Incremental Planning to Control a Time-constrained, Blackboard-based Problem Solver

Edmund H. Durfee and Victor R. Lesser

COINS Technical Report 87-07

February 1987

Abstract

A problem solver's control component must resolve uncertainty about which possible solutions to pursue and about what actions will best lead to the desired solutions. In this paper, we describe a planner that improves the control decisions made by a blackboard-based problem solver. This planner abstracts the problem solving state to recognize possible competing and compatible solutions and roughly predicts 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. The planner also attempts to meet time constraints by using its predictions about the time needs of its plans to recognize and revise plans that will exceed problem solving deadlines. The planner modifies plans to more quickly generate inferior but still acceptable solutions whenever possible. By predicting whether plans to generate alternative solutions are probably worth pursuing (given enough time) the planner also decides when a satisfactory solution has been found and problem solving should be terminated. We describe how these mechanisms have been implemented as part of a blackboard-based problem solver, and we discuss how they improve problem solving decisions, reduce overall computation, and allow the problem solver to meet deadlines and to decide when it has done enough work on a problem.

This technical report subsumes the original technical report with this number that was entitled "Planning to meet deadlines in a blackboard-based problem solver."

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. Edmund Durfee was also supported by an IBM Graduate Fellowship.

1. Introduction

A problem solver begins with knowledge about the problem domain and data about the current situation, and constructs solutions by applying knowledge to data. The problem solver's *control* component must decide what actions the problem solver will take, where each action is to apply some knowledge to certain data. This component must not only resolve uncertainty about what actions will lead to desirable solutions, but also about what solutions will be desirable. The resolution of this uncertainty is particularly difficult in problem domains where the space of possible solutions (the search space) is very large, the knowledge available to the problem solver is diverse and computationally expensive to apply, and the data is extensive and potentially errorful.

To improve control decisions in such domains, we view control as an incremental planning process. In this paper, we explore how a planning component for control can be incorporated into a blackboard-based problem solver. Like most planners, a problem solver's planner must resolve uncertainty about which sequence of actions will satisfy its long-term goals, where its long-term goals are possible solutions to construct. 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, a blackboard-based problem solver typically performs *data-directed* problem solving: it incrementally constructs larger, more encompassing partial solutions until a complete solution is reached [9]. Because the problem solver explicitly represents complete solutions only *after* it has solved the problem, the problem solver's planner must generate its own representation that allows it to recognize potential solutions in advance.

When unable to recognize potential solutions in advance, a data-directed, blackboard-based problem solver bases its control decisions 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 [12]. Extensions to the blackboard architecture unify data-directed and goal-directed control by representing possible extensions and refinements to partial solutions as explicit goals [2]. 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 [11]. However, none of these approaches develop and use a high-level view of the current problem solving situation to recognize potential solutions and relationships between these solutions. By specifying the characteristics of likely solutions (defining more specific long-term goals) in advance, the problem solver plans actions to selectively processes only its most useful low-level data as it efficiently navigates through a large solution space to form complete solutions.

In this paper, we introduce new planning 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 solutions, 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 [8] 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 for achieving 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 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 [1,4,10,16,18].

In addition, we describe mechanisms that make predictions about how long plans will take and about the expected quality of their different results so that the problem solver can reason about any deadlines it faces. Someone or something usually has a specific, time-dependent use for the solution, and the solution is useless beyond some deadline. Moreover, a deadline is often combined with preferences for solution time, such as "get a good solution as soon as possible and no later than time t." With this type of deadline, the problem solver faces difficult decision about when to *terminate* problem solving: if it has found a solution with time to spare but a better solution might be found with additional

work, then it must balance the conflicting goals of returning a solution as quickly as possible and returning a good solution. By terminating problem solving too early it might miss an important solution, but by terminating too late it might waste valuable time, so it is important to make termination decisions wisely [12,20]. A problem solver can reduce its uncertainty about whether to terminate by predicting whether any pending plan is likely to generate a solution better than those already found. It can also recognize when following a plan might cause it to exceed some deadline, so that either it should modify the plan to take less time (by forming an inferior but still acceptable result) or it should prefer an alternative plan that needs less time.

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. Techniques for making predictions about plans are described in Section 5, along with ways of using the predictions to modify plans to meet deadlines. Section 6 summarizes several experiments that illustrate the benefits and costs of the mechanisms, including how they improve problem solving decisions, reduce overall computation, and allow the problem solver to meet deadlines and to make informed termination decisions. Finally, Section 7 recapitulates our new mechanisms and indicates directions for future research.

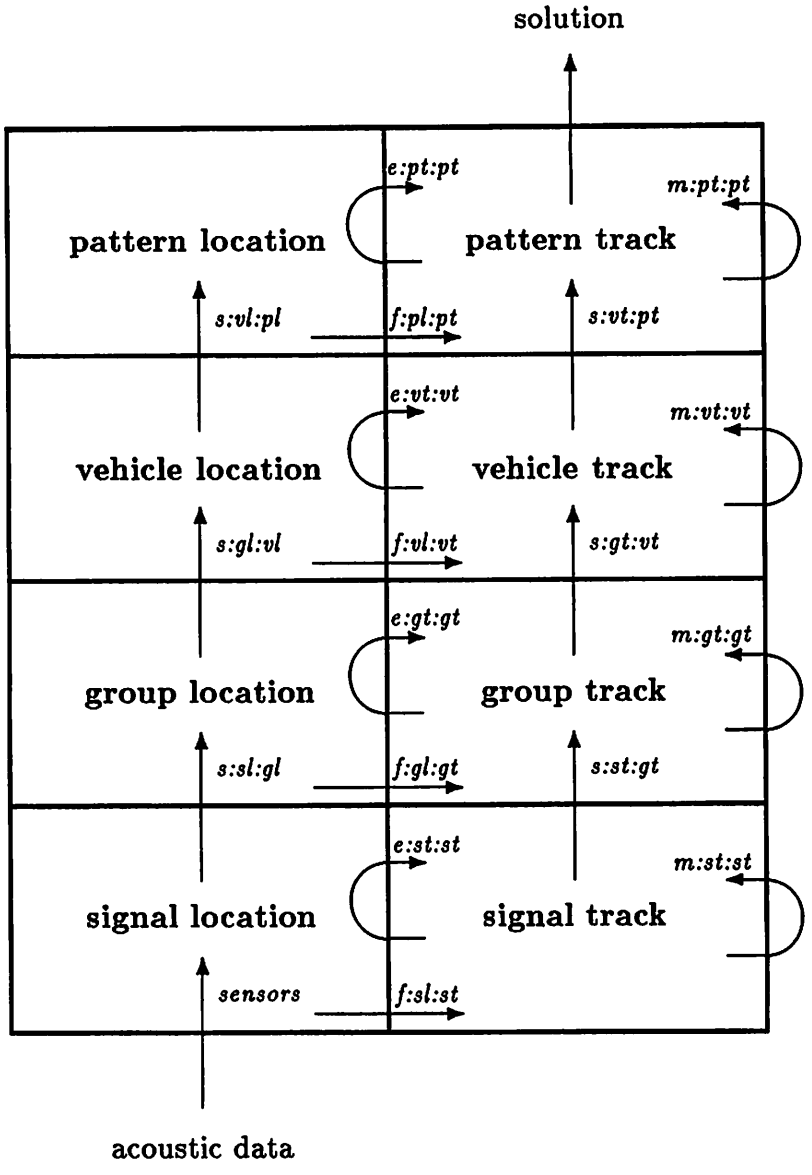
2. A Vehicle Monitoring Problem Solver

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 [14]. The vehicles' characteristic acoustic signals are detected at discrete time intervals, and these signals indicate the types of vehicles passing through the area and their approximate locations at each sensed time. An acoustic sensor's range and accuracy are limited, and the raw data it generates can be errorful, causing non-existent (ghost) vehicles to be "identified" and causing actual vehicles to be

located incorrectly, misidentified, or missed completely. A vehicle monitoring node applies signal processing knowledge to correlate the data, attempting to recognize and eliminate incorrect noisy sensor data as it integrates the correct data into solution tracks. 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).

