Incremental Planning to Control a Blackboard-Based Problem Solver

E.H. Durfee and V.R. Lesser
University of Massachusetts
Amherst, MA 01003

Abstract

To control problem solving activity, a planner must resolve uncertainty about which specific long-term goals (solutions) to pursue and about which sequences of actions will best achieve those goals. In this paper, we describe a planner that abstracts the problem solving state to recognize possible competing and compatible solutions and to roughly predict the importance and expense of developing these solutions. With this information, the planner plans sequences of problem solving activities that most efficiently resolve its uncertainty about which of the possible solutions to work toward. The planner only details actions for the near future because the results of these actions will influence how (and whether) a plan should be pursued. As problem solving proceeds, the planner adds new details to the plan incrementally, and monitors and repairs the plan to insure it achieves its goals whenever possible. Through experiments, we illustrate how these new mechanisms significantly improve problem solving decisions and reduce overall computation. We briefly discuss our current research directions, including how these mechanisms can improve a problem solver's real-time response and can enhance cooperation in a distributed problem solving network.

1. Introduction

A problem solver's planning component must resolve control uncertainty stemming from two principal sources. As in typical planners, it must resolve uncertainty about which sequence of actions will satisfy its long-term goals. Moreover, whereas most planners are given a (possibly prioritized) set of well-defined long-term goals, a problem solver's planner must often resolve uncertainty about the goals to achieve. For example, an interpretation problem solver that integrates large amounts of data into "good" overall interpretations must use its data to determine what specific long-term goals (overall interpretations) it should pursue. Because the set of possible interpretations may be intractably large, the problem solver uses the data to form promising partial interpretations and then extends these to converge on likely complete interpretations. The blackboard-based problem solving architecture developed in Hearsay-II permits such *data-directed* problem solving [1].

In a purely data-directed problem solver, control decisions can be based only on the desirability of the expected immediate results of each action. The Hearsay-II system developed an algorithm for measuring desirability of actions to better focus problem solving [2]. Extensions to the blackboard architecture unify data-directed and goal-directed control by representing possible extensions and refinements to partial solutions as explicit goals [3]. Through goal processing and subgoals, sequences of related actions can be triggered to achieve important goals. Further modifications separate control knowledge and decisions from problem solving activities, permitting the choice of problem solving actions to be influenced by strategic considerations [4]. However, none of these approaches develop and use a high-level view of the current problem solving situation so that the problem solver can recognize and work toward more specific long-term goals.

In this paper, we introduce new mechanisms that allow a blackboard-based problem solver to form such a high-level view. By abstracting its state, the problem solver can recognize possible competing and compatible interpretations, and can use the abstract view of the data to roughly predict the importance and expense of developing potential partial solutions. These mechanisms are much more flexible and complex than those we previously developed [5] and allow the recognition of relationships between distant as well as nearby areas in the solution space. We also present new mechanisms that use the high-level view to form plans to achieve long-term goals. A plan represents specific actions for the near future and more general actions for the distant future. By forming detailed plans only for the near future, the problem solver does not waste time planning for situations that may never

This research was sponsored, in part, by the National Science Foundation under Grant MCS-8306327, by the National Science Foundation under Support and Maintenance Grant DCR-8318776, by the National Science Foundation under CER Grant DCR-8500332, and by the Defense Advanced Research Projects Agency (DOD), monitored by the Office of Naval Research under Contract NR049-041.

arise; by sketching out the entire plan, details for the near-term can be based on a long-term view. As problem solving proceeds, the plan must be monitored (and repaired when necessary), and new actions for the near future are added *incrementally*. Thus, plan formation, monitoring, modification, and execution are interleaved [6,7,8,9,10].

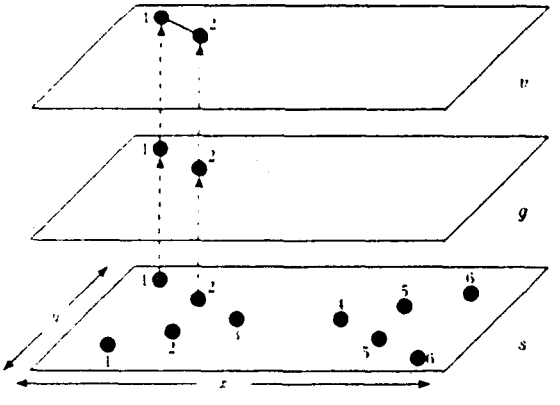
We have implemented and evaluated our new mechanisms in a vehicle monitoring problem solver, where they augment previously developed control mechanisms. In the next section, we briefly describe the vehicle monitoring problem solver. Section 3 provides details about how a high-level view is formed as an abstraction hierarchy. The representation of a plan and the techniques to form and dynamically modify plans are presented in Section 4. In Section 5, experimental results are discussed to illustrate the benefits and the costs of the new mechanisms. Finally, Section 6 recapitulates our approach and describes how the new mechanisms can improve real-time responsiveness and can lead to improved cooperation in a distributed problem solving network.

