Learning Design Plans in a Knowledge-Based Blackboard System

Jenmu Wang
Department of Civil Engineering, Tamkang University
Tamsui, Taipei Hsien 251, Taiwan

Abstract

The control information along a decision route that leads to the creation of a design solution is often referred to as design plan. Knowledge-based design systems frequently use precompiled design plans to control and schedule design activities. It is, however, difficult for design systems to learn new plans. This article introduces the concept of memory-oriented learning and presents an approach to learn design plans from normal design sessions in a blackboard design model.

Keyword: *Design Plans; Machine Learning; Structural Design; Knowledge-Based Systems*

1: Introduction

The knowledge embodied in most of today's knowledge-based systems is static in time. That is, they do not expend or modify their knowledge base through their own experience. If the system response incorrectly for a given problem, it will give the same incorrect answer for that problem until a knowledge engineer modifies its knowledge base. The inability to save previous experiences for future application and modification represents a serious shortcoming of most knowledge-based systems today. Even at the first time, put knowledge into a knowledge base is not an easy job. Knowledge acquisition is always refer to as the bottleneck of building expert systems. Therefore, how to apply machine learning techniques to improve the performance of design systems has been an active research area recently [1]. The following discussion characterizes design systems in terms of the model of learning.

Under the modular and multi-layer concept, a learning system can be divided into a performance element and a learning element [2]. The **performance element** is a problem solver, independent of the learning element. That is it can be run without the learning element. The **learning element** uses machine learning techniques to improve the problem solving ability of the performance element. From the machine learning literature, learning can be categorized to memory-oriented learning and rule learning:

Memory-oriented learning stores previous experiences in order to replay the derivation of their solutions as a reasoning shortcut. So, the core issue of memory-oriented learning is how to organize these experiences so that they can be used again in the appropriate situations. Memory-oriented learning is in fact a learning by analogy process.

Rule learning learns heuristic rules from previous experiences. The rule-base is incrementally refined and created during the normal use of the system. Rule learning is actually an automated knowledge acquisition and skill refinement process.

Two steps are necessary to learn abstracted reasoning rules. Credit and blame must be assigned to the performance of a problem solver first. The rule base then can be modified based on the criticism. A common critical technique is the *ideal trace*, used in LEX [3]. The ideal trace of a solution is compared with the actual trace to identify the positive and negative training instances. The former correspond to the rules that behaved correctly and the latter correspond to the rules that behaved incorrectly.

Some rule modification techniques that can be use to refine the rule-base are summarized in [4]. These are: (a) prioritizing the rules, (b) instantiating a rule, (c) adding extra conditions to a rule's hypothesis, (d) proposing a new heuristic rule, and (e) updating the hypothesis of an incompletely learned rule.

The intelligent computer-aided design system presented in this paper can be conceptually divided into a performance element and a learning element as discussed above. The performance element is based on a blackboard AI architecture that explicitly controls and reasons about its design behavior. The learning element of the system employs memory-oriented learning to capture design strategies as design plans.

0-8186-8218-3/97 $10.00

The following sections of the paper provide an overview of the intelligent design system and describe its representation and acquisition of design plans.

2: Blackboard design model

In a typical blackboard problem-solving model [5 and 6], the knowledge needed to solve a problem is partitioned into independent knowledge sources that are grouped into several knowledge modules in the knowledge base. The knowledge sources modify only a global knowledge structure (the blackboard) and respond opportunistically to the changes on the blackboard. An inference mechanism sequences the execution of knowledge sources according to the current control strategies, and updates the solution and control strategies on the blackboard accordingly. The blackboard model was selected because of its explicit control structure that can easily separate domain and control information and facilitate the capture and replay of design plans.

Usually, the solution space of a design problem is too big to search. Decomposition is a common strategy used by engineers to divide the search space. Domain specific knowledge and general problem-solving skills are required to control the decomposition and to search effectively for the reduced design space. However, the complexity of individual subproblem and the interaction between subdesigns can still make the design an iterative process. The generated design has to be tested against all the design constraints and relevant parts of the design must be modified to overcome any constraint violation. This process may have to be repeated several times to produce an acceptable design.

According to the characteristics of underconstrained parametric design and the blackboard reasoning paradigm, an underlying design model was developed. Designs are divided into subtasks in the design system, and the knowledge for solving individual subtasks is stored in a knowledge base. The problem solver uses that knowledge to develop design solutions through a generate-test-modify paradigm. Top-down decomposition plans are also stored in the knowledge base. A plan represents a design strategy that consists of a sequence of design steps. Each step in a plan is called a goal; goals guide the problem solver toward the desired solution. Different levels of abstraction of plans can help decompose and organize design knowledge. A plan can specify a sequence of goals that produce a subdesign. A plan can be a top level design strategy that points to other plans.

