AI Magazine Volume 9 Number 1 (1988) (© AAAI)

Approximate Processing in Real-Time Problem Solving

Victor R. Lesser, Jasmina Pavlin, & Edmund Durfee

We propose an approach for meeting real-time constraints in AI systems that views (1) time as a resource that should be considered when making control decisions, (2) plans as ways of expressing control decisions, and (3) approximate processing as a way of satisfying time constraints that cannot be achieved through normal processing. In this approach, a real-time problem solver estimates the time required to generate solutions and their quality. This estimate permits the system to anticipate whether the current objectives will be met in time. The system can then take corrective actions and form lower-quality solutions within the time constraints. These actions can involve modifying existing plans or forming radically different plans that utilize only rough data characteristics and approximate knowledge to achieve a desired speedup. A decision about how to change processing should be situation dependent, based on the current state of processing and the domain-dependent solution criteria. We present preliminary experiments that show how approximate processing helps a vehicle-monitoring problem solver meet deadlines and outline a framework for flexibly meeting real-time constraints.

AI promises to solve many knowledge-intensive problems that cannot be solved in conventional ways. An important issue facing the introduction of advanced AI technology into real-time applications is the ability of an AI system to meet deadlines (Fehling, Joerger, and Sagalowitz 1986; Hayes-Roth 1987a, 1987b). For example, an AI problem solver often must meet deadlines because someone or something has a specific, time-dependent use for the solution. The wider application of AI problem-solving systems, therefore, demands that approaches for meeting real-time constraints be developed.

Typical approaches to real-time computing assume a task's priorities and resource needs (including time) are completely known in advance and are unrelated to those of other tasks, so that a control component can schedule tasks based on their individual characteristics (Ramamritham and Stankovic 1984; Stankovic, Ramamritham, and Cheng 1985). If more tasks exist than the system can process, the decision about which tasks to ignore is simple and local, usually based only on task priority. However, tasks in problem-solving applications are interdependent because they search different parts of the solution space to solve related subproblems. Because the solution space can be too large to search exhaustively within any practical time limit, search is heuristic, and only a small number of all potential tasks are performed. Although it might be able to form rough estimates about the time needed to perform this heuristic search, a problem solver cannot precisely determine the time needs beforehand because the information that influences heuristic decisions might change as search progresses. Moreover, solving some subproblems can affect the importance and time needs of tasks to solve related subproblems. In an interpretation system (Lesser and Corkill 1983), for example, a problem solver cannot fully predict the time needed to identify the most consistent interpretation until it has taken some steps to characterize the amount and type of noise in its input data. To satisfy real-time constraints, therefore, a problem solver's control component cannot and should not reason about individual tasks but instead should reason about how groups of tasks lead to acceptable overall solutions.

When the real-time control problem is viewed from the perspective of the overall solution, the emphasis of real-time response changes. Although faster hardware and more efficient and predictable use of resources by the underlying operating system can improve the real-time processing of individual tasks, a problem solver also needs real-time control mechanisms for dealing with the larger-grained issues in solving complex problems involving hundreds of interdependent tasks. Our research concentrates on developing these mechanisms (and is not concerned with low-level activities such as input-output and interrupt handling). Our goal is to develop problem solvers that will adaptively generate the most acceptable solutions which meet deadlines and the users' needs when they cannot find optimal solutions in time. Our criterion for problem-solving success is, thus, similar to the idea of "satisficing" developed by March and Simon (1958) (Simon 1969). In addition, because a real-time problem solver uses imprecise predictions about its

activities, it cannot guarantee it will meet deadlines—just that it will meet deadlines if its predictions do not underestimate the time needs of problem-solving tasks. To reduce the risk of exceeding deadlines, the problem solver can leave extra time "just in case," but this extra time can cause it, in its haste, to generate poorer solutions than it had to.

Our new approach to real-time control requires that the system reason about its objectives, its problem-solving state, and the plans for achieving its objective from the current state. An objective represents the criteria for acceptable solutions and should define a range of acceptable solutions and preferences among them (as is illustrated later). A problem-solving state includes partial solutions obtained; active and satisfied goals; and relationships among goals, currently pursued plans, and alternative plans. A plan consists of a sequence of problem-solving activities, the estimated time needs of these activities, and the results they are predicted to generate.

The ability to specify problem-solving plans is necessary not only for recognizing that an expected solution cannot be derived within the time constraints from the current state but also for evaluating alternative solution paths and finding out whether a proposed solution is feasible from the current state. If the problem solver's rough estimate of when the best possible solution will be formed exceeds the time constraints, the system should be able to modify its plans and perform approximate processing that trades off the quality of a solution against the time to derive it, where the dimensions of solution quality are domain specific.