2. A Vehicle Monitoring Problem Solver

A vehicle monitoring problem solving *node*, as implemented in the Distributed Vehicle Monitoring Testbed (DVMT), applies simplified signal processing knowledge to acoustically sensed data in an attempt to identify, locate, and track patterns of vehicles moving through a two-dimensional space [11]. Each node has a blackboard-based problem solving architecture, with knowledge sources and levels of abstraction appropriate for vehicle monitoring. A *knowledge source* (KS) performs the basic problem solving tasks of extending and refining *hypotheses* (partial solutions). The architecture includes a goal blackboard and goal processing module, and through goal processing a node forms knowledge source instantiations (KSIs) that represent potential KS applications on specific hypotheses to satisfy certain goals. KSIs are prioritized based both on the estimated beliefs of the hypotheses each may produce and on the ratings of the goals each is expected to satisfy. The goal processing component also recognizes interactions between goals and adjusts their ratings appropriately; for example, subgoals of an important goal might have their ratings boosted. Goal processing can therefore alter KSI rankings to help focus the node's problem solving actions on achieving the subgoals of important goals [3].

A hypothesis is characterized by one or more *time-locations* (where the vehicle was at discrete sensed times), by an *event-class* (classifying the frequency or vehicle type), by a *belief* (the confidence in the accuracy of the hypothesis), and by a *blackboard-level* (depending on the amount of processing that has been done on the data). Synthesis KSs take one or more hypotheses at one blackboard-level and use event-class constraints to generate hypotheses at the next higher blackboard-level. Extension KSs take several hypotheses at a given blackboard-level and use constraints about allowable vehicle movements (maximum velocities and accelerations) to form hypotheses at the same blackboard-level that incorporate more time-locations.

For example, in Figure 1 each blackboard-level is represented as a surface with spatial dimensions x and y. At blackboard-level s (signal level) there are 10 hypotheses, each incorporating a single time-location (the time is indicated for each). Two of these hypotheses have been synthesized to blackboard-level g (group level). In turn, these hypotheses have been synthesized to blackboard-level v (vehicle level) where an extension KS has connected them into a single track hypothesis, indicated graphically by connecting the two locations. Problem solving proceeds from this point by having the goal processing component form goals (and subgoals) to extend this track to time 3 and instantiating KSIs to achieve these goals. The highest rated pending KSI is then invoked and triggers the appropriate KS to execute. New hypotheses are posted on the blackboard, causing further goal processing and the cycle repeats until an acceptable track incorporating data at each time is created. One of the potential solutions is indicated at blackboard-level v in Figure 1.



Blackboard-levels are represented as surfaces containing hypotheses (with associated sensed times). Hypotheses at higher blackboard-levels are synthesized from lower level data, and a potential solution is illustrated with a dotted track at blackboard-level v.

Figure 1: An Example Problem Solving State.

3. A High-level View for Planning and Control

Planning about how to solve a problem often requires viewing the problem from a different perspective. For example, a chemist generally develops a plan for deriving a new compound not by entering a laboratory and envisioning possible sequences of actions but by representing the problem with symbols and using these symbols to hypothesize possible derivation paths. By transforming the problem into this representation, the chemist can more easily sketch out possible solutions and spot reactions that lead nowhere, thereby improving the decisions about the actions to take in the laboratory.

A blackboard-based, vehicle monitoring problem solver requires the same capabilities. Transforming the node's problem solving state into a suitable representation for planning requires domain knowledge to recognize relationships—in particular, long-term relationships—in the data. This transformation is accomplished by incrementally clustering data into increasingly abstract groups based on the attributes of the data: the hypotheses can be clustered based on one attribute, the resulting clusters can be further clustered based on another attribute, and so on. The transformed representation is thus a hierarchy of clusters where higher-level clusters abstract the information of lower-level clusters. More or less detailed views of the problem solving situation are found by accessing the appropriate level of this abstraction hierarchy, and clusters at the same level are linked by their relationships (such as having adjacent time frames or blackboard-levels, or corresponding to nearby spatial regions).