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*. The hypotheses are organized on a blackboard with four blackboard-levels: **signal** (for low-level analyses of the sensory data), **group** (for collections of harmonically related signals), **vehicle** (for collections of groups that correspond to given vehicle types), and **pattern** (for collections of spatially related vehicle types such as vehicles moving in a formation). Each of these blackboard-levels is split into a blackboard-level for location hypotheses (which have one time-location) and a blackboard-level for track hypotheses (which have a sequence of time-locations). In total, nodes have eight blackboard-levels with appropriate KSs for combining hypotheses on one level to generate more encompassing hypotheses on the same or on a higher level (Figure 2). A synthesis KS uses knowledge about how different distributions of frequencies are indicative of certain types of vehicles: the KS finds combinations of compatible data and filters out noisy data to identify likely vehicles. Formation, extension, and merge KSs takes several hypotheses that each indicate where a particular type of vehicle may have been at some sequence of sensed times, and uses vehicle movement (velocity and acceleration) constraints to combine the hypotheses into a single hypothesis with a longer track.

A *knowledge source instantiation* (KSI) represents the potential application of a particular KS to specific hypotheses. Each node maintains a queue of pending KSIs and, at any given time, must rank the KSIs to decide which one to invoke next. We use an extended Hearsay-II architecture (Figure 3) that lets nodes reason more fully about the intentions or *goals* of the KSIs [2]. As explicit representations of the node's intentions to abstract and extend hypotheses, goals are stored on a separate goal blackboard and are



KSs combine hypotheses to form more encompassing hypotheses on the same or higher levels. Synthesis (s:) KSs generate higher level hypotheses out of compatible lower level hypotheses. Formation (f:) KSs form a track hypothesis from two combinable location hypotheses. Extension (e:) KSs combine a track hypothesis with a compatible location hypothesis to extend the track. Merge (m:) KSs combine two shorter track hypotheses that are compatible into a single longer track. The sensors KS creates signal location hypotheses out of sensed data. Communication KSs are not shown.

Figure 1: Levels of Abstraction and Knowledge Sources.

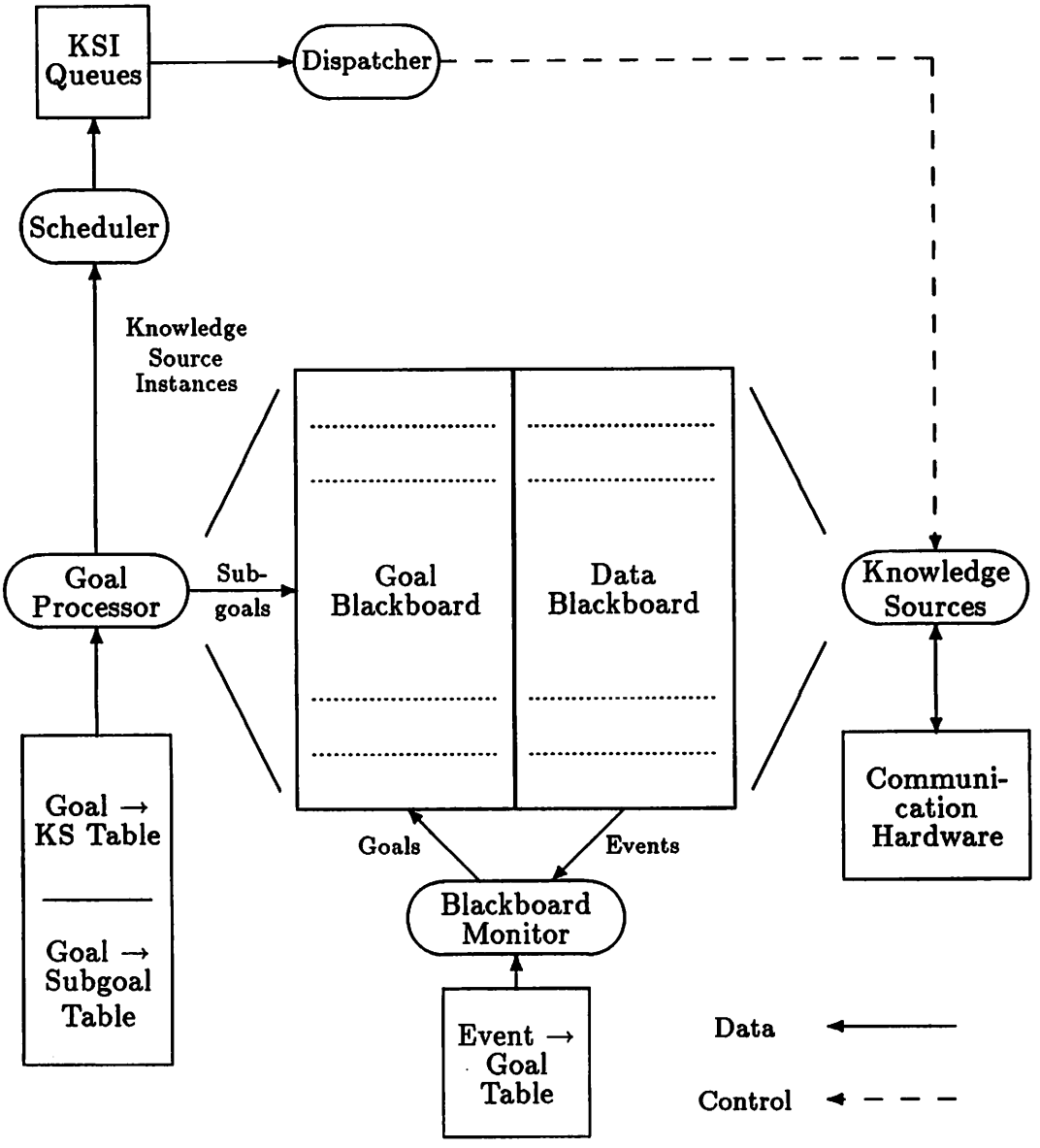
given importance ratings. A goal processing component recognizes interactions between goals and adjusts their ratings appropriately (for example, subgoals of an important goal might have their ratings boosted). The scheduler ranks a KSI based both on the estimated beliefs of the hypotheses it may produce and on the ratings of the goals it is expected to satisfy. Appropriate goal processing can therefore alter KSI rankings to improve local control decisions.

Given a particular problem solving environment, the DVMT simulates the nodes' activities as follows.[1] Each node begins by transforming any sensed data into a set of signal location hypotheses (using the sensors KS). With each new signal hypothesis, the node generates goals to improve upon it and forms KSIs to achieve these goals. After it has created all of its signal hypotheses, a node then chooses a KSI to invoke. The KSI invocation may cause the creation of new hypotheses, which stimulate the generation of more goals, which in turn may cause more KSIs to be formed. The node then chooses another KSI and the cycle repeats until the problem solver generates acceptable solutions. The criteria for deciding whether a hypothesis at the pattern blackboard-level represents an acceptable solution is statically defined before problem solving begins: the user specifies some minimum belief, expected spatial and temporal characteristics, etc.[2] In Section 5, mechanisms that allow the node to decide whether satisfactory solutions have been found based on dynamic criteria—its knowledge about other potential solutions that it can develop—are described.

A snapshot of the contents of the signal, group, and vehicle blackboard-levels while solving a sample problem is shown in Figure 3, where each blackboard-level is represented as a surface with spatial dimensions x and y. At the signal blackboard-level s, there are 10 hypotheses, each for a single time-location (the time is indicated for each). Two of these hypotheses have been synthesized to the group blackboard-level g. In turn, these hypotheses have been synthesized to the vehicle blackboard-level v, where they have been connected into a single track hypothesis (graphically, the two locations are linked). Problem solving proceeds from this point by having the goal processing component form goals

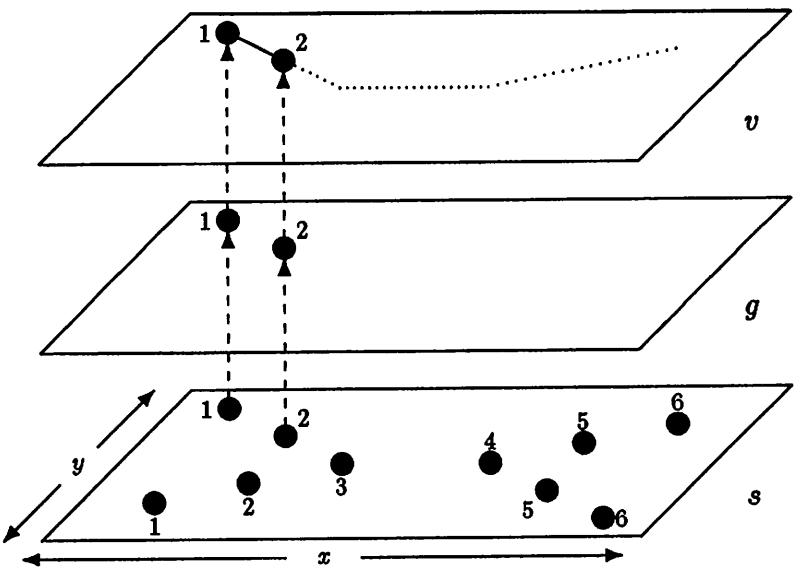
[1] More detailed descriptions can be found elsewhere [3,14].