To meet its deadlines, the problem solver could make a rough pass at solving the problem and then use any remaining time to incrementally refine this solution. Alternatively, it could incorporate just enough approximations into its plan steps to generate a solution within the time constraints. The advantage of incremental refinement is that the system has some solution at any given time, but refining parts of a solution and propagating these changes can be costly. In this article, therefore, we concentrate on incorporating approximate processing into plans to efficiently construct acceptable solutions within the time constraints. However, although we do not plan for incremental refinement, such refinement is possible within our framework if the plan generates a solution before the deadline (because of changes in the deadline or tasks taking less time than expected).

Although approximate processing techniques are not new in computer science, our work develops a taxonomy of approximations, using an interpretation problem as an example domain. In interpretation problems, processing speedup can be achieved by reducing solution quality along one or more of the following dimensions: completeness (some solution aspects are ignored), precision (some solution parameters are not determined exactly), and certainty (some supporting or detracting evidence is not considered). For example, consider a vehicle-monitoring system that must track vehicles and supply the results to a vehicle coordinator. To ensure that the vehicles do not collide with buildings or each other, the vehicle coordinator supplies the vehicle-monitoring problem solver with the following objective:

> Within 10 seconds, supply the vehicle types, positions, and movement characteristics for as many vehicles moving through the environment as possible, giving preference to tracking vehicles of type v_1 and determining vehicle directions with high precision.

The best possible answer regardless of time constraints might be as follows:

> The following events and their corresponding features are certain: Vehicle type v_1 located at l_1, moving with speed s_2 and direction d_1. Vehicle type v_2 located at l_2, moving with speed s_1 and direction d_2. No other vehicles are present in the environment.

To meet the deadline, however, the problem solver might be unable to form the best answer and instead concentrates on generating an acceptable solution, as follows:

> From a cursory analysis, it is likely that there exists a vehicle of type v_1 located near l_1, moving with speed between s_0 and s_1 and in direction d_1. Other vehicles might be present.

This answer is acceptable because it is formed within the time constraints and meets the other criteria (tracks correct vehicle type and finds exact direction). However, it lacks precision (about location and speed of vehicle of type v_1), completeness (because the vehicle of type v_2 is missing), and certainty (because it is likely rather than certain). It is important to note that the different dimensions of solution quality are interdependent. For example, reducing the completeness of a solution often reduces its certainty as well: An incomplete solution leaves room for doubt about whether the unused information supports or refutes this solution.

Trade-offs in solution time versus quality should depend on the system's objectives and its current problem-solving state. For example, if the system's primary objective is to track large vehicles (type v_1), possibly because collisions involving large vehicles are costly, then an incomplete map that excludes other vehicles is almost as good as a complete map. If the current problem-solving state includes a cloud of noisy, ambiguous signals, then a bounded approximation for data might be appropriate. In short, when attacking problems to meet objectives and time constraints in a range of situations, the system needs an arsenal of different approximations. We have identified three major types of approximations: (1) approximate search strategies explore a smaller portion of the search space than would be the case during normal processing, (2) data approximations provide an abstract view of data resulting in a simpler space being searched, and (3) knowledge approximations simplify the knowledge being applied in the system so that the search space can be explored quickly.

We study these types of approximations in detail in the remainder of this article, describing specific techniques for approximations, preliminary experiments with some of these techniques, and a framework for selecting techniques for particular situations. Although approximate processing is

potentially useful in any system regardless of its problem-solving architecture, a blackboard architecture (Erman et al. 1980; Nii 1986) has inherent flexibility and can easily be extended to incorporate approximate processing. Therefore, we examine approximate processing in the context of a particular blackboard-based problem-solving system for vehicle monitoring.

The next section discusses the desired properties of approximate processing and gives an overview of approximate processing in our specific blackboard-based vehicle-monitoring system. Specific approximations for each of the three major types are then described in turn: approximate search strategies, data approximations, and knowledge approximations. For each approximation, we discuss the problem-solving situation for which the approximation is appropriate and its effect on the solution quality (completeness, precision, and certainty). Preliminary Experiments then describes some experiments with approximate processing in the Distributed Vehicle-Monitoring Testbed (DVMT) to better illustrate its utility. In A Framework for Approximate Processing, we outline how the different types of approximate processing can be incorporated into a framework for flexibly meeting real-time constraints. Finally, Summary and Future Research discusses our current and future research directions.

Approximate Processing and Vehicle Monitoring

Our approach to approximate processing assumes that the following are desirable properties of approximations.