We have implemented a set of knowledge-based clustering mechanisms for vehicle monitoring, each of which takes clusters at one level as input and forms output clusters at a new level. Each mechanism uses different domain-dependent relationships, including:

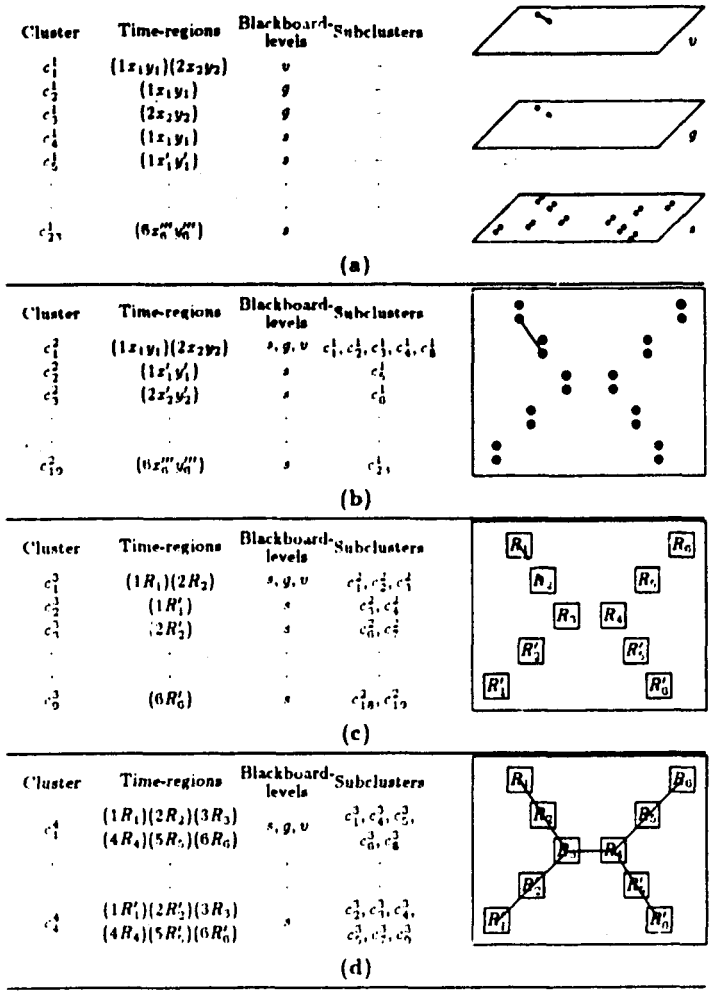
- **temporal relationships:** the output cluster combines any input clusters that represent data in adjacent time frames and that are spatially near enough to satisfy simple constraints about how far a vehicle can travel in one time unit.
- **spatial relationships:** the output cluster combines any input clusters that represent data for the same time frames and that are spatially near enough to represent sensor noise around a single vehicle.
- **blackboard-level relationships:** the output cluster combines any input clusters that represent the same data at different blackboard-levels.
- **event-class relationships:** the output cluster combines any input clusters that represent data corresponding to the same event-class (type of vehicle).
- **belief relationships:** the output cluster combines any input clusters that represent data with similar beliefs.

The abstraction hierarchy is formed by sequentially applying the clustering mechanisms. The order of application depends on the *bias* of the problem solver: since the order of clustering affects which relationships are most emphasized at the highest levels of the abstraction hierarchy, the problem solver should cluster to emphasize the relationships it expects to most significantly influence its control decisions. Issues in representing bias and modifying inappropriate bias are discussed elsewhere [12].

To illustrate clustering, consider the clustering sequence in Figure 2, which has been simplified by ignoring many cluster attributes such as event-classes, beliefs, volume of data, and amount of pending work; only a cluster's blackboard-levels (a cluster can incorporate more than one) and its time-regions (indicating a region rather than a specific location for a certain time) are discussed. Initially, the problem solving state is nearly identical to that in Figure 1, except that for each hypothesis in Figure 1 there are now two hypotheses at the same sensed time and slightly different locations. In Figure 2a, each cluster c_n^l (where l is the level in the abstraction hierarchy) corresponds to a single hypothesis, and the graphical representation of the clusters mirrors a representation of the hypotheses. By clustering based on blackboard-level, a second level of the abstraction hierarchy is formed with 19 clusters (Figure 2b). As is shown graphically, this clustering "collapses" the blackboard by combining clusters at the previous abstraction level that correspond to the same data at different blackboard-levels. In Figure 2c, clustering by spatial relationships forms 9 clusters. Clusters at the second abstraction level whose regions were close spatially for a given sensed time are combined into a single cluster. Finally, clustering by temporal relationships in Figure 2d combines any clusters at the third abstraction level that correspond to adjacent sensed times and whose regions satisfy weak vehicle velocity constraints.

The highest level clusters, as illustrated in Figure 2d, indicate four rough estimates about potential solutions: a vehicle moving through regions $R_1R_2R_3R_4R_5R_6$, through $R_1R_2R_3R_4R_5'R_6'$, through $R_1'R_2'R_3R_4R_5R_6$, or through $R_1'R_2'R_3R_4R_5'R_6'$. The problem solver could use this view to improve its control decisions about what short-term actions to pursue. For example, this view allows the problem solver to recognize that all potential solutions pass through R_3 at sensed time 3 and R_4 at sensed time 4. By boosting the ratings of KSIs in these regions, the problem solver can focus on building high-level results that are most likely to be part of any eventual solution.