[2] In many cases, the user specifies exactly the solution that the node should find, but this information is treated as an "oracle" that only tells the node when a solution has been found—the information cannot be used to guide a node's problem solving decisions.



A KS forms hypotheses on the data blackboard, which in turn trigger the formation of goals on the goal blackboard. The goal processor forms KSIs to satisfy the goals, and the KSIs are scheduled. The dispatcher then invokes the KS for the best pending KSI and the cycle repeats.

Figure 2: DVMT Node Architecture.



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 3: An Example Problem Solving State.

(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 3.

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 *and cheaply* 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). A cluster therefore contains:

- A rough specification of the possible solutions that could be constructed from the abstracted data, including a sequence of time-regions (the vehicle's probable movements), and a subset of vehicle types (the vehicle's probable type);
- The characteristics of its abstracted data that could affect the expense of processing and the quality of possible results, including the range of beliefs of the abstracted hypotheses, the blackboard-levels of those hypotheses, the amount of spatial noise (ambiguity in the vehicle's spatial attributes) and frequency noise (ambiguity about what type of vehicle was sensed);
- Pointers to other clusters that abstract hypotheses related to those in this cluster (related temporally, spatially, by blackboard-level, by event-class, or by belief).

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. Since the order of clustering affects which relationships are most emphasized at the highest levels of the abstraction hierarchy, the problem solver clusters to emphasize the relationships it expects to most significantly influence its control decisions. In our vehicle monitoring problem solver, for example, the most important relationships are temporal and spatial: the problem solver constructs solutions by extending partial tracks using data for adjacent sensed times in nearby regions. The problem solver therefore clusters data first based on less important attributes like event-class and blackboard-level, and then later based on spatial and temporal relationships. Issues in clustering in different orders to emphasize other relationships are discussed elsewhere [5].

Because the problem solver works in a dynamic environment, it may receive new sensor data at any time. New data can cause new clusters to be added to the hierarchy and can cause existing clusters to be changed.[3] However, because the majority of clusters and relationships between clusters may be unaffected by new data, the planner should avoid simply discarding the old hierarchy and generating a completely new one. The clustering mechanisms have thus been developed so that new data can be incorporated into the clustering hierarchy by updating the existing hierarchy. The mechanisms begin by using

[3]Because locally generated hypotheses all fall within the scope of clusters already in the clustering hierarchy, they need not be incorporated into it: they would not substantially alter the high-level view provided by the hierarchy.

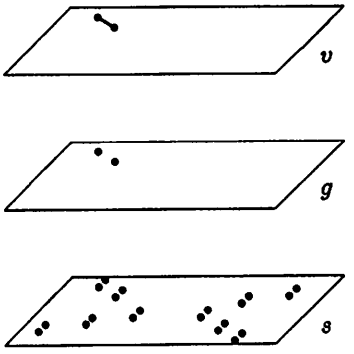
the new data to modify relevant existing clusters at the lowest level if there are any, and otherwise to generate new clusters at this level. After updating the relationships between the lowest level clusters, the clustering mechanisms can then follow these relationships to propagate changes throughout the hierarchy.

To illustrate clustering, consider the clustering sequence in Figure 4, 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 3, except that for each hypothesis in Figure 3 there are now two hypotheses at the same sensed time and slightly different locations. In Figure 4a, 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 4b). 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 4c, 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 4d 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 4d, 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.

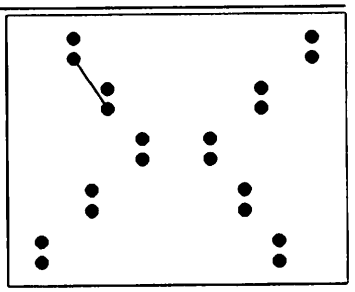
In some respects, the formation of the abstraction hierarchy is akin to a rough pass at

Cluster	Time-regions	Blackboard-levels	Subclusters
c_1^1	$(1x_1y_1)(2x_2y_2)$	v	–
c_2^1	$(1x_1y_1)$	g	–
c_3^1	$(2x_2y_2)$	g	–
c_4^1	$(1x_1y_1)$	s	–
c_5^1	$(1x_1'y_1')$	s	–
.	.	.	.
.	.	.	.
c_{23}^1	$(6x_6'''y_6''')$	s	–



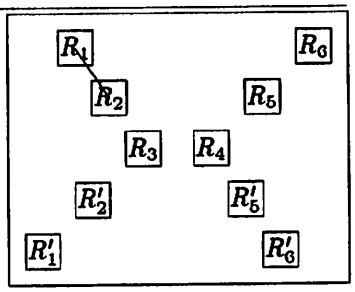
(a)

Cluster	Time-regions	Blackboard-levels	Subclusters
c_1^2	$(1x_1y_1)(2x_2y_2)$	s, g, v	$c_1^1, c_2^1, c_3^1, c_4^1, c_8^1$
c_2^2	$(1x_1'y_1')$	s	c_5^1
c_3^2	$(2x_2'y_2')$	s	c_6^1
.	.	.	.
.	.	.	.
c_{19}^2	$(6x_6'''y_6''')$	s	c_{23}^1



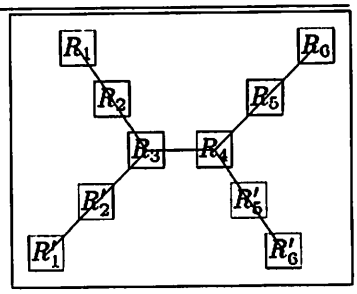
(b)

Cluster	Time-regions	Blackboard-levels	Subclusters
c_1^3	$(1R_1)(2R_2)$	s, g, v	c_1^2, c_2^2, c_3^2
c_2^3	$(1R_1')$	s	c_3^2, c_4^2
c_3^3	$(2R_2')$	s	c_6^2, c_7^2
.	.	.	.
.	.	.	.
c_9^3	$(6R_6')$	s	c_{18}^2, c_{19}^2



(c)

Cluster	Time-regions	Blackboard-levels	Subclusters
c_1^4	$(1R_1)(2R_2)(3R_3)$ $(4R_4)(5R_5)(6R_6)$	s, g, v	$c_1^3, c_4^3, c_5^3,$ c_6^3, c_8^3
.	.	.	.
.	.	.	.
c_4^4	$(1R_1')(2R_2')(3R_3)$ $(4R_4)(5R_5')(6R_6')$	s	$c_2^3, c_3^3, c_4^3,$ c_5^3, c_7^3, c_9^3



(d)

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 4: An Example of Incremental Clustering.

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 example, although the high-level clusters in Figure 4d 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* uses the potential solutions found in the clustering hierarchy as long-term goals when intelligently choosing and ordering 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* (i-goal) represents a desire to extend the track satisfying the previous i-goal into a new

time-location.[4] Because the results of a plan's i-goals must be combined into an overall solution, the problem solver can use the results of one i-goal to focus on actions that will generate compatible results for other i-goals. By pursuing the i-goals in a particular order, therefore, the problem solver can influence the time it takes to solve the problem (the number of actions to achieve the i-goals) and the quality of the solution (how well i-goals' results combine). To determine an order for pursuing the possible i-goals, the planner currently uses three domain-independent heuristics:

Heuristic-1 *Prefer common intermediate goals.* Some i-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 i-goals and then can use these results to better distinguish between the long-term goals. This heuristic is a variation of least-commitment [19].

Heuristic-2 *Prefer less costly intermediate goals.* Some i-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 i-goals that most effectively reflect the relative promise of each long-term goal. When no common i-goals remain, this heuristic triggers work in the areas where the long-term goals differ most.

These heuristics are interdependent. For example, common i-goals may also be more costly, as in one of the experiments described in Section 6.1. The relative influence of each heuristic can be modified parametrically. Moreover, note that the heuristics do not give

[4] 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.

preference to i-goals that involve more highly believed hypotheses. The rationale behind not using belief is that all of the i-goals—those with highly believed *and* those with lowly believed data—must be pursued to complete the plan. The planner should not focus on forming highly believed partial solutions quickly since they may not lead to worthwhile results. Instead, the planner should order the i-goals to most quickly determine whether the *overall* track is worth pursuing and to most efficiently form that track.

