January 1985

Report No. STAN-CS-85-1036
Also numbered: HPP-85-2

Learning Control Heuristics in BB1

by

Barbara Hayes-Roth and Micheal Hewett

Department of Computer Science

Stanford University
Stanford,CA **94305**

Heuristic Programming Project
Report No. **85-2**

January **7, 1985**

Learning Control Heuristics in BB1[1]

Barbara Hayes-Roth and Micheal Hewett

January. **1985**

[1] **This work was supported by a DARPA grant to the Heuristic Programming Project**

Abstract

BB1, a blackboard system building architecture, ameliorates the knowledge acquisition bottleneck with generic knowledge sources that learn control heuristics. Some learning knowledge sources replace the knowledge engineer, interacting directly with domain experts. Others operate autonomously. The paper presents a trace from the illustrative knowledge source, Understand-Preference, running in PROTEAN, a blackboard system for elucidating protein structure. Understand-Preference is triggered when a domain expert overrides one of BB1's scheduling recommendations. It identifies and encodes the heuristic underlying the experts scheduling decision. The trace illustrates how learning knowledge sources exploit BB1's rich representation of domain and control knowledge, actions, and results.

1. The Problem: Learning Control Heuristics

Knowledge acquisition is widely recognized as a major bottleneck in knowledge engineering [1]. Articulating and coding domain knowledge is time-consuming for both the domain expert and the knowledge engineer. Acquiring control knowledge poses additional problems [4, 5]. Control knowledge appears to be more difficult for experts to retrieve than domain knowledge and they have difficulty distinguishing domain and control knowledge. **Experts** produce general heuristics **during** questioning. but use more specific heuristics during problem-solving. Stimulating **experts' retrieval** of a comprehensive set of heuristics may require analysis of many example problems that produce no new domain knowledge.

This paper discusses automatic learning of control heuristics in BB1, a blackboard system-burlding architecture [6]. Section 2 briefly overviews BB1. Section 3 discusses BB1's approach to learning. Section **4** presents a program trace from an illustrative learning knowledge source, Understand-Preference. Section 5 presents conclusions.

All examples are drawn from PROTEAN [2], a BB1 system that attacks the following problem:

> Given a test protein's primary structure sequencing individual amino acids; its secondary structure locating alpha helices, beta sheets, and random coils; NOEs indicating proximate pairs of constituent atoms; vanderWaals radii indicating minimum distances between pairs of structures; indication of which atoms lie on the protein's surface; description of the general shape of the protein; and specification of the structures of individual amino acids:
>
> Determine the (possibly dynamic) locations of all constituent atoms in three-dimensional space.

2. Background: Overview of BB1

BB1 is a domain-independent architecture for building AI systems that control their own problem-solving behavior, explain their problem-solving behavior in terms of an underlying control plan, and learn new control heuristics from experience. The "blackboard control architecture" [7] underlying BB1 extends the standard blackboard architecture [3] in three ways.

1. The blackboard control architecture defines explicit domain and control blackboards.

As in the standard architecture, the blackboard records all solution elements generated during problem-solving and organizes them in different levels and solution intervals. The domain blackboard records solution elements for a domain problem. Its levels, solution intervals. and vocabulary are domain-specific and determined by the application system designer. The control blackboard records solution elements for the control problem: which potential action should the system execute on each problem-solving cycle? These solution elements are decisions about the system's own behavior. Different levels of the control blackboard represent: the Problem the system must solve. sequential problem-solving Strategies, local attentional Foci, general scheduling Policies, the To-Do-Set of feasible actions, and Chosen-Actions scheduled for execution. Its solution intervals distinguish different problem-solving time intervals.

The architecture also provides a vocabulary for control decisions. For example, the heuristic in Figure 1 favors actions whose knowledge sources have the KS-Type "Anchor" until every element at the Secondary-Anchor level has been anchored in at least one partial solution. As illustrated. a heuristic's Goal is a function that returns a numerical measure (O-100) of a potential action's desirability. The Criterion is a predicate that tests for the occurrence of Goal expiration conditions. The Rationale justifies the Goal. The Weight indicates its importance (O-10). The Status indicates whether the Goal is operative in affecting scheduling decisions. The Source and Creator are the triggering information and knowledge source that generated the **decision.** The First-Cycle and Last-Cycle are the first and last problem-solving cycles on which it is operative.

```
Description   "Favor Anchoring Actions"
                to at least one other one"
Goal          (If (Eq KS-Type 'Anchor) Then 100)
Criterion     (for Element in ($Levelnodes 'Secondary-Anchor) always
                ($Find-One ((Level-Is 'Secondary) (Copies Element))))
Rationale     "Develop a comprehensive set of partial solutions"
Weight                8
Status                'Operative
Source                Problem1
Creator               Anchor-First
First-Cycle 9
Last-Cycle 18
```