1. The approximation should have a well-defined effect on the solution characteristics (time, precision, completeness, and certainty) so that the system can determine whether a given approximation might satisfy the objective. A well-defined approximation can be contrasted with approximations that have ill-defined effects, such as eliminating weak input signals or input signals above a certain frequency. The difficulty with these ill-defined approximations is that the quality of the solution does not degrade gracefully as the amount of data being eliminated increases. For example, it might be the case that the weak signal being eliminated is the only way to positively identify an important vehicle.

2. Exact and approximate processing should be well integrated, leading to a continuity of solution approximations. This property implies that incremental refinement for the approximation should be available. If the system is allowed to use more time than originally anticipated, it should be able to generate a proportionally better solution by refining an approximate solution. Knowledge processing activities should also be able to exploit input in which some data are the result of approximate processing, but other data are the result of exact processing.

3. Approximations of knowledge-application activities should be consistent with exact processing. Thus, weaker constraints exploited during approximate processing should not eliminate a result that is compatible with the stronger constraints used during regular processing. For example, if an approximate activity produces a range for a vehicle location, then precise processing should never result in locating the vehicle outside this range.

These properties make it more likely that approximations will lead to results consistent with exact processing whose quality and timeliness can reasonably be bounded. Our approach uses the well-defined behavior of approximations with these properties to predict how the approximations will influence the quality of solutions and the time needed to form them. In contrast, an ill-defined approximation is unpredictable: It might lead to a better-quality answer in the available time, but it might form a result that is inconsistent with exact processing or whose quality and timeliness do not meet the stated criterion. Although we feel predictability is an important character of an approximation, the overriding philosophy of our approach is to get the best answer given the time limitations. Thus, situations exist (for example, extremely tight time constraints) where the system might choose to abandon well-defined approximations that are unable to produce a solution with the appropriate level of quality within the available time and instead use ill-defined approximations. By doing so, the system risks forming erroneous solutions or no solutions at all, but because it predicted that well-defined approximations would not have worked anyway, the risk is justified.

Approximation is useful in blackboard-based interpretation problem-solving systems for tasks such as vehicle monitoring. These systems solve problems by selectively combining data into partial solutions in order to form partial solutions that explain more data by having either larger scopes (such as longer vehicle tracks) or higher abstraction levels (such as classifying a combination of individual signals as a single vehicle object). This combination process uses domain-specific constraints to form consistent interpretation alternatives and eliminate those which are inconsistent. Thus, problem solving can be viewed as a constraint-aggregation process that involves the creation of more encompassing interpretations and the elimination of alternatives which become inconsistent or highly unlikely as a result of an attempt to incorporate additional data. The system's objective is to form complete interpretations that account for all its data in terms of high-level events (such as a pattern of vehicles moving through the area) or non-events (such as echoes or spurious sensor noise).

Our example interpretation application, as mentioned earlier, is the vehicle-monitoring task as implemented in DVMT (Lesser and Corkill 1983). A problem solver in DVMT receives acoustically sensed data at discrete sensed times and applies simplified signal-processing knowledge to these data in order to identify, locate, and track patterns of vehicles moving through a two-dimensional space. A solution consumer (currently the user) specifies to DVMT the solution criteria, including deadlines and preferences for solution characteristics. The DVMT problem solver applies its knowledge sources to its data to extend and refine partial solutions until it forms a solution meeting the

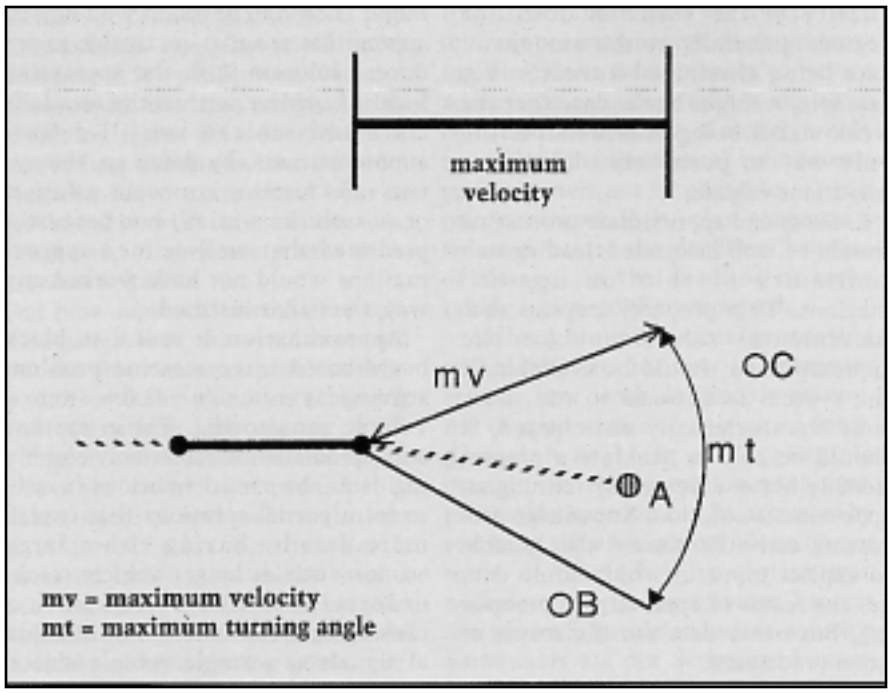

Figure 1. Eliminating Alternatives Based on Movement Constraints.

