an example of these operations which generalizes an instance of the product-mix problem to its theoretical form. Different symbols are used in the figure to indicate the difference between a theoretical model and a model class.

In order to show the functional relationships among attributes in a model, we use directed graphs. Each model is represented as two directed graphs: An elemental model graph and a generic model graph. The difference between them is that the former is built on attributes and the latter is built on attribute classes. *Unlike the entity graph and attribute hierarchy which show semantic relationships of features perceived by the user, model graphs show actual functional relationships in mathematical models.* Practically, modeling becomes a two-step process through which the modeler: (1) converts the perceived entity graphs and attribute hierarchies into model graphs; and (2) translates model graphs into mathematical formulas by inserting proper functional operators.

Definition 11. An *elemental model graph* is a directed graph of nodes and arc. Each node represents an attribute, and each arc represents a functional mapping from one node to another (i.e., the value of the target node is assumed or known to be determined by the origin).

For example, the model (total cost = unit cost × quantity) can be represented as three nodes with two arcs connecting the unit cost node and the quantity node to the total cost node. Fig. 4 illustrates the elemental model graph of the product mix model. [4]

One problem associated with the elemental graph is that when the number of concepts increases, the complexity of the graph may increase exponentially. One way to reduce the complexity

[4] Symbols are used for simplicity.

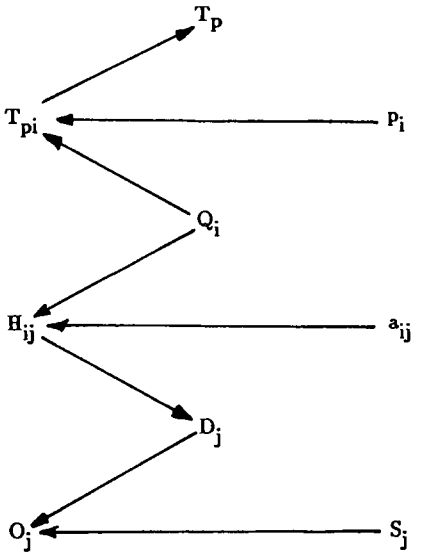


Fig. 5. Generic graph of the product mix model.

is to combine isomorphic substructures and use concept classes when more than one concept in the problem is found to be in the same category. The structure organized on concept classes is called the generic model graph.

Definition 12. A *generic model graph* is a directed graph of nodes and arcs. Each node represents an attribute class. Each arc represents a functional mapping from one node to another.

For example, the left half of the model structure shown in fig. 4 looks like a mirror image of its right half, which means that they are similar and can be combined without losing the generality of the structure. Fig. 5 shows the generic structure for the product mix model. This simplification process can greatly increase the efficiency of feature matching in the analogical process. A detailed discussion of structural similarity is provided later in section 4.2.

Except for two differences, the elemental and generic graphs are similar to Geoffrion's element and genus graphs, respectively. First, a genus graph includes a special node called "test" to handle binary relational functions such as "$X = Y$",1 which is not used in our representation. Second, an element or genus graph in structured modeling mixes index variables and decision variables, and shows index variables at the bottom level. Our elemental and generic graphs include

decision variables only. Discussion on analogical reasoning in structured modeling can be found in [26].

3.3. Case memory

When a problem is encountered, a typical model formulation process includes identifying key features, finding relationships among the concepts, and, finally, constructing mathematical functions to represent their exact relationships. This process indicates that there are three essential elements for analogical modeling that need to be maintained in the case base: Features, logical structures, and mathematical equations. Therefore, a case can be stored as a triple, $\langle E, S, F \rangle$, where E is the problem representation discussed in section 3.1 and S is the generic model graph described in section 3.2, and F is the mathematical form of the model.

Definition 13. A case is represented as a triple $\langle E, S, F \rangle$, where E is a *set of features* reflecting a real world situation; S is a *model structure* of E showing relations between concepts in E; F is a *set of mathematical functions* corresponding to S.

4. Case similarity

Similarity is the basis on which analogical reasoning can be performed. In this section, we discuss case similarity based on our representations of problems and models. Since each case consists of features, model structures, and mathematical functions, similarities can be defined at conceptual, structural, and functional levels. The case-based analogical modeling enables functional similarity to be determined from conceptual and structural similarities. We use the media selection problem and the product mix problem in appendix A as examples to illustrate the similarities we define.

4.1. Conceptual similarity

The conceptual similarity shows the similarity between the features in two problems without considering their functional dependencies in elemental and generic graphs. This similarity can be determined by the semantic description of problems.

Features in two cases may be similar in at least three ways. They may share the same features, have similar features, or have similar entity graphs and attribute hierarchies. When two problems have shared features, their degree of similarity depends on the number of features they share. The strongest similarity exists when two sets of features are exactly the same or that one set is a subset of another. In this case, it is possible that the two problems are the same or one is a subproblem of another. The similarity is the weakest when they share only one common features.

Even if two cases share few common features, individual entities may have similar attributes, which can be measured by their subattributes such as domains, value status, and units. Two attributes defined on the same domain may have similar mathematical behaviors. For example, total cost and total production hold a domain similarity. It should be noted that domain similarity is a necessary but not sufficient condition for attribute similarity. Also, some domains such as integer and real numbers may be compatible in some problems.

In addition to domain similarity, two features may be similar because they have the same value status. The value status of an attribute may fall into one of the following categories: The exact value is known (e.g., \$100), the exact distribution is known (e.g., N(10,5)), the type of distribution is known (e.g., exponential distribution), and completely unknown. Different value status often requires different modeling effort.

The third subattribute that affects attribute similarity is its measurement unit. If two sets of attributes have the same set of units, they may share the same theoretical model. For example, a model that calculates total profit form the profits of three individual products has the same unit as a model that calculates the total cost of two products. In both cases, all elements have the unit "dollar" and share the same theoretical model $T = \Sigma_i X_i$ (where $i = 3$ for the former and $i = 2$ for the latter).

In fact, having identical units is only a special case of unit similarity. What actually governs unit similarity is the *unit pattern* of a model. For a single attribute, its unit pattern is the format of the unit. For example, "dollar" and "pound"

belong to the same unit pattern, but "dollar" and "dollar/pound" are of different patterns. For a set of attributes, the unit pattern shows the relation among their units. An important function of unit patterns is that it allows different units such as dollar and pound to be comparable. For example, $T = \Sigma_i X_i$ is applicable to any problems having the same unit pattern, such as calculating the total production from the productions of individual items and the total nutrition from the nutrition contents of individual fruits.

One way to determine the unit pattern for a set of attributes is to replace their units with variables (e.g., u_i). If the variables can be successfully instantiated with corresponding units in another attribute set, then these two sets have the same unit pattern. For example, the pattern of [dollar, pound, and dollar/pound] can be represented as $[u_1, u_2, u_1/u_2]$ and then be used to match units in other sets. Based on the above discussion, we can define feature similarities as follows.

Definition 14. Two attributes have a *domain similarity* if they are defined on the same value domain.

Definition 15. Two attributes have a *value similarity* if they have the same value status.

Definition 16. Two attributes have a *unit similarity* if they have compatible unit patterns.

Definition 17. Two *attributes are similar* if they have domain, value, and unit similarities.