Figure 1. An Illustrative Control Heuristic

2. The blackboard control architecture defines explicit domain and control knowledge sources.

As in the standard blackboard architecture, independent condition-action knowledge sources generate solution elements during problem-solving and record them on the blackboard. Satisfying a knowledge source's condition produces a knowledge source activation record (KSAR), available for scheduling. Domain knowledge sources operate primarily on the domain blackboard. They are domain-specific and determined by the application system designer. Control knowledge sources operate primarily on the control blackboard. They interpret and modify representations of the system's own knowledge and behavior. Some control knowledge sources are domain-specific; others are domain-independent. All knowledge sources are represented as data structures that are, themselves, available for interpretation and modification. Their attributes are defined by the architecture.

3. A simple, adaptive scheduler chooses a KSAR to execute its action on each problem-solving cycle.

Three "basic" control knowledge sources iterate a three-step problem-solving cycle: (1) enumerate pending KSARs; (2) schedule one KSAR: (3) execute the action of the chosen KSAR. These knowledge sources have no control knowledge but simply adapt to dynamic solution state information recorded on the blackboard. They schedule whichever KSAR best satisfies currently operative control heuristics.

Systems developed in BB1 control their own behavior by dynamically generating, modifying, and

executing control plans on the control blackboard **while working to solve particular domain problems. They** explain their problem-solving **behavior by showing the relationships between individual** problem-solving actions or sequences of actions and the **dynamic control plan (see** [6]).

3. Learning Control Heuristics in BB1

BB1 learns new control heuristics through the actions of generic learning knowledge sources. Like other knowledge sources, they are triggered by blackboard events and generate KSARs that compete for scheduling priority. Their learning actions exploit knowledge of the BB1 environment, including: the structure, vocabulary, and semantics of the control blackboard, the current contents of the control blackboard, the structure and semantics of knowledge sources, the current repertoire of knowledge sources, and prototypical functions and forms for control heuristics.

Learning knowledge sources can learn new control heuristics, more general or more specific forms of known heuristics, or expansions or restrictions on the applicability of known heuristics. Some learning knowledge sources interact with domain experts. For example, the knowledge source Understand-Preference is triggered when a domain expert overrides BB1's scheduling recommendation. Its action determines the reason for the user's preference and encodes it as a new heuristic. Other learning knowledge sources operate autonomously. For example, the knowledge source Attribute-Results is triggered by dramatic improvement (or deterioration) in the current solution to the domain problem. Its action attributes the change in solution rating to preceding actions and encodes a heuristic favoring such actions. Evaluate-Heuristic, another autonomous knowledge source, is triggered when a new control knowledge source is executed. Its action evaluates subsequent changes in solution rating and adjusts the knowledge source's assumed utility and reliability accordingly.

4. Understand-Preference: An Illustration of Learning in BB1 /PROTEAN

This section presents a trace of the knowledge source Understand-Preference learning a new heuristic during a run of the PROTEAN system. BB1's menu-driven interface is self-explanatory. Aspects of the trace not directly related to Understand-Preference's behavior are discussed in [6].