desired characteristics. A partial interpretation is represented as a hypothesis on the blackboard and is characterized by one or more time locations (where the vehicle was at the discrete sensed times), an event class (classifying the frequency or vehicle type), and a belief (the confidence in the accuracy of the hypothesis).

We provide a DVMT problem solver with a planner that uses an abstract view of the problem-solving state to plan sequences of actions for resolving uncertainty about the potential solutions to develop and for developing them (Durfee and Lesser 1986; Durfee and Lesser 1988). The abstract view is built by clustering related data into an abstraction hierarchy and allows the planner to recognize long-term problem-solving goals (to track some type of vehicle through a particular region). The planner finds relationships among these goals, such as competition when different goals involve common data, because for both goals to be true simultaneously, more than one vehicle must occupy a certain space at the same time.

The planner then develops a plan to satisfy each long-term goal: It sketches out a sequence of major plan steps, where each step achieves an intermediate goal of processing and integrating data for a certain sensed time into a developing partial solution. The planner then details specific short-term actions for the next intermediate goal. By interleaving planning and execution, it incrementally adds details to plans when needed. Through this interleaving, the planner remains responsive to unexpected situations that can arise during problem solving. The planner also predicts the time needs and result quality of its plans by forming predictions about the results of the major plan steps and roughly how long they will take, based on the results and time needs of similar steps in the past and on models of problem-solving actions (Durfee and Lesser 1988).

To illustrate vehicle monitoring, an example problem-solving situation is depicted in figure 1 where there are three alternative directions, labeled A, B, and C, to extend the track. However, only alternative A is consistent with vehicle movement constraints. Alternative B can be eliminated because it exceeds the maximum acceleration constraint (it turns too sharply), and point C is eliminated because to have gotten to this point, a vehicle would have exceeded the maximum velocity constraint. If no other tracks can be formed with data points B and C, these data will be interpreted as noise.

Solution quality depends on the application of knowledge to input data. Thus, a solution is complete to the degree that all input data have been incorporated into the solution or interpreted as noise and discarded. A solution is precise if its precision is no lower than the precision of input data, and the certainty of a solution is high to the degree that (1) substantial support exists for the solution (it is consistent with much data), (2) supporting data have high certainty, and (3) the solution is unique (the same data do not support other competing solutions).

These quality characteristics of a solution are affected by the number of alternative partial interpretations examined in problem solving and by how precisely the constraints applied in the incremental aggregation process mirror the domain constraints. Processing time is decreased to meet deadlines by reducing the number of alternatives examined and eliminating or weakening constraints (to reduce the time needed for testing whether the constraints hold), thereby decreasing solution quality.

The number of alternatives can be reduced by changing the search strategy so that a smaller portion of the search space is considered or developed. Another method for reducing the number of alternatives is to take an abstract view of data, where individual data points are clustered together, and this collection is processed as a whole instead.

Constraints can be weakened by simplifying knowledge in the system or not evaluating all the constraints that are applicable. Unfortunately, the weakening of constraints often has the effect of increasing the number of alternatives. For example, in DVMT, a speedup in vehicle tracking can be obtained by only verifying a new vehicle location against the maximum velocity constraint instead of doing a complex verification that considers maximum acceleration as well. Considering only velocity is faster but might let the system hypothesize invalid tracks: The vehicle could not have accelerated to the velocity that it would have needed to generate these data. Weakening constraints can potentially slow down problem solv-

ing because more hypotheses might be formed and pursued. To reduce the number of hypotheses in this situation, the weakening of constraints can be combined with some data abstraction mechanism to group hypotheses into a smaller number of larger entities.

Therefore, many different approximations are possible, each with advantages and disadvantages. In the next three sections, we describe possible approximations for each of the three major types.

Approximate Search Strategies

We consider two ways for limiting the number of alternative interpretations that are examined: limiting the input to knowledge-processing activities and limiting the output of activities. To minimize the impact on the solution quality, it should be highly certain that data being eliminated are inferior to data being retained. The following strategies use corroboration and competition as criteria for pruning inferior alternatives.