Definition 18. Two *entities are similar* if they consist of similar attributes.

The attribute and entity similarities are applicable to individual entities and attributes as well as entity sets and attribute classes. They allow different features to be comparable for similarity. For example, "Process" is an entity set (including inspection and assembly) in the product mix problem that may be compared with "Medium" or "Resource" in the media selection problem (see fig. 6 for its entity graph and attribute hierarchy). Each process has three attributes of known value (total time, time per process and product, unit time). If they match with the attributes of "Resource" (i.e., overall budget, budget per medium, unit budget), the mapping preserve domain, unit, and value similarities. If we match "Process" with "Medium", however, their attribute mappings have domain and unit similarities, but not value similarity. All attribute values are exactly known for "Process", but the number of medium runs is unknown for "Medium." Therefore, the former is a better match. We can say "Process" and "Resource" are similar entities.

4.2. Structural similarity

In addition to measuring the similarity of individual features, similarities may exist among structures such as entity graphs, attribute hierarchies, elemental graphs, and generic graphs. Since elemental graphs are elaborated versions of generic graphs, structural similarity can be defined on entity graphs, attribute hierarchies, and generic graphs. Two major types of similarities exist between graph structures: Homomorphism and isomorphism.

Definition 19. Two structures, S and S', are *homomorphic,* if there exists a function $f: S \to S'$ so that for all $s \in e(S)$, $f(s) \in e(S')$, and for all $a \in r(S)$, $f(a) \in r(S')$, where $e(S)$ means all nodes in S and $r(S)$ means all arcs in S.

In other words, two structures are homomorphic if there is a function that maps all nodes and arcs from one structure to another. Homomorphism between two graphs indicates that one may be a higher-level abstraction of the other. A special case of homomorphism is isomorphism in which the mapping function is one-to-one; that is, for each node or arc in one graph, there exists exactly one corresponding node or arc in the other.

Definition 20. Two structures, S and S', and *isomorphic*, if and only if they are homomorphic and the function f is one-to-one.

Two isomorphic structures must have the same number of elements and their elements must be ordered in an equivalent sequence. The corresponding elements can have different semantic meanings and the functions connecting the elements may be different. This allows the structural

similarity of attribute hierarchies, entity graphs, and generic model graphs to be defined.

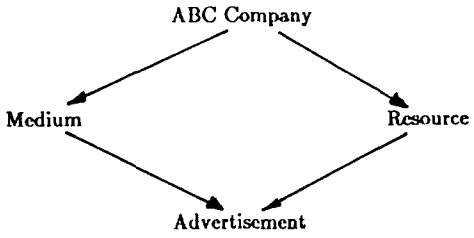
Definition 21. Two *attribute hierarchies are said to be similar* if they are isomorphic and, for each attribute class in one hierarchy, there exists a similar attribute class on the corresponding position in the other.

Definition 22. Two *entity graphs are similar* if they are isomorphic and, for each entity in one graph, there exists a similar entity on the corresponding position in the other.

Definition 23. Two *generic model graphs are similar* if they are isomorphic and, for each node in one graph, there exists a similar one on the corresponding position in the other.

For implementation, one way to identify the similarity of attribute hierarchies is to label the position of each attribute class in two aspects: Its

(a) Entity Graph



(b) Attribute Hierarchy

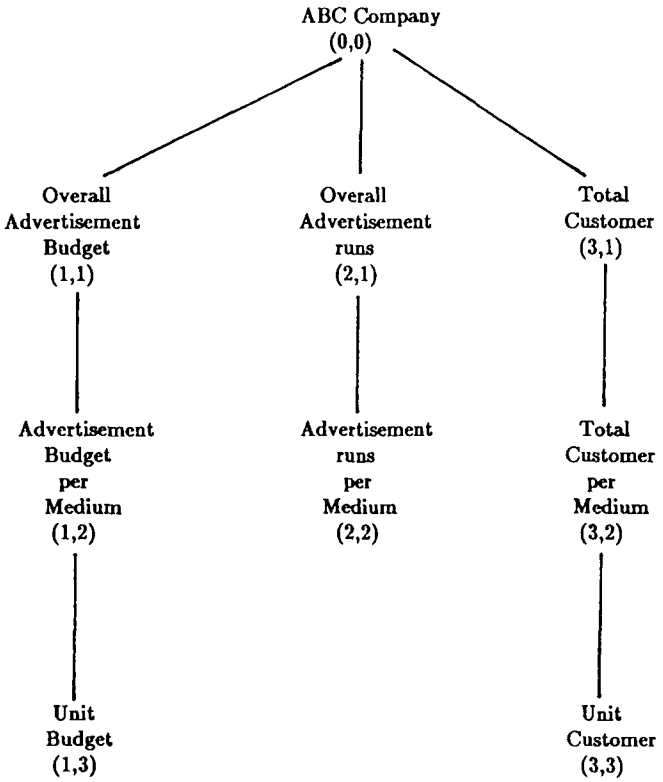


Fig. 6. Conceptual structures of the media selection problem.

branch and hierarchical level. The first label differentiates attributes by their primary units. Each branch name is given a unique identification, such as '1' for the time branch, '2' for the production branch, and '3' for the profit in fig. 2(b). The second label indicates the hierarchical level of an attribute in the branch. The first node in a branch (e.g., total time, total product, and total profit) is considered the first level and is given a label '1'. The levels below the class name level are labeled '2', '3', etc. The root is labeled (0,0). For representation simplicity, we use e_{ij} to denote the hierarchical position of an attribute class, where i and j are its labels. For instance, an attribute that is an instance of the attribute class "production per product" in fig. 2(b) is denoted as e_{22}. After all positions are labeled, two attribute hierarchies can be compared for similarity. Similarly, entity graphs may be similar structurally.

In some situations, similar attributes or entities may be labeled differently in different attribute hierarchies or entity graphs. This happens when a problem is a subproblem of another, or when different model builders represent features at different levels of scrutiny. In order to avoid mismatches due to different labeling conventions, a relabeling function that allows labels to be reorganized can be defined as follows.

Definition 24. A *relabeling function,* f_r, is a one-to-one mapping between two sets of feature labels $E_1 = \{e_{ij}\}$ and $E_1' = \{e_{i'j'}\}$ such that:

(1) for all i in E_1, there exists an i' in E_1'; and
(2) for j_1 and j_2 in E_1 and $j_1 \geqslant j_2$, there exist j_1' and j_2' in E_1' and $j_1' \geqslant j_2'$.

For example, a set of attributes $[e_{12}, e_{14}, e_{22}, e_{24}]$ may be relabeled to become $[e_{11}, e_{12}, e_{21}, e_{22}]$. The relabeling function in this example is $i' = i$ and $j' = j/2$. In general, the relabeling operation allows more flexibility in feature matching. It should be noted that e_{ij} are variables used to represent the hierarchical positions of attributes. Relabeling them should not affect the nature of attribute similarity. Therefore, if there exists a relabeling function that can relabel one attribute hierarchy to make it the same as another, then these two must be similar. Similar operations are applicable to entity graphs.

Theorem 1. (*a*) *Two attribute hierarchies are similar if there exists at least one relabeling function that maps attributes in one hierarchy to similar ones in the other.*
(*b*) *Two entity graphs are similar if there exist at least one relabeling function that maps entites in one graph to similar ones in the other.*

Theorem 1 gives us more flexibility to identify attribute hierarchy and entity graph similarities. Its proof is staightforward and omitted here.