The trace begins on problem-solving cycle 6. BB1 displays all invocable KSARs and recommends KSAR 6 for execution, recording the recommendation on the control blackboard in node Chosen-Action6. It explains its recommendation in terms of the current control plan, including scheduling and integration rules, operative control heuristics, associated ratings for KSAR 6. and KSAR 6's priority. It reports three other KSARs with the same priority. When the expert overrides BB1's recommendation, it modifies Chosen-Action6 accordingly, executes the preferred KSAR 7, and and reports the resulting blackboard event.

```
Cycle 6
To-Do-Set--T/I/A/N: I

Invocable KSARs:
KSAR 6 Anchor-Helix  triggered by (Supersede Secondary-Anchor4)
KSAR 7 Anchor-Helix  triggered by (Supersede Secondary-Anchor5)
[Other KSARs]

Display/Execute/Recommend/Charge-Ahead:  R

I have created event (Add Chosen-Action6)
I recommend KSAR 6
OK/Why/KSAR#: W

KSAR 6 Anchor-Helix  triggered by (Modify Secondary-Anchor4)
Control Plan:
  Scheduling Rule: Highest Priority KSAR
  Integration Rule: Sum of Weighted Ratings
  Strategy1  Anchor-Then-Refine
    Rationale: Develop a comprehensive set of partial solutions before
               deciding which ones to refine at the blob level
    Focus1 (Eq KS-TYpe 'Anchor) Weight 8 Rating 100
  Policy2   (Eq To-BB 'Control) Weight 10 Rating 0
Priority: 800
KSARs with the same Priority:  KSAR 7 KSAR 8 KSAR 9

OK/Why/KSAR#: 7

I have created the event (Supersede Chosen-Action6)
I am executing KSAR 7
I have created the event (Add Secondary1)
```

On cycle 7, BB1 displays the invocable KSARs and recommends and executes KSAR 15, Understand-Preference triggered by the expert's scheduling override on cycle 6.

```
Cycle 7
To-Do-Set--T/I/A/N:  I

KSAR 6 Anchor-Helix  triggered by (Supersede Secondary-Anchor4)
[Other KSARs]
KSAR 15 Understand-Preference  triggered by (Supersede Chosen-Action6)

Display/Execute/Recommend/Charge-Ahead:  R

I have created the event (Add Chosen-Action7)
I recommend KSAR 15
OK/Why/KSAR#:  OK

I am executing KSAR 15
```

Understand-Preference searches for the difference between KSAR 6 and KSAR 7 underlying the expert's preference for KSAR 7. Observing that both KSARs use the same knowledge source, Understand-Preference determines that the expert's preference for KSAR 7 lies in its triggering node, not its triggering event.

```
I would like to understand why you preferred KSAR 7 over KSAR 6.

Both KSARs involve the knowledge source Anchor-Helix.
Do you prefer KSAR 7 because of a difference in:
   Events or Nodes:  N
```

Understand-Preference reports all differences between the two nodes, using appropriate comparative terms for different data types, and determines that the user's preference reflects a difference in values for the attribute Constraints-To-Other-Structures.

```
KSAR 6's Node is Secondary-Anchor4
KSAR 7's Node is Secondary-Anchor5
Do you prefer KSAR 7 because Secondary-Anchor5 has:
(A) Type = Alpha-Helix.  rather than Beta-Sheet?
(B) a higher Number?
(C) a different Sequence?
(D) different External-Constraints?
(E) a higher Number-External-Constraints?
(F) different Constraints-To-Other-Structures?
(G) different Parameters?
Choose A/B/C/D/E:  F
```

Each node's Constraints-To-Other-Structures specifies a list of lists:

```
Secondary-Anchor4
  Constraints-To-Other-Structures
    ( (Secondary-Anchor5 1 ( 7) ) (Secondary-Anchor3 0 N i 1)
      (Secondary-Anchor2 0 Nil ) (Secondary-Anchor1 0 Nil ) )

Secondary-Anchor5
  Constraints-To-Other-Structures
    ((Secondary-Anchor4 1 (7)) (Secondary-Anchor3 0 Nil)
     (Secondary-Anchor2 2 (5 6)) (Secondary-Anchor1 1 (1)))
```

Therefore, Understand-Preference computes several statistics that summarize lists of lists and reports all statistics that distinguish the two values of Constraints-To-Other-Structures.

```
Do you prefer Secondary-Anchor5's Constraints-To-Other-Structures
  because it has:
(A) more non-zero element2s?
(B) a bigger range of element2s (O-2, rather than O-l)?
(C) more non-null element3s?
(D) a bigger range of lengths of element3s (O-2, rather than O-l)?
Choose A/B/C/D: A
```

Having determined that the user prefers KSAR 7 because more of its Constraints-To-Other-Structures have non-zero values in the second list position, Understand-Preference begins constructing a new control heuristic with the prototypical function:

```
(Prefermore (Variable Maxnumber)
   (Itimes Variable (Iquotient 100 Maxnumber))).
```