Eliminating Corroborating Support

Multiple independent sources of evidence for an event increase the certainty of the event. For example, multiple acoustic sources (for example, engine, fan) provide mutually supporting evidence about a vehicle's identity. If the system is faced with time constraints, it can save time by not processing all corroborating data and, thus, not evaluating all available constraints. However, note that this approximation is well-defined because only corroborating data are ignored. Unlike ill-defined approximations that ignore data based on their individual attributes (such as signal strength or frequency), this approximation ignores data that have already been classified to some extent (as corroborating).

The least relevant support—the support that contributes the least to the event identification—should be eliminated first. In the earlier example, if different vehicles are equipped with similar fans but have distinct engine sounds, then the acoustic signals corresponding to the fan should not be processed. The sooner the corroborating data are eliminated from further consideration during the problem-solving process, the more time that can be saved. Thus, not acquiring certain input data would be most beneficial, providing it can be determined a priori that these data will provide corroborating support. This approximation should be considered if the amount of supporting data is large, or the certainty of supporting data is high. The effect of eliminating corroborating support is reduced certainty of the solution in proportion to the amount, relevance, and individual certainties of eliminated support and the relationship among competing interpretations (that is, how well can the system differentiate vehicles based only on their engine sound?).

Some objects in an interpretation system can best be thought of not as single events but as a collection of events with common attributes. In this case, data can provide corroborating evidence for the common attributes but can also support different events. Eliminating this kind of support will affect not only certainty but possibly other solution characteristics as well. For example, consider a vehicle track hypothesis in DVMT supported by signals at different sensed times. The interpretation of the data at one sensed time can corroborate data at adjacent times if they arrive at consistent interpretations for common characteristics, such as the type of vehicle being tracked. Thus, the elimination of data at various sensed times can reduce certainty. Moreover, the elimination of such data also reduces the completeness of the solution by providing less information about consecutive vehicle locations.

It is sometimes possible to change the effect of the approximation on the solution characteristics. In the earlier example, the problem solver can increase completeness in the missing sensed times by reducing precision. It does this by using knowledge about likely vehicle movements (velocity and acceleration constraints) to roughly estimate where the vehicle was during these times. This change would be appropriate when the objective specifies that a complete solution is more important than a precise one. The situation is depicted in figure 2. Figure 2a shows the precise interpretation (a vehicle track with seven sensed times). Figure 2b is an incomplete solution (times 2, 4, and 6 are missing). Figure 2c is an imprecise solution, where a range of possible vehicle locations exists at times 2, 4, and 6.

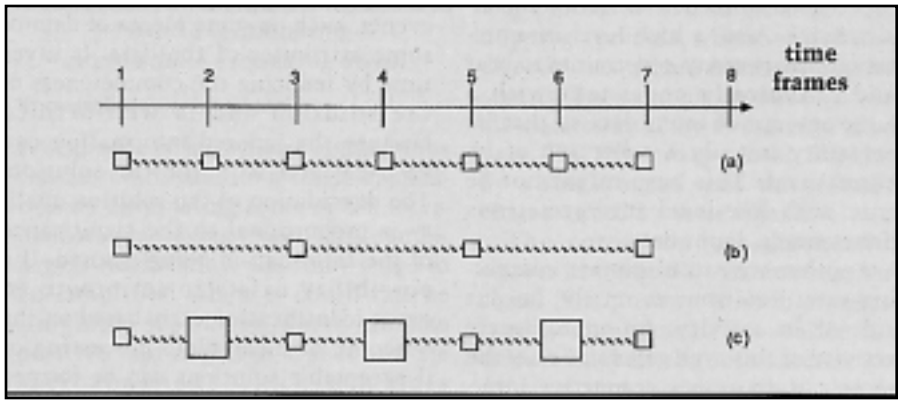

Figure 2. Some Effects of Approximation.

Eliminating Competing Interpretations

Interpretations compete when they support mutually exclusive events. An alternative interpretation can be eliminated if it has a lower certainty than the best competing interpretation after all data have been processed. To meet deadlines, however, a problem solver can eliminate competing interpretations before all the data have been processed. If based on its predictions, the problem solver believes that no amount of processing can make an alternative more certain than the best alternative, then it can eliminate the alternative. However, because predictions can be imperfect, the problem solver runs the risk of

missing better alternatives. To minimize this risk, it should only eliminate alternatives that are expected to have considerably lower certainties, where the minimum difference in certainties is a parameter (currently user supplied). By decreasing this parameter, the problem solver becomes more apt to decrease computation costs by avoiding alternatives, but it becomes less certain that its selected solution is correct. One way of viewing this search approximation is that the more conservative the planner is, the more breadth first the search is; the less conservative the planner is, the more depth first the search becomes.