The problem solver checks design constraints throughout the design as appropriate. Whenever a constraint violation is found, the problem solver has to overcome the design failure by redesign (that is, modifying the partial design). Redesign in the system is based on dependency-directed backtracking with knowledge-based advice. The problem solver's truth maintenance system (TMS) provides dependency links for a design failure. The dependency network is built during design. The design variables involved in the current design failure can be traced through the links. The problem solver analyzes the dependency information and the violated constraints to suggest what part of the partial design to modify and how to fix it. The TMS updates the current design according to the modification. Design values that are no longer valid (as well as values derived from those values) are removed, and the problem solver proceeds from there to its generate-test-modify cycle.

3: System overview

The problem-solving process of the design system is controlled by a scheduler using the activated design strategies (plans and goals) on the blackboard. This section describes the interpretation, rating and scheduling of design actions and the decomposition of design strategies. The discussion focuses on the general concepts of these activities, which lays the groundwork for a more detailed presentation of the plan learning mechanism in the system.

The blackboard of the design system is divided into a design information blackboard (containing design data generated by various actions) and a control blackboard (consisting of the control decisions the system uses to schedule activities). The design solution is generated on the solution blackboard incrementally by applying design actions one at a time. The condition under which a knowledge source (KS) can be executed is specified by a blackboard state with a set of variables to represent different triggering contexts, and the action of the KS is also specified by the same set of variables to represent alternative actions under different contexts. All knowledge sources operate simultaneously at every cycle and generate knowledge source activation records (KSARs) when specified trigger conditions and blackboard contents exist.

No preset order of execution is defined in the system. It dynamically selects a sequence of actions (KSARs) to produce a design according to the design strategies that are in use. At each design step, the schedulers of the system select one KSAR for execution based on how well the action of a certain KSAR matches the intent of the current design strategies on the control blackboard. The design strategies are represented as plans and goals. A goal is a design consideration that the system uses to rate

KSARs. A plan is a sequence of design goals that represents a design strategy.

Different levels of abstraction of design plans can help decompose and organize design knowledge. A plan can specify a sequence of goals that produce a subdesign. A plan can be a top-level design strategy that points to other plans. Therefore, a global design goal can be dynamically expanded into a more detailed design plan (or plans) at run time. The individual goals of that plan may be further expanded if specific plans are available. This top-down plan decomposition with several levels of abstraction is called goal expansion. An Example is given in Figure 1. It is possible that several expansions are available for a given design goal. Therefore, the system frequently has multiple design plans on the blackboard at once.

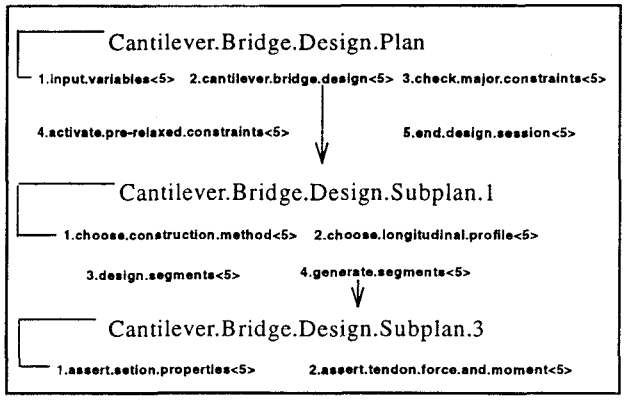

Figure 1: An Example of Goal Expansion

The rating of a KSAR against a plan on the control blackboard is a function of the priority of the action against current goals of the plan and the importance of different goals. There may be more than one plan on the control blackboard. In that case, the rating of a KSAR with respect to all plans is the weighted and normalized sum of the individual ratings. The weighting process takes into account the different importance of plans. The rating of every KSAR is a numeric value ranging from zero to one hundred, which represents a "no match" to a "perfect match". At every cycle, the scheduler selects the highest rated KSAR to execute (or the user can override the scheduler by selecting a different KSAR).