It determines the maximum value of the targetted statistic to provide a scaling parameter. It also obtains a rationale and weight for the heuristic.

```
What is the greatest possible number of non-zero element2s? 5
Why do you prefer more non-zero element2s? Can anchor more structures
How important is your preference (l-lo)? 8
```

Understand-Preference records the new heuristic at the Policy level of the control blackboard where it will influence all subsequent scheduling decisions. It also creates a new control knowledge source to generate the new heuristic during subsequent problem-solving sessions, It determines that the expert wishes to name the knowledge source Maximize-Constraint and obtains a brief description of it.

```
I am recording Policy3:
(Goal '(If (Focus-Level-Is Secondary)
         Then (Prefermore (for X in
              ($Value Focus-Node Constraints-To-Other-Structures)
              count (Greaterp (Elementno 3 X) 0)) 5)))
(Rationale "Can anchor more structures")
(Weight 8)

OK/No:OK

I am creating a new control knowledge source:
Trigger: (Event-Level-Is Problem)
         (Change-Type-Is Add)
         (Eq ($Value Event-Node Type) 'Protein-Analysis)
Pre-Condition: T
Action:
  Rule1
    LHS T
    RHS (Propose changetype Add level Policy attributes
          (Goal '(If (Event-Level-Is Secondary)
                   Then (Prefermore (for element2 in
                     ($Value Focus-Node Constraints-To-Other-Structures)
                     count (Greaterp (Elementno 3 X) 0)) 5))
          (Rationale "Can anchor more structures")
          (Weight 8)

OK/No: OK

What would you like to call this KS: Maximize-Constraint
Please describe it: This KS generates a Policy favoring actions that
operate on structures that have NOEs to many other structures.
```

5. Conclusions

BB1 ameliorates the knowledge acquisition bottleneck with knowledge sources that learn control heuristics. The illustrative knowledge source Understand-Preference relieves the domain expert by: (a) focusing on observed discrepancies between the expert's and the system's scheduling decisions: and (b) hierarchically searching the space of differences between their preferred actions. Understand-Preference eliminates the knowledge engineer entirely, assuming all responsibility for coding heuristics and knowledge sources. In principle, Understand-Preference could acquire all of the domain expert's heuristics. However, it currently learns only a subset of the several types of control heuristics understood by BB1. For example, it cannot learn strategic heuristics such as: First anchor pieces of secondary structure to one another in partial solutions until each piece of secondary structure is anchored to at least one partial solution: Then refine the largest. most constrained partial solutions. Research in progress aims to develop knowledge sources that learn other kinds of heuristics and knowledge sources that operate autonomously, without expert intervention.

BB1's learning knowledge sources derive their power from simple comparison operations performed on rich representations of problem-solving knowledge, actions, and results (see also [8]). These include syntactic representations of domain-specific information and semantic representations of control information.

References

[1] Barr, A., and Feigenbaum, E. A.
The handbook of artificial intelligence.
Los Altos. Ca.: William Kaufmann. Inc.. 1981.

[2] Buchanan, B., Jardetzky, O., Lichtarge, O., Hayes-Roth. B. Altman, R., and Hewett, M.
PROTEAN.
Technical Report, Stanford, Ca.: Stanford University, 1984.

[3] Erman. L.D.. Hayes-Roth, F., Lesser, V.R.. and Reddy, D.R.
The Hearsay-II speech-understanding system: Integrating knowledge to resoive uncertainty.
Computing Surveys 12:213-253, 1980.

[4] Goldin, S. E.. and Hayes-Roth. B.
Individual differences in planning processes.
Technical Report N-1488-ONR, Santa Monica, Ca.: Rand Corporation, 1980.

[5] Hayes-Roth, B.
Flexibility in executive processes.
Technical Report N-I 170-ONR, Santa Monica. Ca.: Rand Corporation. 1980.

[6] Hayes-Roth, B.
BB 1: An architecture for blackboard systems that control. explain, and /earn about their own behavior.
Technical Report HPP-84-16, Stanford, Ca.: Stanford University, 1984.

[7] Hayes-Roth, B.
A blackboard architecture for control.
Artificial Intelligence Journal in press, 1985.

[8] Lenat, D.B., and Brown, J.S.
Why AM and EURISKO appear to work.
Artifical Intelligence 23:269-294, *1964.*