Eliminating competing interpretations is a well-defined approximation even though it might look similar to the thresholding of input data (an ill-defined approximation). The approximations differ in two important ways: elimination is based on competition, and the certainty of high-level interpretations is highly correlated with their correctness. Competition is important because it establishes that only one of a group of interpretations can be true. This fact cannot be established for input data because further processing is needed to verify inconsistencies. Also, a high-level interpretation brings many constraints to bear and is typically consistent with a large amount of input data so that its certainty is truly a reflection of its correctness. This case might not be true with low-level interpretations (for example, input data).

Another way to eliminate competing interpretations is on the output side of an activity. An approximate activity of this type generates only the most certain among competing interpretations. The result is faster processing because of fewer alternatives, and the effect on the solution is in reduced certainty. This approximation stands in between the DVMT planner approximation described earlier and the input data thresholding: It is well-defined when used in the later stages of processing when high-level interpretations are formed. If low-level interpretations are eliminated, lowering the certainty threshold reduces the risk of eliminating correct interpretations but also reduces the speedup because potentially more alternatives are generated. Another way to lower this risk is to allow the activity to act as a "generator function" (Lesser and Erman 1977), initially creating only the most certain output alternatives but having the ability to generate more output if needed (for example, if lack of problem-solving progress is discovered). The disadvantage of this scheme is that the solution time becomes difficult to predict.

Data Approximations

Two ways exist to limit the number of processing alternatives by taking an abstract view of data: incomplete event characterization (some information in the data is ignored during processing) and clustering (processing a cluster of data as a single unit instead of processing each data unit in the cluster separately). Processing incomplete events is faster because the constraints concerning the ignored information do not need to be considered, and clustering speeds up processing by reducing the size of the search space.

Incomplete Event Processing

This type of approximate processing ignores some information about events, such as some pieces of data or some attributes of the data. It saves time by reducing the completeness of the solution and is well-defined because the ignored information can be used later to refine the solution. The degradation of the solution quality is proportional to the significance of the information being ignored. If a possibility exists to terminate an event identification early based on the outcome of incomplete processing or if acceptable solutions can be formed despite ignoring the information, then this approximation is appropriate.

For example, if the location and movement of a small vehicle is less important than the location and movement of a large one, then when faced with tight deadlines, it might be appropriate to first identify only the type of the vehicle based on its characteristic sounds, ignoring its location and movement. If the evidence indicates that the vehicle is small, no need exists for further processing (unless the movements of small and large vehicles are correlated, and thus, locating small vehicles helps in locating large vehicles). However, because large vehicles might also have characteristic movements that aid in their identification, the problem solver should not overlook movement in less time-constrained situations because considering both sounds and movements simultaneously during interpretation might be more efficient (generate fewer alternatives) than considering one first and then going back for the other.

As another example, when faced with tight deadlines the problem solver might determine that it cannot track a vehicle over all sensed times. Rather than eliminating corroborating data over the range of sensed times to reduce certainty (see Eliminating Corroborating Support), the problem solver might decide to develop an incomplete solution that only covers the most recent sensed times but with high certainty (because the data at these adjacent sensed times strongly corroborate each other, the corroboration of less recent data has little effect on certainty). If the purpose of vehicle monitoring is to guide vehicles so that they do not collide, timely and precise information about a vehicle's recent movements is more important than a complete map of a vehicle's movements over all sensed times. This approximation is well-defined, however, because if a problem solver finds that it has additional time, it can later develop the information for earlier sensed times.

Cluster Processing

Processing speedup is obtained by clustering data, extracting cluster characteristics, and performing further processing on these clusters instead of individual data units. This approach is appropriate when correct data are mixed with a large amount of correlated noise caused by data distortion. In this case, substantial processing time is spent in eliminating incorrect interpretations because correct data and noise have similar characteristics. By clustering the data and giving the cluster the combined characteristics of its data, this approximation is well-defined because the cluster satisfies any domain constraints

that any of its encompassed data can, so no possible solutions are overlooked.

For example, meteorological disturbances can cause dispersion of a single acoustic signal into a number of signals of the same frequency in nearby locations. These signals would all be weak, and the corresponding data would have low certainty. By clustering them, the problem solver reduces the number of individual hypotheses to reason about by treating the group as a single, less precise hypothesis. Clustering is, thus, beneficial when certainty is a more critical aspect of the solution than precision: A cluster has a higher certainty than its typical member (data point) because it is more likely to contain the correct data. The loss in precision is proportional to the size of the cluster, and the gain in certainty is dependent on individual certainties of data in the cluster.

Knowledge Approximations

We describe two types of knowledge approximation: approximations that work with data approximations and approximations that summarize several sources of knowledge into a single, less discriminating knowledge source.