Having identified a sequence of i-goals to achieve one or more long-term goals, the planner must determine exactly what KSIs should be invoked to achieve these i-goals. The planner uses coarse 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).[5] The planner uses these KS models to make reasonable predictions about short short sequences of actions to find a sequence that satisfies an i-goal.[6] To reduce the effort spent on planning, the planner only forms detailed plans for the next i-goal: since the results of earlier i-goals influence decisions about how and whether to pursue subsequent i-goals, the planner avoids expending effort forming detailed plans that may never be used.

To find the detailed actions for an i-goal, the planner begins by scanning the abstraction hierarchy for clusters whose data could contribute to achieving the i-goal. The characteristics of this data (summarized by the cluster) are used as input to the KS models. Beginning with this data, the planner models a sequence of KSs and predicts the results and time needs for that sequence. If there are several ways that results satisfying the i-goal could be generated, the planner can model each and choose the one that it expects will generate the best results in the shortest time.[7] In addition, by modeling the sequence of KSs, the planner develops expectations about the results of each action that it uses when monitoring plan execution. When it has found an appropriate sequence of actions, the planner matches the actions to specific KSIs on the KSI-queue: given the initial clusters of data, it

[5]The planner thus has models of KSs that predict a *response frame* based on a *stimulus frame* as in the Hearsay-II system [9].

[6]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.

[7]The user can define a metric for balancing result quality with time needs when deciding which sequence to select. In the experiments outlined in this paper, only one sequence can be found.

finds the KSIs to work on this data and matches actions with KSIs. Since the hypotheses used by later actions (KSIs) may be formed by earlier actions, KSIs must be associated with these actions as the plan is pursued and the earlier actions are taken. When pursuing the plan, the problem solver executes the KSI associated with the next action.

Given the abstraction hierarchy in Figure 4, the planner recognizes that achieving each of the four long-term goals (Figure 4d) entails i-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 i-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[8] 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 $(1 \;\; R_1 or R_1')(2 \;\; R_2 or R_2')(3 \;\; R_3)$ ...);
- 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 where the results of i-goals will be integrated into tracks, depending on the available knowledge sources (in the example, this is level v);
- a record of past actions (initially empty), used by the planner to keep track of the results of past i-goals (so it can determine what results it must plan to integrate) and used by the prediction mechanisms described in Section 5;
- a sequence of the specific actions to take in the short-term (in the example, the detailed plan indicates the KSIs that use synthesis KSs *s:sl:gl* and *s:gl:vl* to drive data in region R_3 at level s to level v);

[8]See Section 6.1 for more information about this plan.

- a rating based on the number of long-term goals being worked on, the effort already invested in the plan (based on the number of past actions), the average ratings of the KSIs corresponding to the detailed short-term actions, and the predicted belief of the solution that the plan is expected to form (see Section 5).

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 by invoking the appropriate KSI. Because this KSI may have been triggered by the previous action's results, the planner may need to locate the KSI on the queue before executing it. If the detailed actions have been exhausted, the planner develops another detailed sequence for the next i-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 checks to see if any new data has arrived that may alter the possible solutions, and if so it incorporates this data into the clustering hierarchy, modifies any plans whose top-level clusters have changed, generates any new plans for new top-level clusters, and then pursues the new and modified plans.

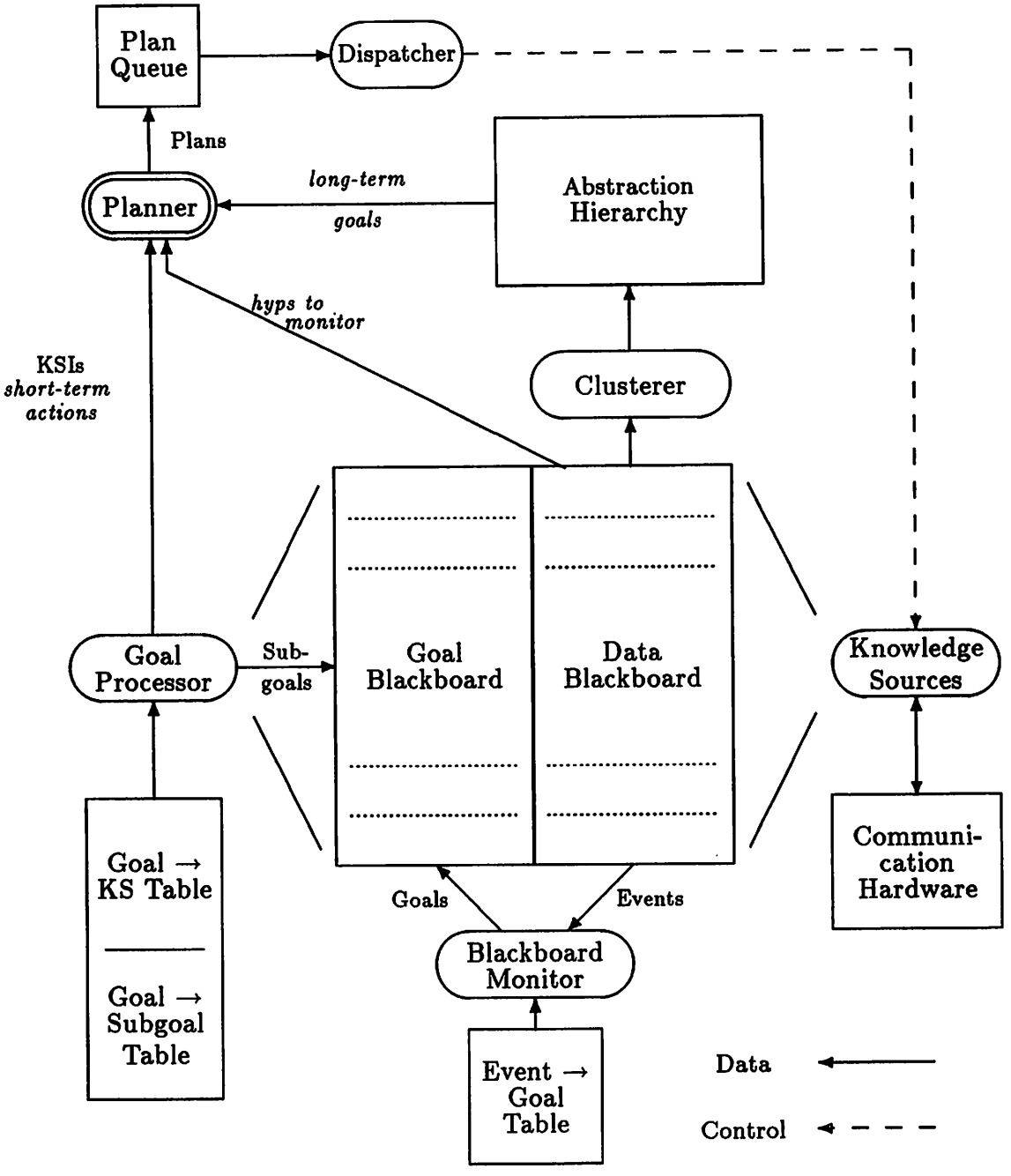
The planner thus generates, monitors, and revises plans, and interleaves these activities with plan execution. In our example, the common i-goals are eventually satisfied and a separate plan must be formed for each of the alternative ways to proceed. After finding a partial track combining data from 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 KSIs in R_2' (this is described more fully in Section 6.1). Because the plan rating is based in part on the ratings of the KSIs that participate in the detailed actions, the plan to work in R_2' next becomes more highly rated than the plan to work in R_2.[9]

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 [13]. 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 Section 6 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 (Figure 5). 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 [19]. 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.

[9] In fact the turn to R_2 exceeds these constraints, as does the turn to R_5', so that the only track that satisfies the constraints is $R_1' R_2' R_3 R_4 R_5 R_6$.



The planner integrates short-term information about specific KSIs that can be performed with the long-term goals found by the clustering mechanisms to form plans. The dispatcher invokes the KS for the next KSI of the best plan. Any new hypotheses that the KS forms are monitored by the planner to make sure that they meet expectations, and the planner alters plans if needed. The cycle then repeats.

Figure 5: The Modified Problem Solving Architecture of a Node.