4: Representation of design plans

Knowledge representation in the design system is based on an object-oriented approach. Design strategies are stored as plan and goal objects in the knowledge base. When design plans and goals are posted on the control blackboard, the system regards them as the control knowledge of the current design session and uses them to rate and schedule design actions. The top-level design plan (Cantilever.Bridge.Design.Plan) in Figure 1 is given in Figure 2 as an example. The following are explanations of the properties of plans:

- GOAL.LIST — a list of goal objects to be activated sequentially. Each element of the list can be a single goal object or a list of goals to be activated at the same time.
- INTENTION—a LISP form that indicates when the plan needs to be deactivated. When it evaluates to true, either the intention of the plan is satisfied or it is no longer applicable.
- WEIGHT — a number between 1 and 10 (inclusive), representing the importance of the plan. It is used in KSAR rating to account for different priorities of design plans. The default value is 5. It can be changed by the user when creating the plan object or by control KSs during design. A plan with a higher weight value is considered more important than a plan with a lower weight and is more influential in scheduling.
- ACTIVE.GOAL—the goal (or goals) currently active in this plan. It is an internal attribute used by the blackboard maintenance mechanism. The default value is NIL, and it will be set to an appropriate value during blackboard updating.
- REMAINING.GOAL.LIST—a list of goals that remain to be activated in this plan. It is an internal attribute with a default value of NIL, and it will be set to an appropriate value by the blackboard maintenance mechanism.

INTENTION:	Valid while the cantilever bridge design parameters (e.g., construction method, longitudinal profile, span depth ratio, slab thickness, tendon type, number of tendons, etc.) are undecided.
GOAL.LIST:	(input.variable cantilever.bridge.design check.major.constraints activate.pre- relaxed.constraints end.design.session)
WEIGHT:	7
ACTIVE.GOAL:	cantilever.bridge.design
REMAINING.GOAL.LIST:	(check.major.constraints activate.pre-re- laxed.constraints end.design.session)

Figure 2: Cantilever.Bridge.Design.Plan

A goal is the primary rating object for the blackboard design system. It can be a part of a design plan or a stand-alone design consideration. All activated goals on the blackboard are used to rate KSARs. The attributes of goals are explained below and an example is given in Figure 3.

- FUNCTION—a LISP form that returns a value between 0 and 100. This is the rating function that evaluates how well a KSAR serves the goal. The higher the returned number is the better the KSAR is for achieving the goal.
- INTENTION—a LISP form that indicates when the goal needs to be deactivated. When it evaluates to true, either the intention of the goal is satisfied or it is no longer applicable.
- WEIGHT — a number between 1 and 10 (inclusive), representing the importance of the goal. It is used in KSAR rating to account for different priorities of design goals. The default weight of a goal is 5. It can be changed by the user when creating the goal object or by control KSs during design. A goal with a higher weight value is considered more important than a goal with a lower weight and is more influential in scheduling.
- INCLUDE.IN—the plan object to which the goal belongs. If the value is NIL, the goal is not attached to any plan. Usually, a stand-alone goal is an important design decision that needs immediate attention. For example, stand-alone redesign goals fix constraint violations.

FUNCTION:	Favor actions instantiated from subclasses of cantilever bridge design generator.
INTENTION:	Valid while the cantilever bridge section properties (e.g., web thickness, top and bottom slab thickness, etc.) are undecided.
WEIGHT:	5
INCLUDED.IN:	Cantilever.Bridge.Design.Plan

Figure 3: The Cantilever.Bridge.Design Goal

5: Learning design plans

After a design session is done, design information is still stored on the blackboard, which includes problem inputs, final solution, intermediate solutions, variable value justifications and design history (a sequence of the executed KSs and their bindings). It is time for the learning element to perform memory-oriented learning. That is to learn design strategies from the just finished design. However, the design strategies do not explicitly exist on the blackboard. The strategy recorder of the learning element analyzes the design history and constraint status as well as other information on the blackboard and captures design strategy in the form of design plans and goals. A design memory is then used to store the recorded plans and goals. The memory is divided into two parts: a memory of plans and a memory of goals.

A recorded plan is a subclass of the plan object discussed in the previous section. It stores the problem-solving or backtracking strategy of a particular previous design. It specifies the sequence of design goals achieved by the previous design steps. The recorded goals will be discussed later in this section.

The strategy recorder in the learning element of the design system is responsible for creating recorded plan objects and saving them in the memory. The control knowledge of a previous design session is abstracted to one global design plan and several redesign plans. The process that the strategy recorder uses to capture design plans is stated below:

- **Identify major design actions.** The design history is analyzed, and all KSs that modified the solution blackboard are gathered. This step is to filter out unnecessary design steps that do not directly contribute to the solution process (e.g., control KSs that only modify the control blackboard).
- **Create associated goals.** A recorded goal is created for each identified major action (KS) in order to prefer the same KS in the future. This process is discussed in detail later.
- **Differentiate design and redesign goals.** The major design actions can be classified into design and redesign actions. Therefore, the recorded goals are assigned to one design plan that represents the major design path and several redesign plans that represent the various backtracking processes. The goals then make up the goal list of their plan.
- **State the intention of the plans.** The intention of a global plan is to generate design value for all the design attributes and to satisfy all the applicable constraints of the design. The intention of a redesign plan is to satisfy all unsatisfied constraints that triggered the redesign process.