Because two problems having similar attribute hierarchies and entity graphs are very likely to have the same model as represented in generic model graphs, we can use attribute hierarchies and entity graphs to derive new models. For example, fig. 6 shows the entity graph and attribute hierarchy of the media selection problem described in appendix A. By comparing figs. 2 and 6, it is clear that the media selection of product mix problems have similar entity graphs and attribute hierarchies. The best mappings between them are as follows:

Entitie set		
Root	↔	Root
Process	↔	Resource
Product	↔	Medium
Production	↔	Advertisement
Attribute class		
Total profit	↔	Total # of customer
Total profit per product	↔	Total # of customer per medium
Unit profit	↔	Unit # of customer
Total production	↔	Total ad. runs
Prod/product	↔	Runs per medium
Total time	↔	Total budget
Time/process	↔	Budget per medium
Unit time	↔	Unit budget

Sometimes, a model is a subset of another. In this case, the structure of the smaller model is also a subset of the larger one. For example, the inventory model that does not allow shortage is a submodel of the inventory model that allows shortage. In this case, similarity at the substructure level exists.

Definition 25. Structure S is a *substructure* of structure L, if $e(S) \subseteq e(L)$ and $r(S) \subseteq (r(L) \cap e(S))$. It is denoted as $S \subseteq L$.

In other words, the nodes of S are a subset of or equal to the nodes of L and the arcs of S are a subset of or equal to the arcs of L. We say that a structure S is *homomorphically embedded* in L if there is a homomorphic function f from S to a substructure of L. If the function f is isomorphic, then we say that S is *isomorphically embedded* in L [9]. A further generalization of the concept allows substructure homomorphism and isomorphism to be defined.

Definition 26. Two structures L and L' have *homomorphic substructures* if there exists at least one substructure S, $S \subseteq L$ and S is homomorphically embedded in L'.

Definition 27. Two structures L and L' have isomorphic substructures if there exists at least one substructure S, $S \subseteq L$ and S is isomorphically embedded in L'.

Similarities at the substructure level allow a new model to be constructed by integrating a few smaller model pieces or using a piece of a large model. Similar concepts have been used in model management literature and a few recent case-based reasoning papers (e.g. [32,33]). In summary, structural similarity can be derived from conceptual similarities. In particular, entity and attribute similarities and isomorphic structures lead to entity graph and attribute hierarchy similarities. These, in turn, lead to the model graph similarity, which allows the generic graph of the new model to be constructed analogically.

4.3. Functional similarity

Functional similarity portrays common features underlying the mathematical equations in different models. There are two types of operators in a typical mathematical equation: Relational and functional operators. A *relational operator* indicates the relationship between two or more elements. A *functional operator* (called a *functor*) converts the value of one or more elements into the value of another. For example, the function $T_c = T_h + T_0$ includes a relational operator '=' and a functor, Σ, that links T_h and T_0. Two functions are said to be similar if they are composed of the same relational operators and functors.

Each relational operator or functor connects a certain number of concepts. These concepts are called its *arguments.* The number of arguments is called its *arity*. For example, '=' links two arguments and, hence, has an arity of two. Operators with arity of one are called unary operators; operators with arity of two are called binary operators; and operators with arity of n are called n-ary operators. The arity of an operator determines its properties and can be used to represent its type.

Definition 28. The *type of a function* is an ordered set that includes relational operators and functors and their associated arities as elements.

For example, the type of $T_c = C_1 + C_2$ is $[=/2; \Sigma/2]$ and the type of $T_c = a_1X_1 + a_2X_2$ is $[=/2; \Pi/2, \Sigma/2]$. They indicate that the first function is composed of a binary relational operator $(=)$ and a binary functor (Σ), and the second one is composed of a binary relation $(=)$ and two binary functors (Π and Σ).

Definition 29. Two *functions* f_1 and f_2 are *similar* if they have the same type.

In addition to functions with the same functional type, partial match of types may exist. For example, $Y = \Sigma_i X_i$ and $Y \geqslant \Sigma_i X_i$ may also be considered similar, though they have different relational operators ($=$ and $\geqslant$). Therefore, a partial match can be called a *weak functional similarity* and a complete match can be called a *strong functional similarity.*

Definition 30. Two *functions* f_1 and f_2 are *weakly similar* if their corresponding relational operators and functors have the same arities.

By this definition, functional types $[\geqslant/2; \Sigma/2]$ and $[=/2; -/2]$ are weakly similar. Since a model may include more than one function, the functional type of a model is a combination of the types of its functions. Two models are considered similar at the functional level if their functional types are similar. For example, the product mix model (a simplified form of the equations shown

in section 3.2) and their corresponding types are as follows:

Model	*Type*
$T_p = p_1Q_1 + p_2Q_2$	$[=/2;\ \Pi/2,\ \Sigma/2]$,
$S_1 \geqslant a_{11}Q_1 + a_{12}Q_2$	$[\geqslant/2;\ \Pi/2,\ \Sigma/2]$,
$A_2 \geqslant a_{21}Q_1 + a_{22}Q_2$	$[\geqslant/2;\ \Pi/2,\ \Sigma/2]$,

4.4. Model similarity

The above discussion on various kinds of similarities provides a foundation for analogical modeling. Overall, model similarity can be determined by conceptual, structural, and functional similarities between two models. Conceptual similarity is measured by subattributes including domain, unit, and value status. Structural similarity is measured by morphisms, i.e., the elements and mapping between elements in different structures. Functional similarity is measured by functional types.

Definition 31. Two *models* M_1 and M_2 are *similar* if they have similar concepts, isomorphic structures, and similar functional types.

In fact, if two cases are similar in features and have isomorphic generic graphs, the only way they can have different models is to assign different functions to corresponding arcs in their generic graphs. The attribute and entity similarities, however, prevent this from happening (remember that the unit similarity requires matching attributes to have similar unit patterns). Case-based reasoning allows model building based on functional similarity inferred from conceptual and structural similarities. The following theorem proves that this is true in a mathematical system involving real numbers, relational operators of '$=$', '$<$', '$>$', '$\leqslant$', and '$\geqslant$', and functors of addition, subtraction, multiplication, division, exponential, square, and cubic (denoted as $\{R; [=, <, >, \leqslant, \geqslant]; [+, -, *, /, \exp, \text{sqr}, \text{cub}]\}$). The functors listed here are only representatives; more may be added.

Theorem 2. In $\{R; [=, <, >, \leqslant, \geqslant]; [+, -, *, /, exp, sqr, cub]\}$, *two models that hold conceptual and structural similatities also hold at least a weak functional similarity.*

Proof. See appendix B.

Since attribute similarity are measured by their subattributes including domain, unit, value status, and their positions in attribute hierarchies, we can further generalize attributes to attribute types that combine their domains, value status, positions, and units. This allows attributes to be compared at a higher level of abstraction.

Definition 32. The *type of an attribute* is a combination of its domain, value status, hierarchical position, and unit.