Knowledge for Data Approximations

One kind of approximate knowledge needs to be introduced into the system as a direct result of data approximations. Data clusters require knowledge-processing activities that deal with the collection of data as a single unit. The approximate knowledge must, therefore, be capable of applying domain constraints on imprecise clusters of data. To ensure that the approximations are well-defined, clusters must satisfy constraints if any combination of the data they encompass satisfies these constraints. Knowledge approximations are also needed to overlook or weaken domain constraints when some information about events is being ignored. For example, to speed up processing by ignoring acceleration information, a problem solver needs approximate knowledge sources that form tracks without examining acceleration constraints.

Potential negative results of applying approximate knowledge to data approximations include an increase in alternative interpretations or increased uncertainty in solutions. Further data abstraction can reduce the number of alternatives or increase certainty but can also reduce precision. For example, if tracks are formed by ignoring acceleration, a larger number of such tracks might meet the weaker constraints. If these tracks have minor differences such as slightly different locations for each sensed time, they could be clustered together, but the result would be a less precise view of exactly where the vehicle was at each sensed time.

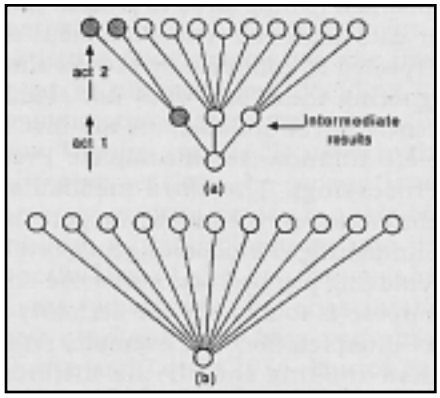

Figure 3. Eliminating Intermediate Processing Steps.

Knowledge Combinations

A sequence of knowledge applications can be combined into a single application by eliminating some of the intermediate processing steps. This combination might allow the knowledge to be simplified when it could not be simplified separately. There are also positive and negative aspects to the approximation, which combines sequential activities by eliminating intermediate steps. On the positive side, time is saved by not forming and posting the intermediate results and eliminating the overhead of scheduling the activity by eliminating the activity. However, when intermediate results are eliminated, opportunities for increasing certainty by corroborating support later during processing can be lost. A more subtle negative effect is in the loss of opportunism in processing, resulting in potentially more search. To understand this phenomenon, consider the example depicted in figure 3. A large "fanout" of interpretations takes place after two successive activities, act1 and act2, are performed. If two activities are combined, then all output alternatives have to be generated, as shown in figure 3b. However, opportunism is lost because if the two steps were scheduled independently, it might have been possible to pursue only certain intermediate interpretations[1] (for example, a shaded intermediate result in figure 3a). If the certainty of this intermediate result is high, exploring the other two (unshaded) intermediate results is unnecessary. In this case, the presence of the intermediate step has reduced the amount of search.

Approximations that combine processing steps are well-defined and are appropriate when the speedup is sufficient to balance the uncertainty coming from weaker constraints and the potential loss of opportunism. For an example from vehicle monitoring (figure 4), consider the two-step activity of identifying a vehicle: First, related acoustic signals are grouped; then, a vehicle is identified based on the characteristics of these groups of related acoustic signals. Contrast this two-step process with a single-step, "level-hopping" process where vehicles are directly recognized from the acoustic signals. For example, consider a vehicle characterized by two groups of signals. The first group can be identified by a single signal, and the second is characterized by three signals.

The corresponding grammar is shown in figure 4a. Assume that only signal A is present. If both groups are equally important in the vehicle identification, then the intermediate processing step of identifying groups gives more certainty to the resulting vehicle because signal A provides 50 percent confirmation of the vehicle's identification. This situation is schematically represented in figure 4b. The elimination of the grouping step corresponds to the simplified grammar shown in figure 4c. The uncertainty is higher because it is not known which group the signal belongs to.

Preliminary Experiments

To develop an initial understanding of how approximate processing affects problem solving, we implemented

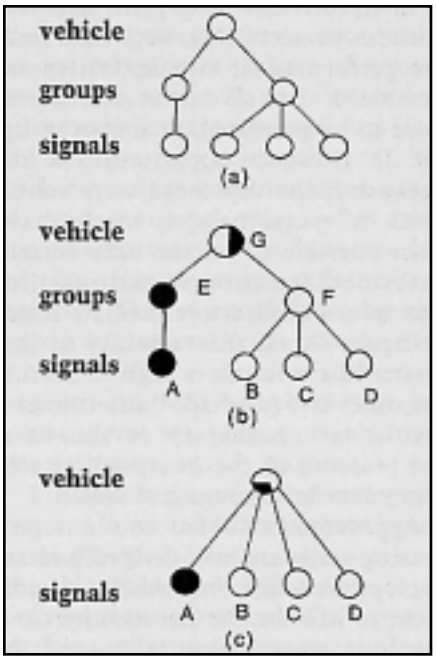

Figure 4. Level Hopping.