5. Predictions About Plans

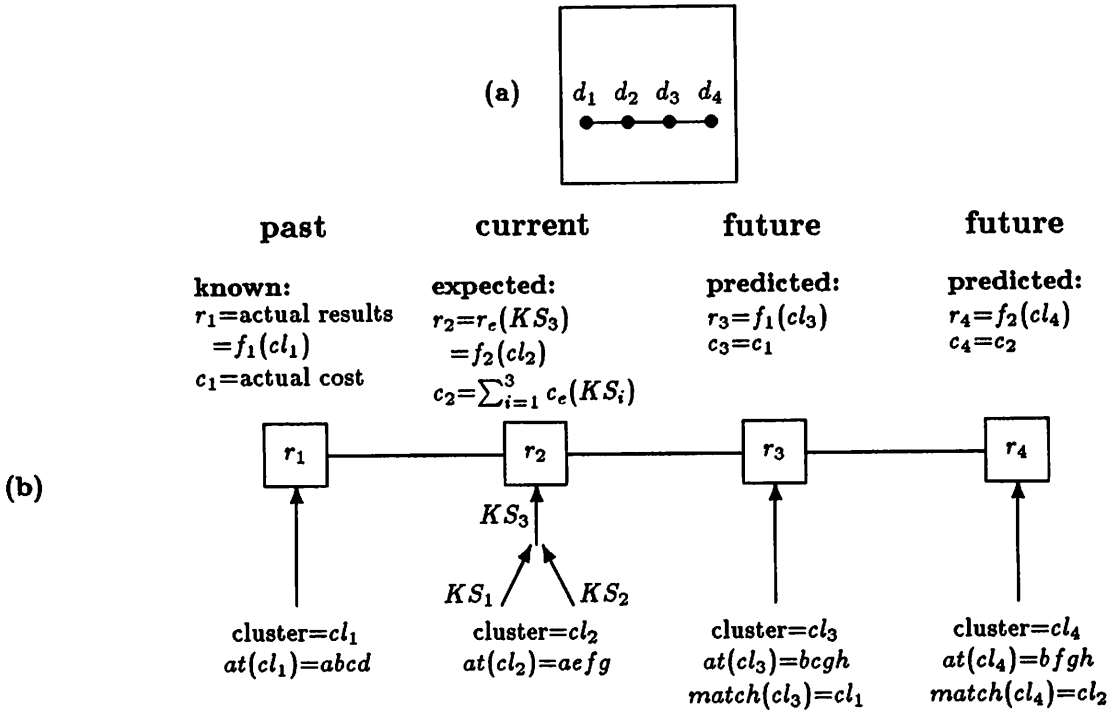
Before it expends effort on a plan, the planner should predict whether the plan is worth pursuing and whether it can be completed in time to meet any deadlines. Using these predictions, the planner can determine whether it should revise plans to better meet any time constraints. In this section, the mechanisms for making and using predictions are described.

5.1 Making Predictions

We have developed mechanisms that use information about the costs and results of the plan's past and current activities, and extrapolates over the future activities to predict the overall costs and results of the plan. In general terms, for a future subgoal the mechanisms find a similar past subgoal (a subgoal with the same basic differences between initial and goal states) and builds expectations for the future based on the past. More specifically, the mechanisms assume that i-goals (subgoals) with similar data (initial states) will be pursued in similar ways: the planner bases predictions for a future activity on the past activity that processed the most similar data. However, the costs of achieving a particular i-goal and the quality of the result depend on the attributes of the data that must be processed (such as how strongly it is sensed, its frequency distribution, and its spatial distribution), and these attributes can vary from one i-goal to another [17].

The process of forming predictions for a simple plan to process the data shown in Figure 6a is illustrated in Figure 6b. The plan will meet its long-term goal of forming a track covering sensed times 1–4 by consecutively achieving its i-goals (generating partial results for each time). When forming the abstraction hierarchy that it uses to develop the plan, the planner clustered together the data for each i-goal. The cluster for an i-goal summarizes the attributes of its data. To predict the costs and results for a future i-goal, our mechanisms match its cluster against the clusters for the current i-goal and any past i-goals, and then extrapolates based on the closest match. Since a plan always has a current i-goal, the mechanisms always have some basis for prediction.

Figure 6 shows the plan in a partially completed state: the i-goal for time 1 was achieved



Overall predicted result (currently) = $average(r_1, r_2, r_3, r_4)$
Overall predicted cost (currently) = $c_1 + c_2 + c_3 + c_4 = 2c_1 + 2c_2$

Definitions:

r_i	result satisfying intermediate goal i
cl_i	initial clustered hypotheses for intermediate goal i
$at(cl_i)$	attributes of cl_i, here simply represented as letters
$match(cl_i)$	closest matching past or current i-goal cluster for cl_i
c_i	cost (in time) of forming r_i from cl_i
f_i	function relating cl_i with r_i for past and current i-goals
KS_j	KS activity j for achieving current i-goal
$r_e(KS_j)$	expected result generated by KS_j based on KS models
$c_e(KS_j)$	expected cost of performing KS_j based on KS models

The plan data is shown in (a). A partially completed plan to form a track connecting the data is shown in (b): intermediate goal 1 (to form r_1) has been achieved, intermediate goal 2 is currently being worked on, and intermediate goals 3 and 4 will be worked on in the future. The results and costs for intermediate goal 1 are known and the expected results and costs for intermediate goal 2 are derived using models of KSs. The results and costs for 3 and 4 are found by matching their clusters against those of 1 and 2, and applying knowledge about the closest match. Note that when the plan started, only the expected results for 1 (the current intermediate goal at that time) were known and predictions for 2–4 were based only on these expectations. As the plan has progressed, the predictions have improved since, for example, better predictions for 4 are made when the more closely matching intermediate goal 2 becomes current. Finally, KS_1 and KS_2 independently process different subsets of the data in cl_2 to generate supporting hypotheses for KS_3 (hence the representation); the belief in KS_3's result is decreased if only one of the supporting KSs is executed.

Figure 6: Simple Example of Making Predictions.

in the past, the i-goal for time 2 is currently being worked on, and the other i-goals are still pending. For a past i-goal (time 1), the results and the cost of forming those results are known. The mechanisms find the relationship between the attributes of that i-goal's cluster and the results achieved, representing the result as some function of the cluster's attributes. For the current i-goal (time 2), the planner has used models of the KSs to estimate the attributes of the results of a KS action (based on the attributes of its input data) and the costs of the action. The overall costs of the i-goal is the sum of the estimated KS costs, and its result is simply the estimated result of the last KS action. Once again, the relationship between this result and the i-goal's cluster is found.

The first step in predicting the results and costs for a future i-goal is to match its cluster against the past and current i-goals' clusters to find the closest match. To simplify Figure 6, we represent the attributes as letters, and the closest match is the past or current cluster with the most letters in common: cl_3 (with attributes $bcgh$) is closer to cl_1 ($abcd$) than cl_2 ($aefg$) while cl_4 ($bfgh$) is more like cl_2 ($aefg$). In our implementation, the attributes currently considered are the clusters' blackboard-levels, the volume of data hypotheses they represent, and the time that the data was processed. Given a future i-goal's cluster, the algorithm scans the past and current clusters for those with the closest blackboard-levels (because data at close blackboard-levels undergo similar processing). If only one is found it is returned, but if two or more are equally close then of these the ones with the closest volume of data are found (since more hypotheses may mean substantially more processing is needed). Again, if only one is found then it is returned, but if several are equally close then the one of these whose i-goal was most recently worked on is returned (because more recent activity probably reflects future activity better).

When the closest match is found, the future i-goal's cost is predicted to be the same as the cost for the matching cluster's i-goal—the processing time needed is expected to be the same (Figure 6). To predict the result quality, the planner uses the relationship between the matching cluster and its result (relating the average belief of the cluster's hypotheses with the result's belief) and predicts that the same relationship will hold between the future i-goal's cluster and result (Figure 6). For example, say the matching i-goal's result has a belief twice that of the average of its cluster's hypotheses (this happens, for instance,

when the KSs see a known frequency distribution in the clustered hypotheses and thus have high belief in their combination). Then the belief of the future i-goal's result is predicted to be twice the average belief of its cluster's hypotheses.