For example, the types of the total and unit profits in fig. 2(b) are $[R, \text{U}, e_{31}, u_1]$ and $[R, \text{K}, e_{33}, u_1/u_2]$, respectively. The value status may be known (K) or unknown (U) in deterministic models. Data distribution such as a normal distribution with known mean and variance (N(K, K)) may exist in stochastic models. Similarly, the generic graph of a model can be generalized by replacing nodes with their corresponding attribute types to eliminate the effect of semantic meanings and make models in different domains comparable. The resulting structure is called the structural type of the model.

Definition 33. The *structural type* of a model is the generic graph of the model (as defined in section 3.3) whose attributes in the graph are replaced with their corresponding types.

Fig. 7 shows the structural type of the product mix model. Since attribute types determine attribute similarity, two models with isomorphic structural types and similar attribute types in matching attributes are functionally similar (theorem 2). Therefore, the structural type of a model is like *the gene of the model,* from which the whole model can be formulated through proper self-duplication. Two structural types are *isomorphic* if the structures are isomorphic and the corresponding attributes types are the same. Two structural types are *homomorphic* if the structures are homomorphic and the concept types are the same. Theorem 2 can be further elaborated to facilitate model construction.

Lemma 1. Strong model similarity Two models M_1 *and* M_2 *are strongly similar if they have a*

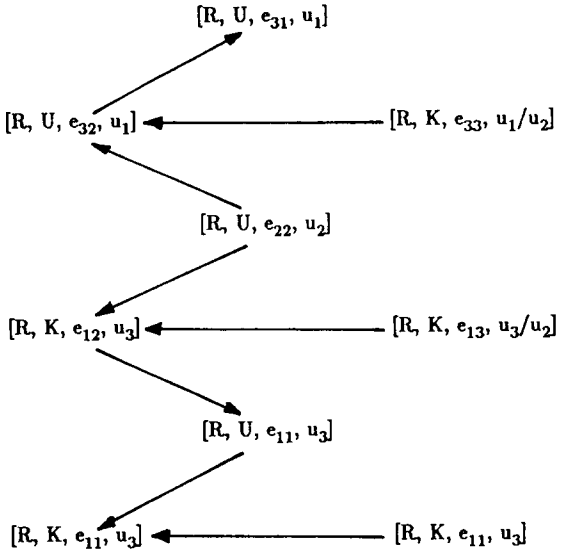


Fig. 7. Structural type of the product mix model.

conceptual similarity, isomorphic structural types, and the same corresponding relational operators.

Lemma 2. Weak model similarity *Two models M_1 and M_2 are weakly similar if they have a conceptual similarity and isomorphic structural types.*

Lemma 3. Partial model similarity *Two models M_1 and M_2 are partially similar if they have a conceptual similarity and isomorphic substructure types. A partial similarity is strong if the corresponding relational operators are the same. It is weak, otherwise.*

5. Process of case-based reasoning and learning

Analogical model construction using model similarity includes several steps. First, the new problem is analyzed to identify its entities, attributes, entity graph, attribute hierarchy, and attribute types. Second, a proper analogue is located using similarities found between the problem to be solved and existing cases. Mechanisms for feature matching play a central role in this stage. Third, transformation operations are performed to develop the new model from the analogue. Finally, the new model is evaluated by the model builder, modified if necessary, and then used for problem solving. We shall use the media selection problem as an example to illustrate how its model can be constructed analogically.

5.1. Problem analysis

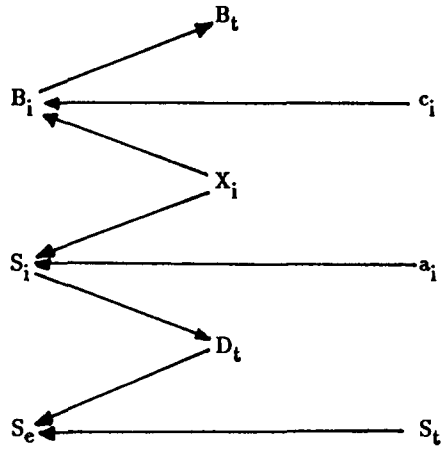
In problem analysis, features including their entities, attributes and subattributes are indentified. Then, an entity graph and an attribute hierarchy are constructed to organize those features. Finally, attribute types are defined for the problem. The resulting attribute types are keys for finding analogous models. In the process, an effort must be made to identify as many explicit and implicit features and relationships as possible. In general, the more completely the initial features and relationships are identified, the easier it is to find a good analogy. Sometimes, functions may also be identified. These functions usually result from definitions of attributes or other explicit statements of relationships. They are useful in evaluating the hypothetical model generated analogically.

In the media selection problem, for instance, we can identify three sets of entities – medium (newspaper, magazine, and TV), resource (budget), and advertisement (linking medium and resource), which are described by the following nineteen attributes:

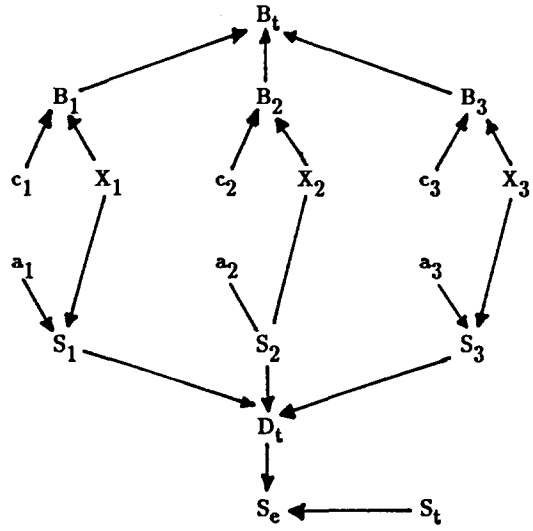
B_t = total number of customers;
B_1 = number of customers by newspaper ads.;
B_2 = number of customers by magazine ads.;
B_3 = number of customers by TV ads.;
c_1 = number of customers per newspaper ad.;
c_2 = number of customers per magazine ad.;
c_3 = number of customers per TV ad.;
X_1 = number of ads. run on newspaper;
X_2 = number of ads. run on magazine;
X_3 = number of ads. run on TV;
S_t = total amount of budget currently available;
D_t = total demand;
S_e = extra amount of money that may be available;
S_1 = total amount of money for newspaper ads.;
S_2 = total amount of money for magazine ads.;
S_3 = total amount of money for TV ads.
a_1 = unit cost for each newspaper ad.;
a_2 = unit cost for each magazine ad.;
a_3 = unit cost for each TV ad.;

These attributes can be grouped into seven classes. Fig. 6 shows how the entities and attributes are organized. Each node except the root in fig. 6(a) is an entity set. Each node except the root and node (2,1) is an attribute class. If we use u_1 to stand for the number of buyers, u_2 for the

(a) Derived Generic Graph



(b) Derived Elemental Graph



Notation: B_t = Total no. of customers; B_1 = No. of customers by newspaper;
B_2 = No. of customers by magazine; B_3 = No. of customers by TV;
c_1 = No. of customers per newspaper ad.; c_2 = No. of customers per magazine ad.;
c_3 = No. of customers per TV ad.; X_1 = No. of ads. on newspaper;
X_2 = No. of ads on magazine; X_3 = No. of ads. on TV;
S_t = Total available budget; D_t = Total ad. expenses; S_e = Extra budget;
S_1 = Total money for newspaper ads; S_2 = Total money for magazine ads.;
S_3 = Total money for TV ads.; a_1 = Unit cost for newspaper ads;
a_2 = Unit cost for magazine ads.; a_3 = Unit cost for TV ads.