In some respects, the formation of the abstraction hierarchy is akin to a rough pass at solving the problem, as indeed it must be if it is to indicate where the possible solutions may lie. However, abstraction differs from problem solving because it ignores many important constraints needed to solve the problem. Forming the abstraction hierarchy is thus much less computationally expensive than problem solving, and results in a representation that is too inexact as a problem solution but is suitable for control. For



A sequence of clustering steps are illustrated both with tables (left) and graphically (right). c_i^l represents cluster i at level l of the abstraction hierarchy. In (a), each cluster is a hypothesis. These are clustered by blackboard-level to get (b); note that graphically the levels have been collapsed into one. These clusters are then grouped by spatial relationships to form (c), which in turn is clustered by temporal relationships to form (d).

Figure 2: An Example of Incremental Clustering.

example, although the high-level clusters in Figure 2d indicate that there are four potential solutions, three of these are actually impossible based on the more stringent constraints applied by the KSs. The high-level view afforded by the abstraction hierarchy therefore does not provide answers but only rough indications about the long-term promise of various areas of the solution space, and this additional knowledge can be employed by the problem solver to make better control decisions as it chooses its next task.

4. . Incremental Planning

The *planner* further improves control decisions by intelligently ordering the problem solving actions. Even with the high-level view, uncertainty remains about whether each long-term goal can actually be achieved, about whether an action that might contribute to achieving a long-term goal will actually do so (since long-term goals are inexact), and about how to most economically form a desired result (since the same result can often be derived in different ways). The planner reduces control uncertainty in two ways. First, it orders the intermediate goals for achieving long-term goals so that the results of working on earlier intermediate goals can diminish the uncertainty about how (and whether) to work on later intermediate goals. Second, the planner forms a detailed sequence of steps to achieve the next intermediate goal: it determines the least costly way to form a result to satisfy the goal. The planner thus sketches out long-term intentions as sequences of intermediate goals, and forms detailed plans about the best way to achieve the next intermediate goal.

A long-term vehicle monitoring goal to generate a track consisting of several time-locations can be reduced into a series of intermediate goals, where each intermediate goal represents a desire to extend the track satisfying the previous intermediate goal into a new time-location.[1] To determine an order for pursuing the possible intermediate goals, the planner currently uses three domain-independent heuristics:

Heuristic-1 *Prefer common intermediate goals.* Some intermediate goals may be common to several long-term goals. If uncertain about which of these long-term goals to pursue, the planner can postpone its decision by working on common intermediate goals and then can use these results to better distinguish between the long-term goals. This heuristic is a variation of least-commitment [13].

Heuristic-2 *Prefer less costly intermediate goals.* Some intermediate goals may be more costly to achieve than others. The planner can quickly estimate the relative costs of developing results in different areas by comparing their corresponding clusters at a high level of the abstraction hierarchy: the number of event-classes and the spatial range of the data in a cluster roughly indicates how many potentially competing hypotheses might have to be produced. This heuristic causes the planner to develop results more quickly. If these results are creditable they provide predictive information, otherwise the planner can abandon the plan after a minimum of effort.

Heuristic-3 *Prefer discriminative intermediate goals.* When the planner must discriminate between possible long-term goals, it should prefer to work on intermediate goals that most effectively indicate the relative promise of each long-term goal. When no common intermediate goals remain, therefore, this heuristic triggers work in the areas where the long-term goals differ most.

These heuristics are interdependent. For example, common intermediate goals may also be more costly, as in one of the experiments described in the next section. The relative influence of each heuristic can be modified parametrically.

Having identified a sequence of intermediate goals to achieve one or more long-term goals, the planner can reduce its uncertainty about how to satisfy these intermediate goals by planning in more detail. If the planner possesses models of the KSs that roughly indicate both the costs of a particular action and the general characteristics of the output of that action (based on the characteristics of the input), then the planner can search for the best of the alternative ways to satisfy an intermediate goal. We have provided the planner for our vehicle monitoring problem solver with coarse KS models that allow it to make reasonable predictions about short sequences of actions to find the sequences that best achieve intermediate goals.[2] To reduce the effort spent on planning, the planner only forms detailed plans for the next intermediate goal: since the results of earlier intermediate goals influence decisions about how and whether to pursue subsequent intermediate goals, the planner avoids expending effort forming detailed plans that may never be used.