The predicted overall cost of a plan is the sum of the i-goals' costs. The predicted overall result of the plan is also a combination of the i-goal's results. When KSs combine data for individual time-locations into a track, the belief of the track hypothesis is a combination of the individual time-location beliefs—usually the average of these beliefs decreased by some penalties for any unlikely vehicle movements (which can only be recognized over a sequence of time-locations). Because the plan's high-level view is too imprecise to recognize unlikely vehicle movements, the prediction mechanisms estimate the overall result's belief simply as the average of the beliefs of the i-goals' actual or predicted results. The predictions thus tend to overestimate rather than underestimate the actual result.

As a plan is pursued, predictions about that plan will generally improve, both because actual costs and results replace those predicted so that overall predictions improve, and because more past experience increases the chances of finding more relevant past i-goals for making predictions about future i-goals. The plan in Figure 6, when it was just starting, could only base predictions on the expected result and cost of the i-goal for time 1 (which was its current i-goal at that time). Since it now can also base predictions on the i-goal for time 2, it can make better predictions about time 4's i-goal (which matches 2 more closely than 1), and when time 3 becomes the current i-goal, the predictions for 4 will be even better (since it most closely matches 3). By interleaving planning and execution, the predictions about future activities tend to improve as experience with the plan is gained. In fact, by maintaining an extensive database of all its past problem solving experience, the planner could gain enough knowledge to make very good predictions for any contingency. However, techniques for efficiently acquiring, saving, and perhaps generalizing this knowledge are machine learning tasks that our research does not address.

5.2 Using Predictions

A problem solver is seldom given exactly as much time as it needs to solve a problem, especially when the costs of problem solving are initially uncertain. When given extra time,

the problem solver should make intelligent decisions about when to terminate problem solving: it should sufficiently explore the possible solutions to be reasonably confident that it has found the best one, but should avoid wasting time generating solutions that could not possibly be of use. When given less time than it needs, it should revise its plans to generate some inferior but still acceptable solution if it can.

When a node has found a solution with time to spare and needs to decide whether to terminate problem solving, the planner compares the solution with the predicted results of competing plans. Since the node is attempting to form the best solution, any plans that predict better results should be pursued. Any plans that predict much worse results should not. Plans that predict somewhat worse results might still be worth pursuing, since the predictions may underestimate the actual results, especially when the plan's i-goals work with very dissimilar data. Our mechanisms allow the user to specify how close a competing plan's predicted results must be to the best solution already found for the plan to warrant further work. The planner can be conservative if the window of acceptable plans is so large that it is unlikely to miss any good solutions, or if the window is very small the node can more quickly propose a solution at the risk of missing a better solution had it kept looking. Unlike our earlier termination technique of using statically defined criteria such as generating a hypothesis with certain predefined characteristics, the new mechanisms allow the node to dynamically compare the solutions it has developed with those it predicts it could develop and decide when it has found the best solutions.

Instead of having extra time and needing to decide whether it is worthwhile exploring alternative solutions, the node might have too little time to generate even a single solution. The node might face tight deadlines. Without the ability to predict how long a plan will take, the node would simply pursue a plan and hope to finish in time. Our new mechanisms, however, allow the node to roughly predict how long a plan will take. Before it gets far with a plan, the node can recognize that the predicted time needed by the plan will probably exceed the time available, and can do something about it. The planner can respond to this situation in any of a number of ways [15]: it can reduce the needs of the plan by making the plan's long-term goal less constrained (for example, it may plan to form a shorter track); it may replace costly problem solving activities with less costly activities

which may produce inferior but acceptable results; or it may choose another, cheaper plan that can be finished.

Our current implementation has two simple mechanisms for revising a plan to meet deadlines. The first mechanism is to reduce the scope of the plan's long-term goals by ignoring data for some of the sensed times. Because a vehicle's more recent movements are usually most important (for recognizing pending collisions with other vehicles, for example), our mechanism simply drops i-goals for earlier sensed data until it predicts that the plan to process the remaining data will meet the deadlines. The other mechanism is to drop plan steps that corroborate hypotheses and increase the belief in the solution but do not affect the scope of the solution. For the current i-goal in Figure 6, for example, KS_1 and KS_2 both supply supporting hypotheses for KS_3. The results formed by KS_3 will be better (more highly believed) if both supporting KSs are executed, but an inferior result can still be made if one of the two is dropped. In a given situation where a plan will exceed deadlines, preferences about which of these mechanisms to try first depends on whether the agent needing the solution (currently by the user) tells the node that belief is more important than scope or *vice versa*.[10]

6. Evaluation

We illustrate the advantages and the costs of our planner in three problem solving situations. To simplify the discussion, the situations emphasize certain aspects of the planning mechanisms. The first presents an example of how the high-level view allows the planner to rearrange problem solving activities so that it more quickly identifies promising activities and more effectively solves the problem, all with an acceptable amount of overhead. The second situation shows how incremental planning is advantageous when some goals cannot be achieved so that plan monitoring and repair are needed. Finally, the third situation examines how the prediction mechanisms improve decisions about which plans to pursue and allow the planner to decide when to revise plans to meet deadlines and when to terminate

[10]Since the KSs compute belief as essentially the average of the individual pieces of a track, belief does not necessarily increase with increasing scope—belief is more a function of the amount of processing done on each piece of a solution than of how many pieces have been processed. Thus, our two mechanisms can treat scope and belief as being relatively independent attributes of a solution.

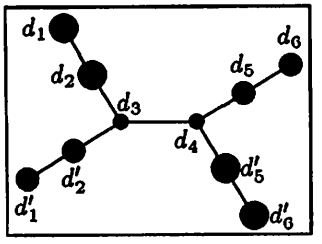
problem solving since better solutions will probably not be found.

6.1 Situation 1: Planning to Resolve Uncertainty

The first problem solving situation is shown in Figure 7. This situation is the same as in Figure 4 except that each region has one hypothesis. Also note that the data in the common regions is most weakly sensed. When evaluating how the new mechanisms perform in this situation, 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 using problem solving time: since each KSI is simulated to take 1 time unit, generating a solution at an earlier time means fewer KSIs were invoked so better control decisions were made. 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 for this problem solving situation are summarized in Table 1. To determine the effects of the new mechanisms, the problem was solved both with and without them, and for each case the simulated problem solving time (which equals 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. Finally, a variation on this environment introduces noisy data into the common areas of the plans to determine how emphasizing different heuristics for ordering i-goals can affect problem solving.

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 1. Without these mechanisms (E1), the problem solver begins with the most highly sensed data (d_1, d_2, d'_5, and d'_6). 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), the planner



d_i = data for sensed time i

● = strongly sensed

● = moderately sensed

● = weakly sensed

possible solutions: $ps_1 = d_1d_2d_3d_4d_5d_6$
$ps_2 = d_1d_2d_3d_4d'_5d'_6$
$ps_3 = d'_1d'_2d_3d_4d_5d_6$
$ps_4 = d'_1d'_2d_3d_4d'_5d'_6$

acceptable solutions: ps_3

Time	Plan	Possible Solutions	Current Result	Pending I-goal Order
1	$plan_1$	ps_1, ps_2, ps_3, ps_4	none	(3 4 2 1 5 6)
8	$plan_1$	ps_1, ps_2	d_3d_4	(2 1 5 6)
	$plan_2$	ps_3, ps_4	d_3d_4	(2 1 5 6)
16	$plan_1$	ps_1, ps_2	d_3d_4	(2 1 5 6)
	$plan_2$	ps_3	$d'_1d'_2d_3d_4$	(5 6)
	$plan_3$	ps_4	$d'_1d'_2d_3d_4$	(5 6)
24	$plan_1$	ps_1, ps_2	d_3d_4	(2 1 5 6)
	$plan_2$	ps_3	$d'_1d'_2d_3d_4d_5d_6$	()
	$plan_3$	ps_4	$d'_1d'_2d_3d_4$	(5 6)

The first problem solving situation is displayed, along with the possible solutions found by clustering the data and the acceptable solutions that can eventually be generated by the problem solver. The plans at various problem solving times are shown to indicate the evolution of the plans as problem solving progresses. For each plan is given the possible solutions that it is expected to work toward, the result it has generated to this point, and the pending i-goals in the order that they will be attempted.

Figure 7: Problem Situation 1.