Fig. 8. Model structures of the media selection problem.

number of runs and u_3 for dollar, the types of the attribute classes (S_t, S_e, and D_t are in the same class but have different value status) are:

B_t: [R, U, e_{31}, u_1]

B_1, B_2, B_3: [R, U, e_{32}, u_1]

c_1, c_2, c_3: [R, K, e_{33}, u_1/u_2]

X_1, X_2, X_3: [R, U, e_{22}, u_2]

S_t, S_e: [R, K, e_{11}, u_3]

D_t: [R, U, e_{11}, u_3]

S_1, S_2, S_3: [R, U, e_{12}, u_3]

a_1, a_2, a_3: [R, K, e_{13}, u_3/u_2].

5.2. Feature mapping

After the features of the new problem and their types have been identified, the MMS searches the case base for similar problems with known models. Measuring similarities between problems is the key in this step. [5] Strictly speaking, we can require that domains, units, value status, and hierarchical positions be exactly the same for similar attributes and all attribute to be similar for similar entities. In actual implementation, however, these rules may be relaxed. For instance, domain similarity may be relaxed to allow attributes defined on different but compatible domains to be similar. This relaxation would allow an integer program to be constructed based on its structural similarity to a linear program, because the only difference between them is the value domains of their variables.

Another issue of feature mapping is to determine the degree of similarity when no perfect analogy exists. A similarity metric is necessary in this case to compare partially similar cases and choose the one with the highest similarity score. The simplest similarity metric is to count the number of matching attributes and relationships. We can assign one point to each matching attributes or relationships and 0 to a mismatch. The overall similarity score is the number of proper matches divided by the total number of matches.

$$\text{Similarity score} = \sum_{i=1}^{m} \delta_i / m,$$

where m = total number of matches between two problems, and $\delta_i = 1$ for a proper match and 0 otherwise.

For example, a comparison between the media selection and product mix problems indicate that: (1) for each attribute class in the former, there exists a corresponding class in the latter; and (2) their entity graphs and attribute hierarchies (as shown in figs. 2 and 6) are isomorphic. Therefore, these two problems are similar, though they have different numbers of entity and attribute instances. The following mappings can be established between them. The numbers in parentheses indicate the numbers of instances. Relabeling is unnecessary in this particular example.

Media selection		*Product mix*		*Type*
B_t	(1)	T_p	(1)	[R, U, e_{31}, u_1]
B_i	(3)	T_{pi}	(2)	[R, U, e_{32}, u_1]
c_i	(3)	p_i	(2)	[R, K, e_{33}, u_1/u_2]
X_i	(3)	Q_i	(2)	[R, U, e_{22}, u_2]
D_t	(1)	D_j	(2)	[R, U, e_{11}, u_3]
S_t	(1)	S_j	(2)	[R, K, e_{11}, u_3]
S_e	(1)	Q_j	(2)	[R, K, e_{11}, u_3]
S_i	(3)	H_{ij}	(4)	[R, U, e_{12}, u_3]
a_i	(3)	a_{ij}	(4)	[R, K, e_{13}, u_3/u_2]

After a proper analogy is found, the system retrieves the model graphs of the analogue and then replaces their elements with counterparts to form model graphs for the new problem. The major operations in this step are *substitution* and *instantiation.* In our example, the structural type in fig. 7 can be instantiated to become that shown in fig. 8(a) and further elaborated to Figure 8(b).

Finally, the nodes and arcs in model graphs must be verified by the model builder to insure that the functions assigned to matching arcs are similar. The verification process focuses on two aspects of the analogical reasoning: Consistency and completeness. By consistency, we mean all labeling and transformation are done consistently. Any inconsistency can result in errors in the resulting model. The completeness criterion is necessary to ensure the formulated model covers all features in the problem. This is particularly important when the analogue is not a perfect one, i.e., some improper mappings exist in the analogy and the similarity score is less than 1.0 [25]. Theorem 2 and lemmas 1–3 can be used to facilitate the verification process. In the media

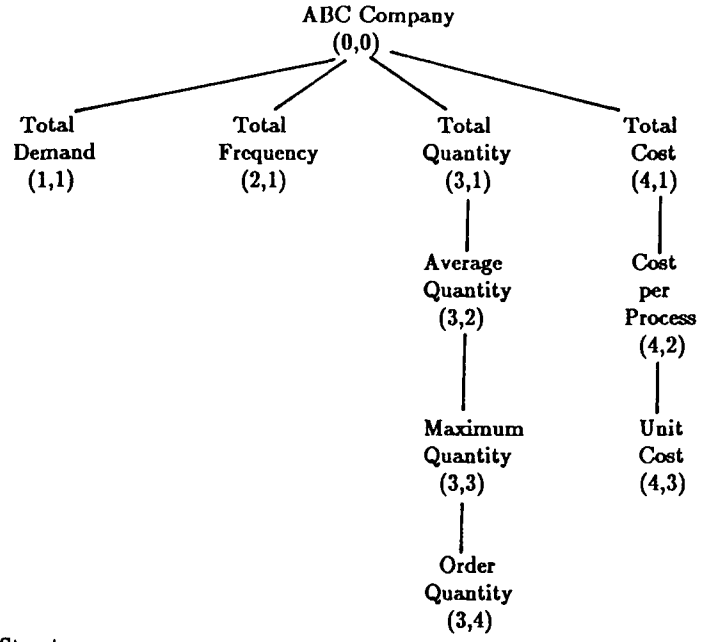
[5] Different approaches for measuring similarities may be used in different implementations.

selection and product mix problems, all relational operators and functors are members of the mathematical system covered by theorem 2, which guarantees that conceptual and structural similarities lead to functional similarity.

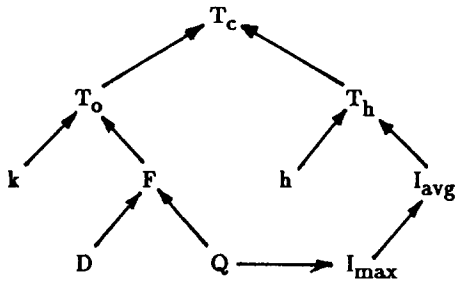
5.3. Model transformation

After functional similarity between two models is assured, the new model can be formulated. The basic principle governing model transformation is

(a) Concept hierarchy



(b) Model Structure



(d) Functions

$$T_c = T_o + T_h$$
$$T_o = h * F$$
$$F = D/Q$$
$$T_h = h * I_{avg}$$
$$I_{avg} = I_{max}^2/2Q$$
$$I_{max} = Q$$

Notation: T_c = Total cost; T_o = Total ordering cost; T_h = Total carrying cost; k = Unit ordering cost; F = order frequency; h = Unit carrying cost; I_{avg} = Average inventory; I_{max} = maximum inventory; D = demand; Q = Order quantity

Fig. 9. Representation of the existing inventory model (no shortage).

attribute substitution, which allows attributes associated with a function to be replaced with their counterparts in another problem. In our example, the media selection model can be constructed by simply replacing the parameters and variables of the Product Mix model by their counterparts found in feature mapping. The resulting model includes the following functions.