Given the abstraction hierarchy in Figure 2, the planner recognizes that achieving each of the four long-term goals (Figure 2d) entails intermediate goals of tracking the vehicle through these regions. Influenced predominantly by Heuristic-1, the planner decides to initially work toward all four long-term goals at the same time by achieving their common intermediate goals. A detailed sequence of actions to drive the data in R_3 at level s to level v is then formulated. The planner creates a plan whose attributes (and their values in this example) are:

- the long-term goals the plan contributes to achieving (in the example, there are four);
- the predicted, underspecified time-regions of the eventual solution (in the example, the time regions are $(1\ \ R_1 or R_1')(2\ \ R_2 or R_2')(3\ \ R_3)\ \ldots\ )$;
- the predicted vehicle type(s) of the eventual solution (in the example, there is only one type of vehicle considered);
- the order of intermediate goals (in the example, begin with sensed time 3, then time 4, and then work both backward to earlier times and forward to later times);
- the blackboard-level for tracking, depending on the available knowledge sources (in the example, this is level v);
- a record of past actions, updated as actions are taken (initially empty);
- a sequence of the specific actions to take in the short-term (in the example, the detailed plan is to drive data in region R_3 at level s to level v);
- a rating based on the number of long-term goals being worked on, the effort already invested in the plan, the average ratings of the KSIs corresponding to the detailed short-term actions, the average belief of the partial solutions previously formed by the plan, and the predicted beliefs of the partial solutions to be formed by the detailed activities.

[1] In general terms, an intermediate goal in any interpretation task is to process a new piece of information and to integrate it into the current partial interpretation.

[2] If the predicted cost of satisfying an intermediate goal deviates substantially from the crude estimate based on the abstract view, the ordering of the intermediate goals may need to be revised.

As each predicted action is consecutively pursued, the record of past actions is updated and the actual results of the action are compared with the general characteristics predicted by the planner. When these agree, the next action in the detailed short-term sequence is performed if there is one, otherwise the planner develops another detailed sequence for the next intermediate goal. In our example, after forming results in R_3 at a high blackboard-level, the planner forms a sequence of actions to do the same in R_4. When the actual and predicted results disagree (since the planner's models of the KSs may be inaccurate), the planner must modify the plan by introducing additional actions that can get the plan back on track. If no such actions exist, the plan is aborted and the next highest rated plan is pursued. If the planner exhausts its plans before forming a complete solution, it reforms the abstraction hierarchy (incorporating new information and/or clustering to stress different problem attributes) and attempts to find new plans. Throughout this paper, we assume for simplicity that no important new information arrives after the abstraction hierarchy is formed; when part of a more dynamic environment, the node will update its abstraction hierarchy and plans whenever such information becomes available.

The planner thus generates, monitors, and revises plans, and interleaves these activities with plan execution. In our example, the common intermediate goals are eventually satisfied and a separate plan must be formed for each of the alternative ways to proceed. After finding a partial track combining data from sensed times 3 and 4, the planner decides to extend this track backward to sensed time 2. The long-term goals indicate that work should be done in either R_2 or R'_2. A plan is generated for each of the two possibilities, and the more highly rated of these plans is followed. Note, however, that the partial track already developed can provide predictive information that, through goal processing, can increase the rating of work in one of these regions and not the other. In this case, constraints that limit a vehicle's turning rate are used when goal processing (subgoaling) to increase the ratings of KSI's in R'_2, thus making the plan to work there next more highly rated.[3]

The planner and goal processing thus work in tandem to improve problem solving performance. The goal processing uses a detailed view of local interactions between hypotheses, goals, and KSIs to differentiate between alternative actions. Goal processing can be computationally wasteful, however, when it is invoked based on strictly local criteria. Without the knowledge of long-term reasons for building a hypothesis, the problem solver simply forms goals to extend and refine the hypothesis in all possible ways. These goals are further processed (subgoaled) if they are at certain blackboard-levels, again regardless of any long-term justification for doing so. With its long-term view, the planner can drastically reduce the amount of goal processing. As it pursues, monitors, and repairs plans, the planner identifies areas where goals and subgoals could improve its decisions and selectively invokes goal processing to form only those goals that it needs. As the experimental results in the next section indicate, providing the planner with the ability to control goal processing can dramatically reduce control overhead.

In summary, we have developed mechanisms that permit incremental planning of problem solving activities in a blackboard-based problem solver. These mechanisms interleave planning and execution, monitoring plans and replanning when necessary. We base these mechanisms on having a high-level, long-term view of problem solving and on having acceptable models of problem solving actions. Furthermore, note that incremental planning may be inappropriate in domains where details about actions in the distant future can highly constrain the options in the near future. In these domains, constraints must be used to detail an entire plan before acting [3]. However, in unpredictable domains, incremental planning, plan monitoring, and plan repair are crucial to effective control since plans about the near future cannot depend on future states that may never arrive.