The strategy recorder is also responsible for creating recorded goal objects and saving them in the goal memory. A recorded goal represents one step of the recorded plan that it belongs to. It contains a rating function to evaluate the usefulness of future KSARs for reproducing

the effect that resulted from the past action taken at that step.

Three different rating levels are created for each recorded goal:

- **High rating**—A high rating is given to the KSARs instantiated from the same KS that previously triggered the KSAR used to accomplish the goal (i.e., KSARs from the same KS).
- **Moderate rating**—KSARs that modify the same design objects and attributes attain a moderate rating.
- **Limited rating**—Same types of actions (i.e., KSARs with same parent class as the previously used KSAR) score a limited rating.

In this way, each step of a recorded plan can be followed precisely, closely or loosely by the system's control mechanism when the plan is reused (that is, reposted on the blackboard). However, further experiment shows that the level of backtracking is the most important guidance that a recorded backtracking goal can give to later designs. The backtracking level is recorded with the bindings of the previously executed backtracking KSAR. Therefore, to capture that decision, the following rating level is added to the top of the rating condition list for recorded goals that are extracted from previous backtracking provoking actions. That is four rating conditions are used instead of three. The highest rating is given by the condition stated below:

- **Excellent rating** —The best rating is given to the KSARs that are the same as previously used to accomplish the redesign goal (i.e., KSARs from the same KS with the same bindings).

The intention of a recorded goal is the negation of the trigger condition of the KS that previously accomplished the design step. A goal is not applicable when its intention is true. The action is not appropriate or is already executed when its trigger condition is not true. Therefore, when the trigger condition of a previous action is not true, the recorded goal denoting that action is not applicable (either the goal is already accomplished or the intended purpose of the goal is not desirable). The formulation works well with the plan and goal updating mechanism of the blackboard design system. Inapplicable goals from past plans are delayed or skipped under this setup.

6: Conclusions

This approach was tested for structural design of balanced cantilever bridges. The recording and reuse of design plans and goals in the blackboard design system are fundamentally supported by the underlying model of the system, and the model directly influences the representation of design plans and goals. The control knowledge captured is of little use unless it can be recognized by system's control mechanism. Therefore, the recorded plans and goals are expressed in a format that can later be directly posted on the control blackboard by control actions to change the behavior of the design system.

The initial result of this research revealed the difficulties of capturing design strategies as design plans. While recording the designer's steps as design plans, the system can capture the explicit basis of the design plan, such as the ingredient design attributes, satisfied constraints, unsatisfied constraints, performed actions, etc. However, the implicit supporting reasoning behind design plans can not be acquired (e.g., judgment based on unrepresented design considerations and analysis of values from several objects, etc.). The system assures the fundamental appropriateness of transferring previous plans. However, the good result of plan reuse is not promised. To fully assess the correctness of plan reuse, the system needs to capture the undeclared reasons associated with individual human design decision. Furthermore, the system can not justify design steps of its user. It only records the designer's steps and assumes their correctness. A possible solution to these problems is to build an interface to let the user provide justifications whenever the system's decisions are overridden.

References

1. Duffy, Alex H.B., Brown, David C. and Maher, Mary Lou, "Special Issue: Machine learning in design," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 10, pp. 81-82, Cambridge University Press, 1996.
2. Smith, Reid G., Mitchell, Tom M., Chestek, Richard A. and Buchanan Bruce G., "Model for Learning Systems," *Proceedings of the Fifth IJCAI*, Fifth International Joint Conference on Artificial Intelligence, Cambridge, Massachusetts, pp. 338-343, August, 1977.
3. Mitchell, Tom M., *Learning and Problem Solving*, Technical Report LCSR-TR-45, Department of Computer Science, Rutgers University, New Brunswick, New Jersey 08903, June 1983.
4. Bundy, Alan, Silver, Bernard and Plummer, Dave, An Analytical Comparison of Some Rule Learning Programs, Research Paper No. 215, Department of Artificial Intelligence, University of Edinburgh, 1984.
5. Hayes-Roth, Barbara, "Blackboard Architecture for Control," *Artificial Intelligence*, Vol. 26, pp. 251-321, 1985.
6. Nii, H. Penny, "Blackboard Systems : The Blackboard Model of Problem Solving and Evolution of Blackboard Architectures—Part One," *AI Magazine*, Vol. 7, No. 2, pp. 38-53, 1986.