$B_t = B_1 + B_2 + B_3$ {Total customers},
$B_1 = c_1 X_1$ {Customers from newspaper},
$B_2 = c_2 X_2$ {Customers from magazine},
$B_3 = c_3 X_3$ {Customers from TV},
$S_1 = a_1 X_1$ {budget for newspaper},
$S_2 = a_2 X_2$ {budget for magazine},
$S_3 = a_3 X_3$ {budget for TV},
$D_t = S_1 + S_2 + S_3$ {Total desired budget},
$S_0 \geqslant D_t - S_t$ {Extra available budget}.

Combined with the goal of maximizing B_t, this set of functions can be easily aggregated to become the canonical form of the media selection model usually seen in textbooks, as follows.

$$\max \quad B_t = \sum_i c_i X_i,$$

$$\text{s.t.} \quad \sum_i a_i X_i \leqslant S_t.$$

5.4. Model validation

The final stage of analogical model construction is to validate the newly created model. It focuses on examining the usefulness of the developed model and is different from model verification that checks for consistency and completeness. Model validation can be conducted by comparing real and model-generated data. The model builder and the user are responsible for this process.

Generally speaking, model validity can be decomposed into: (1) technical validity, including data validity, logical and mathematical validity, and predictive validity; (2) operational validity concerning sensitivity analysis and implementation validity; and (3) dynamic validity which concerns maintaining and updating the model [3,13,35]. Validation of a model is a complicated topic nearly independent of the model development process. A detailed discussion is out of the scope of this article.

5.5. Application to model modification: An inventory example

In addition to constructing new models from a similar one, analogical modeling can also be applied to modify existing models. We use the inventory control problem in appendix A to illustrates model modification based on submodel similarity. The existing inventory model consists of two entities (ordering and carrying operations) and ten attributes that can be grouped into nine attribute classes as shown in fig. 9(a). The problem is how to add a new entity, inventory shortage, into the model to meet the new policy.

The attribute hierarchy shown in fig. 9(a) and the information available in the problem description allow us to determine the attribute types of the existing model as follows (u_1, u_2, and u_3 stand for dollar, unit of merchandise, and order, respectively).

T_c $[R, \text{U}, e_{41}, u_1]$ {total inventory cost},
T_h, T_o $[R, \text{U}, e_{42}, u_1]$ {total holding and ordering costs},
h $[R, \text{K}, e_{43}, u_1/u_2]$ {unit holding cost},
k $[R, \text{K}, e_{43}, u_1/u_3]$ {unit ordering cost},
I_{avg} $[R, \text{U}, e_{32}, u_2]$ {average inventory},
I_{max} $[R, \text{U}, e_{33}, u_2]$ {maximum inventory level},
Q $[R, \text{U}, e_{34}, u_2/u_3]$ {order quantity},
F $[R, \text{U}, e_{21}, u_3]$ {order frequency},
D $[R, \text{K}, e_{11}, u_2]$ {demand}.

Given the above information, the first step to modify the model analogically is to define the new entity needed in the new model. Four attributes are affiliated with the new entity

T_s $[R, \text{U}, e_{42}, u_1]$ {total shortage cost}
s $[R, \text{K}, e_{43}, u_1/u_2]$ {unit shortage cost}
S_{avg} $[R, \text{U}, e_{32}, u_2]$ {average shortage level}
S $[R, \text{K}, e_{33}, u_2]$ {maximum shortage allowed}

Then, the new attributes are compared and matched with those of the existing inventory

model. The following mappings can be found in this step.

$T_s \leftrightarrow T_o$ or T_h

$s \leftrightarrow h$

$S_{avg} \leftrightarrow I_{avg}$

$S \leftrightarrow I_{max}$

The first three matches are perfect, but the last one is imperfect. The value status of the maximum inventory (I_{max}) is unknown but the value status of the maximum shortage allowed (S) is known. These mappings suggest that the structure of the new entity is similar to either the substructure (T_o, h, I_{avg}, I_{max}) or the substructure (T_h, h, I_{avg}, I_{max}). As shown in fig. 9(b), the substructure (T_o, h, I_{avg}, I_{max}) does not make sense. Hence, substructure (T_h, h, I_{avg}, I_{max}) serves as a basis for building a new structure that links T_s, s, S_{avg}, and S. Since Q links to I_{max} and T_h links to T_c, similar arcs between S and I_{max} and between T_s and T_c can be installed. The resulting graph for the new model is shown in fig. 10. The bold lines indicate the new substructure built analogically.

Finally, new mathematical equations can be derived by substitution and integration. The new model is as follows.

$T_c = T_o + T_h + T_s,$

$T_o = hF,$

$F = D/Q,$

$T_h = hI_{avg},$

$I_{avg} = I_{max}^2/2Q,$

$T_s = sS_{avg},$

$S_{avg} = S^2/2Q,$

$I_{max} = Q - S.$

They can easily be simplified to the following form usually seen in textbooks.

$$T_c = kD/Q + h(Q-S)^2/2Q + sS^2/2Q.$$

5.6. Case-based learning

Learning can take place in the whole process of case-based analogical reasoning. The most important stage, however, is after the model has been formulated and applied. There are many different types of model management knowledge that can be learned by machines. For example, machines may learn that a model representation is inappropriate and revise it. The machine may also learn that certain knowledge is necessary in determining the applicability of a particular type of models.

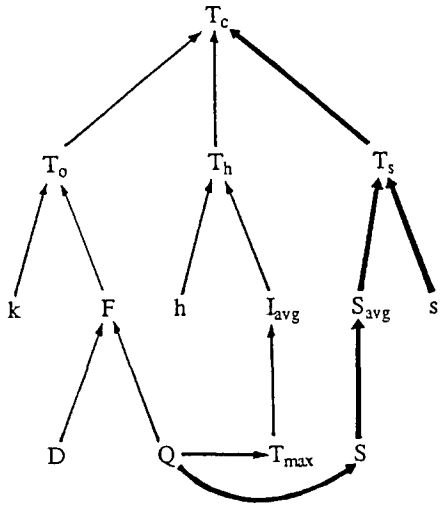
For case-based MMS, several types of learning can be implemented. The rudimentary form is to store the new case in the case base for future use. The system simply memorizes all cases. Because of the complexity in today's decision environment, it is impossible to store all cases in the case base. Selectivity is important. Therefore, before storing a case, the system needs to determine whether it is worth memorizing. This involves assessing the cost and benefit of memorizing vis-a-vis forgetting a case. General heuristics for case storage are:

(1) the system stores unique rather than duplicate cases;
(2) If a case happens to be a superset of another, then the system stores the case and delete the later.

Considering the examples we discussed previously, the media selection model is the same as the product mix model and hence will not be stored in the case base. The inventory model with shortage allowed is a superset of the old inventory model. Therefore, the new one should overwrite the old one.

The similarity measure we used in analogical reasoning can also help determine which case to store. If an analogy is perfect (i.e., the similarity score equals 1.0), then keep the old case and forget the new one because they are essentially the same. If the analogy is imperfect, then we have two choices. First, we store both cases. This is necessary when the difference between the two cases are significant. Second, instead of storing the new case, we identify the difference between the two cases and generalize the difference into rules. Inductive and explanation-based learning methods are useful for generalization. Once rules are learned, the new cases can be forgotten. Fig. 11 illustrates the process.