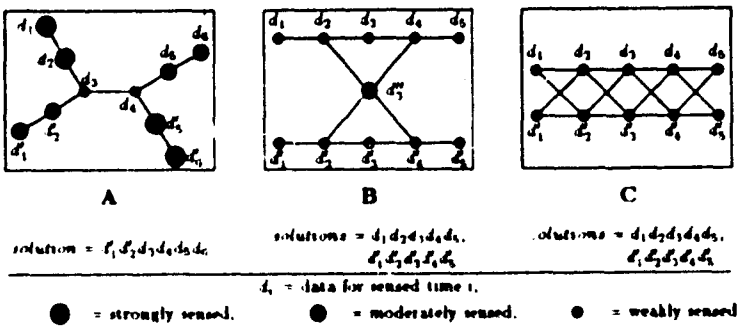
5. Experiments in Incremental Planning

We illustrate the advantages and the costs of our planner in several problem solving situations, shown in Figure 3. Situation A is the same as in Figure 2 except that each region only has one hypothesis. Also note that the data in the common regions is most weakly sensed. In situation B, no areas are common to all possible solutions, and issues in plan monitoring and repair are therefore stressed. Finally, situation C has many potential solutions, where each appears equally likely from a high-level view.

When evaluating the new mechanisms, we consider two important factors: how well do they improve control decisions (reduce the number of incorrect decisions), and how much additional overhead do they introduce to achieve this improvement. Since each control decision causes the invocation of a KSI, the first factor is measured by counting KSIs invoked—the fewer the KSIs, the better the control decisions. The second factor is measured as the actual computation time (runtime) required by a node to solve a problem, representing the combined costs of problem solving and control computation.

The experimental results are summarized in Table 1. To determine the effects of the new mechanisms, each problem situation was solved both with and without them, and for each case the number of KSIs and the computation time were measured. We also measured the number of goals generated during problem solving to illustrate how control overhead can be reduced by having the planner control the goal processing.

[3] In fact the turn to R_3 exceeds these constraints, as does the turn to R'_3, so that the only track that satisfies the constraints is $R'_1R'_2R_3R_4R_5R_6$.



Three problem solving situations are displayed. The possible tracks (found in the abstraction hierarchy) are indicated by connecting the related data points, and the acceptable solution(s) for each situation are given.

Figure 3: The Experimental Problem Situations.

Experiments E1 and E2 illustrate how the new mechanisms can dramatically reduce both the number of KSIs invoked and the computation time needed to solve the problem in situation A. Without these mechanisms (E1), the problem solver begins with the most highly sensed data (d_1, d_2, d'_3, and d'_4). This incorrect data actually corresponds to *noise* and may have been formed due to sensor errors or echoes in the sensed area. The problem solver attempts to combine this data through d_3 and d_4 but fails because of turning constraints, and then it uses the results from d_3 and d_4 to eventually work its way back out to the moderately sensed correct data. With the new mechanisms (E2), problem solving begins at d_3 and d_4 and, because the track formed ($d_3 d_4$) triggers goal processing to stimulate work on the moderate data, the solution is found much more quickly (in fact, in *optimal* time 14). The planner controls goal processing to generate and process only those goals that further the plan; if goal processing is done independently of the planner (E3), the overhead of the planner coupled with the only slightly diminished goal processing overhead (the number of goals is only modestly reduced, comparing E3 with E1) nullifies the computation time saved on actual problem solving. Moreover, because earlier, less constrained goals are subgoaled, control decisions deteriorate and more KSIs must be invoked.

The improvements in experiment E2 were due to the initial work done in the common areas d_3 and d_4 triggered by Heuristic-1. Situation A' is identical to situation A except that areas d_3 and d_4 contain numerous competing hypotheses. If the planner initially works in those areas (E5), then many KSIs are required to develop all of these hypotheses—fewer KSIs are invoked without planning at all (E4). However, by estimating the relative costs of the alternative intermediate goals, the planner can determine that d_3 and d_4, although twice as common as the other areas, are likely to be more than twice as costly to work on. Heuristic-2 overrides Heuristic-1, and a plan is formed to develop the other areas first and then use these results to more tightly control processing in d_3 and d_4. The number of KSIs and the computation time are thus reduced (E6).

Expt	Situ	Plan?	KSIs	Rtime	Goals	Comments
E1	A	no	58	17.2	262	-
E2	A	yes	24	8.1	49	-
E3	A	yes	32	19.4	203	independent goal processing and planning
E4	A'	no	58	19.9	284	noise in d_3, d_4
E5	A'	yes	64	17.3	112	noise in d_3, d_4, Heuristic-1 predominant
E6	A'	yes	18	16.5	71	noise in d_3, d_4, Heuristic-2 predominant
E7	B	no	73	21.4	371	-
E8	B	yes	45	11.8	60	-
E9	B	yes	45	20.6	257	independent goal processing and planning
E10	C	no	85	29.8	465	-
E11	C	yes	44	19.3	75	-

Legend

Situ: The problem situation.
Plan?: Are the new planning mechanisms used?
KSIs: Number of KSIs invoked to find solution.
Rtime: The total runtime (computation time) to find solution (*in minutes*).
Goals: The number of goals formed and processed.
Comments: Additional aspects of the experiment.