Expt	Plan?	STime	Rtime	Goals	Comments
E1	no	58	17.2	262	–
E2	yes	24	8.1	49	–
E3	yes	32	19.4	203	independent goal processing and planning
E4	no	58	19.9	284	noise in d_3, d_4
E5	yes	64	17.3	112	noise in d_3, d_4; Heuristic-1 predominant
E6	yes	38	16.5	71	noise in d_3, d_4; Heuristic-2 predominant

Legend

Plan?: Are the new planning mechanisms used?
Stime: The simulated time (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: Experiment Summary for Situation 1.

recognizes that all four possible solutions share common data at d_3 and d_4 and thus builds a single plan to initially pursue all four possible solutions (Figure 7, time 1). When the track d_3d_4 has been formed, the plan is divided into two separate plans: one to extend toward data d_2 and the other to extend toward data d'_2 (Figure 9, time 8). The track d_3d_4 is used by the goal processing mechanisms to form a goal indicating where good data for sensed time 2 is most likely to lie, as is shown in Figure 8.

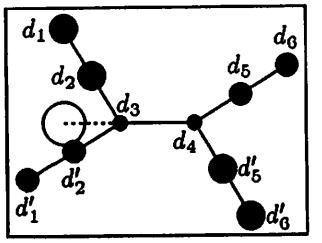


Figure 8: Expected Extension of Track d_3d_4.

This goal overlaps only with the data in d'_2, so the KSIs to work on this data have their ratings increased while the ratings for KSIs in d_2 are unchanged. Although originally the KSIs for d_2 were more highly rated (since they work with more strongly sensed data), the KSIs to work in d'_2 are now more highly rated. The plan to work on d'_2 is therefore pursued because its more highly rated KSIs cause its rating to increase. Similarly, when this plan is divided into separate plans to extend either to d_5 or to d'_5 (Figure 7, time 16), goal processing affects KSI ratings so that the plan to work on d_5 is preferred. Since the planner avoids processing the strongly sensed but noisy data in d_1, d_2, d'_5, and d'_6, the solution is found much more quickly (in fact, in *optimal* time [7]).

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, without the planner to control it, the goal processing builds goals and subgoals based on less complete results, and these less precise subgoals do not selectively increase the ratings of appropriate KSIs. For example, with the planner controlling goal processing, a goal to find data for sensed time 2 is only generated when the track covering d_3d_4 is formed, and this goal selectively increases the KSI ratings

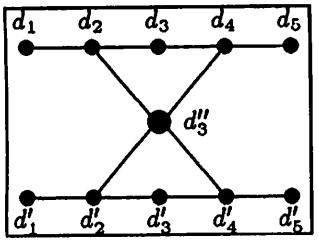
for d_2' as described above. When goal processing is not controlled by the planner, a goal to find data for sensed time 2 is formed earlier, based only on the result from d_3. Without information about later vehicle locations, the goal indicates that data in any direction around d_3 is equally likely, and so increases the ratings of KSIs in both d_2 and d_2'—the KSIs in d_2 remain more highly rated. The planner then prefers the plan to work on d_2 over the correct plan. If the planner does not control goal processing, therefore, control decisions deteriorate and more KSIs may be invoked (E3 compared to E2).

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. In experiments E4–E6, areas d_3 and d_4 contain numerous competing hypotheses. If the planner initially works in those areas (E5), then substantial time (KSIs) is required to develop all of these hypotheses—more time than if the planner was not used at all (E4). However, by estimating the relative costs of the alternative i-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 simulated time needed to solve the problem and the actual computation time are thus reduced (E6).

6.2 Situation 2: Monitoring and Repairing Plans

In the second problem solving situation, shown in Figure 9, two solutions must be found, corresponding to two vehicles moving in parallel. Note that no areas are common to all possible solutions. The experimental results for this environment are summarized in Table 2. 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 provided by the abstraction hierarchy allows the planner in experiment E8 to recognize the six possible 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



d_i = data for sensed time i

● = strongly sensed

● = moderately sensed

● = weakly sensed

possible solutions: $ps_1 = d_1d_2d_3d_4d_5$
$ps_2 = d_1'd_2'd_3'd_4'd_5'$
$ps_3 = d_1d_2d_3''d_4d_5$
$ps_4 = d_1d_2d_3''d_4'd_5'$
$ps_5 = d_1'd_2'd_3''d_4d_5$
$ps_6 = d_1'd_2'd_3''d_4'd_5'$

acceptable solutions: ps_1, ps_2

Time	Plan	Possible Solutions	Current Result	Pending I-goal Order
1	$plan_1$	$ps_3 - ps_6$	none	(3 2 1 4 5)
	$plan_2$	ps_1	none	(3 2 1 4 5)
	$plan_3$	ps_2	none	(3 2 1 4 5)
4	$plan_1$	ps_3, ps_4	d_3''	(2 1 4 5)
	$plan_4$	ps_5, ps_6	d_3''	(2 1 4 5)
8	$plan_1$	ps_3, ps_4	d_2d_3''	(1 4 5)
12	$plan_1$*	ps_3, ps_4	d_1d_2	(4 5)
20	$plan_1$*	ps_3, ps_4	d_1d_2	(4 5)
	$plan_4$*	ps_5, ps_6	$d_1'd_2'$	(4 5)
23	$plan_2$	ps_1	d_3	(2 1 4 5)
	$plan_4$*	ps_5, ps_6	$d_1'd_2'$	(4 5)
24	$plan_2$ **	ps_1	$d_1d_2d_3$	(4 5)
36	$plan_2$	ps_1	$d_1d_2d_3d_4d_5$	()
	$plan_3$	ps_2	d_3'	(2 1 4 5)
37	$plan_3$ **	ps_2	$d_1'd_2'd_3'$	(4 5)
45	$plan_1$*	ps_3, ps_4	d_1d_2	(4 5)
	$plan_2$	ps_1	$d_1d_2d_3d_4d_5$	()
	$plan_3$	ps_2	$d_1'd_2'd_3'd_4'd_5'$	()
	$plan_4$*	ps_5, ps_6	$d_1'd_2'$	(4 5)

The second problem solving situation is displayed, along with the possible solutions found by clustering the data and the acceptable solutions that can eventually be generated by the problem solver. The plans at various problem solving times are shown to indicate the evolution of the plans as problem solving progresses. Plans not included at a given time have not changed since the last time. For each plan is given the possible solutions that it is expected to work toward, the result it has generated to this point, and the pending i-goals in the order that they will be attempted. Plans marked with a * have been aborted, and plans marked with ** have been repaired.

Figure 9: Problem Situation 2.

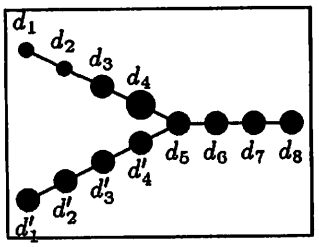
indicates a preference for d_3 and d_3'). Since it covers the most long-term goals, $plan_1$ is pursued first—a reasonable strategy because effort is expended on the solution path if the plan succeeds, and if the plan fails then the largest possible number of candidate solutions are eliminated. After developing d_3'' (Figure 9, time 4), $plan_1$ is divided into two plans to combine this data with either d_2 or d_2'. One of these equally rated plans, in this case $plan_1$, is chosen arbitrarily and forms the track $d_2 d_3''$, which then must be combined with d_1 (Figure 9, time 8). However, because of vehicle turning constraints, only $d_1 d_2$ rather than $d_1 d_2 d_3''$ is formed (Figure 9, time 12). 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 (Figure 9, time 20). $Plan_2$ is then invoked, and after developing d_3 (Figure 9, time 23) it finds that d_2 has already been developed (by the first aborted plan). However, the plan monitor detects that the predicted result, $d_2 d_3$ was not formed, and the plan is repaired by inserting a new action (triggering a new KSI) that takes advantage of the previous formation of $d_1 d_2$ (the KSI finds that hypothesis already on the blackboard) to generate $d_1 d_2 d_3$ (Figure 9, time 24). 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 pursued, repaired (Figure 9, time 36, 37), and successfully completed (Figure 9, time 45). Finally, note once again that, if the planner does not control goal processing (E9), unnecessary overhead costs are incurred.

Expt	Plan?	STime	Rtime	Goals	Comments
E7	no	73	21.4	371	–
E8	yes	45	11.8	60	–
E9	yes	45	20.6	257	independent goal processing and planning

Legend

Plan?:	Are the new planning mechanisms used?
Stime:	The simulated time (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 2: Experiment Summary for Situation 2.



d_i = data for sensed time i

● = strongly sensed

● = moderately sensed

● = weakly sensed

possible solutions: $ps_1 = d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8$

$ps_2 = d'_1 d'_2 d'_3 d'_4 d_5 d_6 d_7 d_8$

best solutions: ps_2

The third problem solving situation is displayed, along with the possible solutions found by clustering the data and the best solutions that can eventually be generated by the problem solver (since in this case both of the possible solutions are valid solutions but one is better than the other).