Inductive or explanation-based learning can generate rules that are useful to classify and index cases in the case base, identify case similarity, evaluate case transformation, repair flaws,



Notation: T_s = Total shortage cost; S_{avg} = Average shortage; s = Unit shortage cost; S = Maximum shortage.

Fig. 10. Structure of the new inventory model (shortage allowed).

and govern system learning. Case classification and indexing are important to increase the efficiency when the case base grows to a certain degree. Similar cases can be grouped into classes, which allows analogical retrieval of similar cases to be performed at two levels: (1) determining the class to which the new problem belongs; and (2) locating the exact case similar to the problem. For example, the inventory control problem can be classified as an optimization problem and only

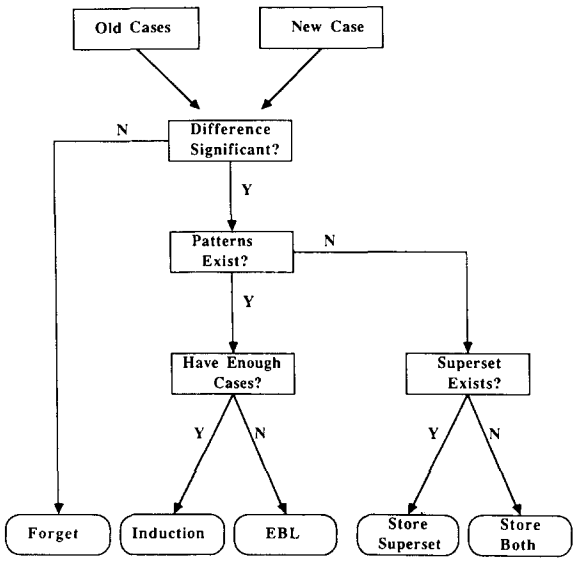


Fig. 11. Case-based learning process.

compared with cases in the same class. Case-based learning may allow classes to grow or to be redefined. In addition, if the cases in a certain class are found to be similar enough to form rules, inductive learning techniques can be applied to generate the rules. These rules can replace the cases to improve the efficiency of the reasoning process.

Knowledge for determining case similarity can also be refined in case-based learning. We defined a similarity measure previously. This measure can be refined to become more accurate. Another set of knowledge that can be learned is rules used for evaluating the analogical mapping. In our approach, semantics play an insignificant role in analogical reasoning. In some cases, semantic knowledge may be incorporated. Learning can also occur even if the analogical reasoning fails. Knowledge can be obtained by comparing successful and failed cases to improve the effectiveness of the reasoning process.

6. Concluding remarks and further research

In this article, an approach that applies case-based analogical reasoning and learning to model formulation has been presented. The discussion began with a definition of case-based analogical reasoning and learning. Then, problem and model similarities were defined at conceptual, structural, and functional levels. Conceptual similarity is measured on individual attributes and entities. Structural similarity is determined by structural isomorphism or homomorphism. Functional similarity is determined by the types of relational and functional operators. Finally, the process for analogical reasoning was presented and knowledge that can be learned in the process was discussed.

The contribution of this research is two-fold. First, it initiates research in an area that has been widely used in model construction practice but underexplored in existing modeling literature. Second, the discussion of model similarity, analogical reasoning, and case-based learning provides a foundation for further research in developing intelligent MMS. Sample issues for future research are listed below.

First, how can analogical modeling be used in a more general way? In this research, the measure of model similarity is relatively rigid, which

requires good matches in domains, unit patterns, value status, and hierarchical positions. Although this reduces the risk of finding poor analogies, it also reduces the chance of finding creative analogies. Sometimes, the new problem may be unclear and the modeler may be unable to define all concepts at the beginning. In this case, supporting partial matches and multiple ways of viewing a case is essential to finding creative analogies. Both machine and human learning are necessary to contribute to this area.

Second, what are the computational costs involved in the case-based reasoning process? A major part of analogical reasoning is feature mapping, which is theoretically NP-complete. We use generic graphs to reduce the computational cost. Compared with the rule-based approach, however, case-based reasoning may still be too expensive, especially when the case base is large and poorly-indexed. Research is necessary to balance the use of rules and cases and develop good indexing mechanisms.

Third, how can large-scale models whose structures cannot be simplified by structural isomorphisms be built efficiently? One way to overcome the problem is to break the model into submodels. The submodels built separately can then be integrated to form a larger model. In order for this approach to work without leading to suboptimization, however, theories and guidelines for model decomposition and integration must be developed or learned. This is another interesting area for future studies.

Finally, how can a particular learning method be chosen to maximize the learning effect? Although we discussed the application of case-based learning after model formulation, no specific details of a particular method are provided. This is primarily because there are many potential machine learning methods that can be applied to different stages of the case-based reasoning and learning process. Each of the matches between learning methods and processes for case-based reasoning can be a research project for the future.

References

[1] K.D. Ashley and E.R. Rissland, Compare and Contrast: A Test of Expertise, in: AAAI-87 Proceedings (Morgan Kaufmann, San Mateo, CA, 1987) 273–278.

[2] W.M. Bain, Case-based Reasoning: A Computer Model of Subjective Assessment, Ph.D. Dissertation (Yale University, 1986).

[3] O. Balci and R.G. Sargent, Bibliography on Validation of Simulation Models, Newsletter – TIMS College in Simulation and Gaming (Spring 1980) 11–15.

[4] R.W. Blanning, Issues in the Design of Relational Model Management Systems, AFIPS Conference Proceedings (1983) 395–401.

[5] R.W. Blanning, A Relational Framework for Joint Implementation in Model Management Systems, Decision Support Systems 1, No. 1 (1985) 69–81.

[6] R.W. Blanning, A Relational Framework for Assertion Management, Decision Support Systems 1. No. 2 (1985) 167–172.

[7] R.W. Blanning, An Entity-Relationship Approach to Model Management, Decision Support Systems 2, No. 1 (1986) 65–72.

[8] J.G. Carbonell, Learning by Analogy: Formulating and Generalizing Plans From Past Experience, in: R.S. Michalski et al., Eds, Machine Learning: An Artificial Intelligence Approach, Vol. I, (Morgan Kaufmann, San Mateo, CA, 1983).

[9] D.V. Dalen, Logic and Structure, 2nd ed. (Springer, Verlag, Berlin, 1983).

[10] DARPA, Proceedings of Case-Based Reasoning Workshop (Morgan Kaufmann, San Mateo, CA, May 1989).

[11] N. Dershowitz, Programming by Analogy, in: R.S. Michalski et al., Eds., Machine Learning: An Artificial Intelligence Approach, Vol. II, (Morgan Kaufmann, San Mateo, CA, 1986) 395–423.

[12] D.R. Dolk and B.R. Konsynski, Knowledge Representation for Model Management Systems, IEEE Transactions on Software Engineering 10, No. 6 (1984) 619–628.

[13] S.I. Gass, Decision-Aiding Models: Validation, Assessment, and Related Issues for Policy Analysis, Operations Research 31, No. 4 (1983) 603–631.