Table 1: A Summary of the Experimental Results.

In situation B, two solutions must be found, corresponding to two vehicles moving in parallel. Without the planner (E7), problem solving begins with the most strongly sensed data (the noise in the center of the area) and works outward from there. Only after many incorrect decisions to form short tracks that cannot be incorporated into longer solutions does the problem solver generate the two solutions. The high-level view of this situation, as provided by the abstraction hierarchy, allows the planner in experiment E8 to recognize six possible alternative solutions, four of which pass through d_3'' (the most common area). The planner initially forms $plan_1$, $plan_2$, and $plan_3$, beginning in d_3'', d_3, and d_3' respectively (Heuristic-1 triggers the preference for d_3'', and subsequently Heuristic-3 indicates a preference for d_3 and d_3'). Since it covers the most long-term goals, $plan_1$ is pursued first—a reasonable strategy because effort is expended on the solution path if the plan succeeds, and if the plan fails then the largest possible number of candidate solutions are eliminated. After developing d_3'', $plan_1$ is divided into two plans to combine this data with either d_2 or d_2'. One of these equally rated plans is chosen arbitrarily and forms the track d_2d_3'', which then must be combined with d_1. However, because of vehicle turning constraints, only d_1d_2 rather than $d_1d_2d_3''$ is formed. The plan monitor flags an error, an attempt to repair the plan fails, and the plan aborts. Similarly, the plan to form $d_1'd_2'd_3''$ eventually aborts. $Plan_2$ is then invoked, and after developing d_3 it finds that d_2 has already been developed (by the first aborted plan). However, the plan monitor detects that the predicted result, d_2d_3 was not formed, and the plan is repaired by inserting a new action that takes advantage of the previous formation of d_1d_2 to generate $d_1d_2d_3$. The predictions are then more than satisfied, and the plan continues until a solution is formed. The plan to form the other solution is similarly successfully completed. Finally, note once again that, if the planner does not control goal processing (E9), unnecessary overhead costs are incurred, although this time the control decisions (KSIs) are not degraded.

Situation C also represents two vehicles moving in parallel, but this time they are closer and the data points are all equally well sensed. Without the new mechanisms (E10), control decisions in this situation have little to go on: from a local perspective, one area looks as good as another. The problem solver thus develops the data points in parallel, then forms all tracks between pairs of points, then combines these into larger tracks, until finally it forms the two solution tracks. The planner uses the possible solutions from the abstraction hierarchy to focus on generating longer tracks sooner, and by monitoring its actions to extend its tracks, the planner more quickly recognizes failed extensions and redirects processing toward more promising extensions. The new mechanisms thus improve control decisions (reduce the KSIs) without adding excessive computational overhead (E11). However, the planner must consider 32 possible solutions in this case and does incur significant overhead. For complex situations, additional control mechanisms may be needed by the planner to more flexibly manage the large numbers of possibilities.

6. The Implications of Abstraction and Planning

We have described and evaluated mechanisms for improving control decisions in a blackboard-based vehicle monitoring problem solver. Our approach is to develop an abstract view of the current problem solving situation and to use this view to better predict both the long-term significance and cost of alternative actions. By recognizing and planning to achieve long-term goals, problem solving is more focused. By using the abstraction hierarchy when making planning decisions, problem solving can be more cost effective. Finally, by interleaving plan generation, monitoring, and repair with plan execution, the mechanisms lead to more versatile planning, where actions to achieve the system's (problem solving) goals and actions to satisfy the planner's needs (resolve its own uncertainty) are integrated into a single plan.

This approach can be generally applied to blackboard-based problem solvers. Abstraction requires exploiting relationships in the data—relationships that are used by the knowledge sources as well—such as allowable combinations of speech sounds [1] or how various errands are related spatially or temporally [4].[4] Planning requires simple models of KSs, recognition of intermediate goals (to extend a phrase in speech, to add another errand to a plan), and heuristics to order the intermediate goals. We believe that many if not all blackboard-based problem solvers (and more generally, problem solvers whose long-term goals depend on their current situation) could incorporate similar abstraction and planning mechanisms to improve their control decisions.