Figure 10: Problem Situation 3.

6.3 Situation 3: Dealing With Time Constraints

In the third situation, shown in Figure 10, there are only two possible solutions: the vehicle can start either in the upper-left or lower-left corner. The track extending to the upper-left involves weak and strong data, but more weak than strong, while the lower track is moderately sensed throughout. The overall belief of the lower track is higher than the upper, and therefore represents the best solution. The experimental results are summarized in Table 3. For each experiment is shown the time when the best solution was found, the time the problem solving terminated (when appropriate), the sensed times of the solution track, the belief of the solution track, and comments about the experiment.

We begin by showing that predictions can improve a node's control decisions about which plans to pursue. For experiment E10, the new prediction mechanisms are not employed. The node begins by forming the common track $d_5 d_6 d_7 d_8$ and then extends this track backward to earlier times. Because the d_4 is more strongly sensed than d'_4, it extends $d_5 d_6 d_7 d_8$ toward the upper-left. Only when the node integrates the weak data (d_1 and d_2) does its belief in the overall track diminish so that it perceives the lower track as a potentially better solution. It therefore forms both possible solutions. With the new prediction mechanisms (experiment E11), the planner predicts that the upper track's weak data will reduce belief in that track, and that the overall belief of the lower track will probably be better. Hence, after it generates $d_5 d_6 d_7 d_8$, it pursues the lower track and

Expt	STime	TTime	Track	Belief	Comments
E10	57	–	1 – 8	*high*	No predictions
E11	40	–	1 – 8	*high*	Predictions
E12	40	57	1 – 8	*high*	Explore all solutions
E13	40	40	1 – 8	*high*	Stop with best predicted solution
E14	–	36	–	–	No predictions, deadline = 36
E15	36	36	1 – 8	*mod*	Predictions, deadline = 36, large scope
E16	32	32	1 – 8	*low*	Predictions, deadline = 32, large scope
E17	30	30	2 – 8	*low*	Predictions, deadline = 30, large scope
E18	28	30	4 – 8	*high*	Predictions, deadline = 30, high belief

Legend

Expt:	The experiment
STime:	Time at which the best solution was found
TTime:	Time at which problem solving terminated
Track:	Times spanned by best solution track
Belief:	Belief in best solution track

Table 3: Experiment Summary for Situation 3.

generates the best solution substantially sooner than in E10.

Depending on how conservatively it should solve the problem, the node may terminate problem solving only after it has explored every possible solution, or it may end as soon as it believes (based on its predictions) that it has found the best solution. In experiment E12, the node explores all possible solutions, and therefore terminates problem solving much later than in experiment E13 where it stops as soon as it predicts that it has found the best solution. Although both experiments find the best solution equally fast, E12 spends a lot of time and energy verifying that it was the best solution. The predictions in these experiments are sufficiently accurate so that E13's decision to terminate is correct. In other problem situations such a decision might be premature—the predictions might underestimate the quality of potential results and the best solution might be missed. How conservative a node's termination decisions should be depends on the problem situation and how much time it has.

In this environment, a highly believed hypothesis spanning the entire track cannot be formed in less than 40 time units. A deadline of 36 time units was used in experiments E14 and E15. Since any partially developed track needs more work before it meets solution criteria (it must be processed to the pattern blackboard-level), a node without prediction

abilities has no solutions by time 36 (E14). When it can predict that there is insufficient time to form the best solution (E15), the planner revises the plan to meet the deadline: told by the experimenter (the consumer of the solution) to favor scope over belief, it removes enough plan steps (KSs for supporting hypotheses) so that it still forms a solution covering the entire track but with only a moderate belief. With only 32 time units to work with (E16), the planner removes plan steps so that the solution covers the entire track but with low belief.

Given a deadline of time 30 (E17), the planner must employ both mechanisms since it still expects to exceed the deadline after it has removed all the unnecessary plan steps. It reduces the scope of the goal, dropping the earliest sensed time (time 1), and generates a hypotheses with low belief spanning times 2–8. Finally, given the same deadline of time 30 but told by the experimenter to generate a solution with high belief, the planner no longer drops steps (that would reduce belief) but instead reduces the scope of the solution to span times 4–8 (E18). Note that in these environments where each plan step (KSI) takes one time unit, the planner can drop just enough plan steps to meet the deadline exactly (E15 – E17). When it needs to get high belief and drops entire i-goals for less important sensed times, the planner may not meet deadlines exactly (E18) since these i-goals take several steps to achieve.

7. Conclusions

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 [9] or how various errands are related spatially or temporally [11].[11] Planning requires simple models of KSs (mapping *stimulus frames* to *response frames* [9]), 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 blackboard-based problem solvers (and more generally, data-directed problem solvers) could incorporate similar abstraction and planning mechanisms to improve their control decisions.

By making predictions about its plans, a blackboard-based problem solver can perform effectively when faced with time constraints. If it predicts that a plan will exceed a deadline, the planner can revise the plan appropriately, while if it has time to spare, the planner can more intelligently decide whether to terminate problem solving after finding a solution if it predicts that pursuing alternative plans will be a waste of time. This paper described how the prediction mechanisms work: how future activities are related to past activities so that past experience can be used to estimate how things will go in the future, and how predictions are used to meet deadlines and to make termination decisions. Basing predictions on past experience assumes that past and future activities are somehow related. If this assumption is false, then other mechanisms for prediction that get expectations some other way are needed. However, in our system, as in typical blackboard-based problem solvers, this assumption is generally true: the problem solver will usually go through fairly repetitive actions as it extends and refines different partial solutions. Also, the quality of the predictions depends on the characteristics of the situation and the past experiences of the node. The planner should not expect the predictions to be completely accurate, and thus should use the predictions only to influence its decisions, not dictate them, so that it retains flexibility to explore alternative solutions that might be better.

Not only are the planning and prediction mechanisms important for building blackboard-

[11] 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 [9].

based problem solvers that can work effectively by themselves, but they are also vital for coordinating cooperating problem solvers. A network of such problem solvers that are cooperatively solving a single problem could communicate about their plans and predictions, 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 [6,5,7,8]. 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.

Our new mechanisms, though they address issues previously neglected, should be integrated with other control techniques to be fully flexible. 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 [11]. We also expect to improve on the mechanisms that use the predictions. Our techniques for modifying plans to meet deadlines are still rudimentary, involving simply reducing the scope of the desired solution or dropping plan steps. We want to explore a variety of mechanisms for replacing costly actions with less costly ones, for developing less exact hypotheses when time is limited, and for deciding the best way to modify a plan in a specific situation [15]. In addition, we want to better explore how termination criteria can adapt to different problem solving states. Based on the results we have outlined in this paper, we anticipate that the further development of mechanisms for developing abstract views and incremental planning to control blackboard-based problem solvers, as well as mechanisms for making predictions and using them to meet problem solving deadlines, will greatly enhance the performance of these problem solving systems, will lead to better coordination in distributed problem solving networks, and will increase our understanding of planning and action in highly uncertain domains.

References

[1] 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.

[2] 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.

[3] Daniel David Corkill. *A Framework for Organizational Self-Design in Distributed Problem Solving Networks.* PhD thesis, University of Massachusetts, Amherst, Massachusetts 01003, February 1983. Available as Technical Report 82-33, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, December 1982.

[4] 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.

[5] 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.

[6] Edmund H. Durfee and Victor R. Lesser. *Using Partial Global Plans to Coordinate Distributed Problem Solvers.* Technical Report 87-06, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, January 1987. Also to appear in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, August, 1987.

[7] 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.*

[8] 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.

[9] 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.

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

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

[12] 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.

[13] Joseph A. Hernandez, Daniel D. Corkill, and Victor R. Lesser. *Goal Processing in Blackboard Architectures.* Technical Report 87-24, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, March 1987.

[14] 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.

[15] Victor R. Lesser and Jasmina Pavlin. *Real-time control in AI problem solving.* Technical Report 87-?, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, 1987 (in preparation).

[16] 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.

[17] Jasmina Pavlin. Predicting the performance of distributed knowledge-based systems: a modeling approach. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 314–319, August 1983.

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

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

[20] W. A. Woods. Shortfall and density scoring strategies for speech understanding control. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 13–26, August 1977.