[14] A.M. Geoffrion, Introduction to Structured Modeling, Management Science 33, No. 5 (1987) 547–588.

[15] A.M. Geoffrion, SML: A Model Definition Language for Structured Modeling, Working Paper No. 360 (UCLA, Los Angeles, CA, May 1988).

[16] K.J. Hammond, Case-Based Planning (Academic Press, Boston, MA, 1989).

[17] M. Hesse, Models and Analogies in Science (Notre Dame University Press, 1966).

[18] G. Klein, Developing Model String for Model Management, Journal of MIS 3, No. 2 (1986) 94–110.

[19] J.L. Kolodner, Extending Problem Solving Capabilities Through Case-Based Approach, in: J.L. Kolodner, Ed., Proceedings of a Workshop on Case-based Reasoning (Morgan Kauffman, San Mateo, CA, 1988).

[20] R. Krishnan, "A Logic Modeling Language for Automated Model Construction, Decision Support Systems 6 (1990) 123–152.

[21] T.-P. Liang, A Graph-Based Approach to Model Management, Proceedings of the International Conference on Information Systems (San Diego, CA, 1986).

[22] T.-P. Liang, Reasoning in Model Management Systems, Proceedings of the 21st Hawaii International Conference on Systems Sciences, IEEE Computer Society Order Number 843, Vol. 3 (1988) 461–470.

[23] T.-P. Liang, Development of a Knowledge-Based Model Management System, Operations Research 36, No. 6 (1988) 849–863.
[24] T.-P. Liang, Reasoning for Automated Model Integration, Applied Artificial Intelligence 4 (1990) 337–358.
[25] T.-P. Liang, Modeling By Analogy: A Case-Based Approach to Automated Linear Program Formulation, HICSS-24 Proceedings, Vol. III, IEEE Press (1991) 276–283.
[26] T.-P. Liang, SAM: An Analogical Modeling Language for Structured Modeling, Working Paper (Krannert Graduate School of Management, Purdue University, 1991).
[27] T.-P. Liang and C.V. Jones, Meta-Design Considerations in Developing Model Management Systems, Decision Sciences 19, No. 1 (1988) 72–92.
[28] T.-P. Liang and B.R. Konsynski, Modeling by Analogy: Use of Analogical Reasoning in Model Management Systems, The First ISDSS Proceedings (Austin, TX, 1990) 405–421.
[29] P. Ma, F.H. Murphy and E.A. Stohr, A Graphics Interface for Linear Programming, Communications of the ACM 32, No. 8 (1989) 996–1022.
[30] W.A. Muhana and R.A. Pick, Composite Models in SYMMS, Proceedings of the 21st Hawaii International Conference on Systems Sciences, IEEE Computer Society Order Number 843, Vol. 3 (1988) 418–427.
[31] F.H. Murphy and E.A. Stohr, An Intelligent System for Formulating Linear Programs, Decision Support Systems 2, No. 1 (1986) 39–47.
[32] D. Navinchandra, K.P. Sycara and S. Narasimhan, Behavioral Synthesis in CADET: A Case-Based Design Tool, Proceedings of the 7th IEEE Conference on Artificial Intelligence (Miami, FL, Feb. 1991).
[33] M. Redmond, Distributed Cases for Case-Based Reasoning; Facilitating Use of Multiple Cases, AAAI-90 Proceedings (1990) 304–309.
[34] C.K. Riesbeck and R.C. Schank, Inside Case-based Reasoning (Lawrence Erlbaum, Hillsdale, NJ, 1989).
[35] R.E. Schellenberger, Criteria for Assessing Model Validity for Managerial Purposes, Decision Sciences 5, No. 4 (1974) 644–653.
[36] M.J. Shaw, P. Tu and P. De, Applying Machine Learning to Model Management in Decision Support Systems, Decision Support Systems 4, No. 3 (1988) 285–305.
[37] R.L. Simpson, A Computer Model of Case-Based Reasoning in Problem Solving: An Investigation in the Domain of Dispute Mediation, Technical Report GIT-ICS-85/18 (School of Information and Computer Science, georgia Institute of Technology, 1985).
[38] M.M. Sklar, R.A. Pick and W.H. Sklar, An Automated Approach to LP Formulation, Working paper (Department of QAIS, University of Cincinnati, 1989).
[39] R.J. Sternberg, Intelligence Information Processing and Analogical Reasoning (Lawrence Erlbaum, Hillsdale, NJ, 1977).
[40] R.J. Sternberg, Reasoning, Problem Solving, and Human Intelligence (Cambridge University Press, Cambridge, 1982).
[41] E.P. Sycara, Resolving Adversarial Conflicts: An Approach to Integrating Case-Based and Analytic Methods, Technical Report GIT-ICS-87/26 (School of Information and Computer Science, Georgia Institute of Technology, 1987).
[42] B.P. Zeigler, Multifacetted Modelling and Discrete Event Simulation (Academic Press, New York, 1984).

Appendix A: Sample problems

Product mix problem ABC company produces dot-matrix and laser printers. A dot-matrix printer takes two hours to assemble and 10 minutes to inspect. A laser printer takes one hour to assemble and 30 minutes to inspect. The company has one inspection and two assembly lines; each of which operates no more than 40 hours a week. No overtime is allowed. The profits of producing dot-matrix and laser printers are \$100 and \$200 per unit, respectively. How many dot-matrix and laser printers should the company produce per week to maximize its profit.

Media selection problem ABC Company advertises its printers in newspapers, in magazines, and on television. Based on findings from marketing research, each dollar spent in newspaper advertisement reaches 10 potential customers, each dollar in magazines reaches 12 potential customers, and each dollar in TV reaches 15 potential customers. It costs \$500 to run a newspaper advertisement, \$700 for magazines, and \$1000 for TV. The company has decided to spend no more than \$20 000 on advertising. How should the company budget for each medium in order to reach the maximum number of potential buyers?

Inventory problem ABC Company has decided to change its inventory control from a no-shortage policy to allow a maximum shortage of S units. The demand, ordering and unit holding costs remain the same, i.e., D (units), k (\$/order), h (\$/unit), respectively. The unit shortage cost is estimated to be s (\$/unit). How can the existing inventory control model (as shown in fig. 9) be modified to determine the economic order quantity (Q (unit/order) that meets the new policy?

Appendix B: Proof of theorem 2

First, Similar structures imply that for each function in a model there is a corresponding

function with the same number of components in its counterpart. Then, we have to prove that the relational operators and functors in these functions have the same type. Since all relational operators are binary, this meets at least the requirements for a weak functional similarity. The functors fall into two categories: Binary (including [+, −, *, /]) and unary (including [exp, sqr, cub]). By definition, two isomorphic structures have one-to-one mappings between nodes and between arcs. Therefore, each pair of corresponding relations should be in the same category (i.e., either both are unary or both are binary). In the system, different unary functors result in different units (e.g., applying a square function to a distance measure generates a unit of square foot; whereas applying a cubic function generates a unit of cubic foot). Hence unit similarity prevents from assigning different functions to a pair of matching arcs in two similar hierarchies. For the binary functors, only plus and minus generate the same unit. If we consider minus as a special case of the Σ functor, then they are equivalent. Therefore, we prove that two models holding conceptual and structural similarities must have at least a weak functional similarity.