The benefits of this approach extend beyond the examples demonstrated in this paper. For example, goal satisfaction and problem solving termination are important issues in blackboard-based problem solvers. Given its underspecified goals of forming "good" solutions with its input, how does the problem solver recognize when it has found such solutions or when it can improve a solution? The more global view of the problem provided by the abstraction hierarchy helps the problem solver discover areas where improvements are possible and potentially worthwhile. The enumeration of possible solutions, and the success or failure to achieve them, similarly improves the problem solver's ability to determine when a solution is the best of the possible alternatives.

These mechanisms also help a problem solver to make informed decisions about how best to solve a problem under real-time constraints. The KS models provide estimates of the cost (in time) of possible activities so that the amount of time to achieve the next intermediate goal can be predicted. By exploiting the similarities between intermediate goals, moreover, these predictions can be generalized over all intermediate goals (making allowances for more or less costly areas as indicated by the abstraction hierarchy) and the time needs for the entire plan can be predicted. With this prediction, the planner can modify the plan (eliminate actions that unnecessarily increase the belief in a hypothesis, replace expensive actions with actions that inexpensively achieve less exact results) until the predicted time costs satisfy the time constraints.

[4] In fact, the WORD-SEQ knowledge source in the Hearsay-II speech understanding system essentially is a clustering mechanism: by applying weak grammatical constraints about pairwise sequences of words, WORD-SEQ generated approximate word sequences solely to control the application of the more expensive PARSE KS that applied full grammatical constraints about sequences of arbitrary length [1].

Finally, planning and prediction are vital to cooperation among problem solvers. A network of such problem solvers that are cooperatively solving a single problem could communicate about their plans, indicating what partial solutions they expect to generate and when. With this information, each problem solver can coordinate its activities with the others to generate and exchange useful results more efficiently, thereby improving network problem solving performance [12,14,5]. In essence, the problem solvers together form a distributed plan. The use of incremental planning, plan monitoring, and plan repair is particularly appropriate in such domains due to the inherent unpredictability of future actions and interactions.

The mechanisms we have outlined in this paper provide the basis for these possibilities. We are currently augmenting the mechanisms with capabilities to perform in more complex, dynamic environments: to modify the abstraction hierarchy when important unexpected situations arise and to model and plan for potential future situations (the arrival of more data to be processed, the actions of other problem solvers). Our new mechanisms, though they address issues previously neglected, should be integrated with other control techniques to be fully flexible, as seen in experiment E11. The combination of our mechanisms and goal processing has proved fruitful, and we believe that our mechanisms could similarly benefit by being integrated with other control approaches such as a blackboard architecture for control [4]. Based on the results we have outlined in this paper, we anticipate that the further development of mechanisms for developing abstract views and for incremental planning to control blackboard-based problem solvers will greatly enhance the performance of these problem solving systems, will lead to improved real-time response and to better coordination in distributed problem solving networks, and will increase our understanding of planning and action in highly uncertain domains.

References

[1] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II speech understanding system: integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, June 1980.

[2] Frederick Hayes-Roth and Victor R. Lesser. Focus of attention in the Hearsay-II speech understanding system. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 27–35, August 1977.

[3] Daniel D. Corkill, Victor R. Lesser, and Eva Hudlicka. Unifying data-directed and goal-directed control: an example and experiments. In *Proceedings of the Second National Conference on Artificial Intelligence*, pages 143–147, August 1982.

[4] Barbara Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26:251–321, 1985.

[5] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Increasing coherence in a distributed problem solving network. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 1025–1030, August 1985.

[6] R. T. Chien and S. Weissman. Planning and execution in incompletely specified environments. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pages 169–174, August 1975.

[7] Randall Davis. *A model for planning in a multi-agent environment: steps toward principles of teamwork.* Technical Report MIT AI Working Paper 217, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, Massachusetts, June 1981.

[8] Jerome A. Feldman and Robert F. Sproull. Decision theory and artificial intelligence II: the hungry monkey. *Cognitive Science*, 1:158–192, 1977.

[9] Gordon I. McCalla, Larry Reid, and Peter F. Schneider. Plan creation, plan execution, and knowledge acquisition in a dynamic microworld. *International Journal of Man-Machine Studies*, 16:89–112, 1982.

[10] Earl D. Sacerdoti. Problem solving tactics. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pages 1077–1085, August 1979.

[11] Victor R. Lesser and Daniel D. Corkill. The distributed vehicle monitoring testbed: a tool for investigating distributed problem solving networks. *AI Magazine*, 4(3):15–33, Fall 1983.

[12] Edmund H. Durfee. *An Approach to Cooperation: Planning and Communication in a Distributed Problem Solving Network.* Technical Report 86-09, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, March 1986.

[13] Mark Stefik. Planning with constraints. *Artificial Intelligence*, 16:111–140, 1981.

[14] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. *Coherent Cooperation Among Communicating Problem Solvers.* Technical Report 85-15, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, April 1985. Also to appear in *IEEE Transactions on Computers.*