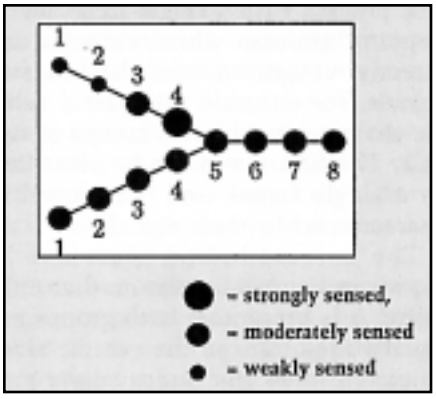

Figure 5.
The Experimental Problem Situation.

The problem-solving situation is shown. The possible tracks are indicated by connecting the related data points, and the time each data point was sensed is indicated. The size of a data point represents the strength of the detected signals.

three approximation mechanisms as part of DVMT. These mechanisms reduce problem-solving time to better meet deadlines in different ways. The first mechanism eliminates competing interpretations (see Eliminating Competing Interpretations); we studied how a different value for the minimum difference between expected certainties affects the time needed to solve the problem. The second mechanism generates incomplete solutions by incompletely processing the data (see Incomplete Event Processing): Because a vehicle's most recent movements are important for collision avoidance, the problem-solving plan is modified to drop steps to process earlier data until the remaining steps are expected to complete by the deadline. Ignoring these data does not significantly affect certainty in the incomplete solution (see Incomplete Event Processing). The third mechanism eliminates corroborating support (see Eliminating Corroborating Support) by removing planned actions whose only purpose is to increase the certainty in an interpretation. For example, rather than looking for all the different sounds a particular type of vehicle makes, the problem solver concludes (with lower certainty) that the vehicle is present based on a few sounds.

The user specifies the objectives to DVMT as a particular deadline for the solution and an ordering in which it should apply the approximation mechanisms. We illustrate the experimental results using the environment in figure 5. Two competing solutions exist in this environment: The vehicle can start either in the upper-left or lower-left corner. These alternatives compete because two vehicles cannot be present—they cannot be in the same place at the same time at sensed times 5-8. The track extending to the upper left involves weak and strong data but more weak than strong, and 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 data for the upper extension were perhaps the result of echoes in the environment.

The experimental results are summarized in table 1. For each experiment is shown the time when the solution was returned by the problem solver (where the execution of a single knowledge source takes one time unit), the user-supplied deadline given to the problem solver, the sensed times of the solution track, the belief of the solution track, and comments about the experiment.

When given a long time to solve the problem (E1), the problem solver does not need to use any approximate processing mechanisms. In particular, it can be conservative about eliminating competing alternative interpretations. In this case, the user-specified parameter is set so that the problem solver never trusts its predictions: Given time, the problem solver always explores alternatives to make sure that they do not yield better solutions than what it has formed already. Therefore, even though it generates the best solution at time 40 (as in E2), the problem solver examines the other solution and does not respond with an answer until time 57, when it has formed and discarded the inferior alternative. If only given 40 time units to solve the problem (E2), the problem solver cannot explore the competing solutions, and it uses its predictions to conclude that it has found the best solution. It returns this solution at time 40. Although both experiments find the best solution equally fast, E1 spends a lot of time and energy verifying that it was the best solution, whereas E2 predicts the upper track will be inferior and does not pursue it. The predictions in these experiments are sufficiently accurate so that E2's decision is correct. In other problem situations, such a decision might be premature—the predictions might underestimate the quality of as yet undeveloped results, and the best solution might be missed. How conservative the decisions should be depends on the problem situation and how much time is available.

A highly believed hypothesis spanning the entire track cannot be formed in less than 40 time units in this particular environment. Given a deadline of 36 time units (E3), the problem solver predicts there is insufficient time to form the best solution and, thus, revises its plan to meet the deadline. Told by the experimenter

Expt	STime	Deadline	Track	Belief	Comments
E1	57	80	1-8	*high*	Explores all solutions
E2	40	40	1-8	*high*	Stops with best predicted solution
E3	36	36	1-8	*mod*	Sacrifice certainty
E4	32	32	1-8	*low*	Sacrifice certainty
E5	30	30	2-8	*low*	Sacrifice certainty
E6	28	30	4-8	*high*	Sacrifice completeness

Legend

Expt: The experiment
STime: Time at which the best solution was found
Deadline: Time at which solution must be returned
Track: Times spanned by best solution track
Belief: Belief in best solution track

Table 1. Experiment Summary.

(the consumer of the solution) to prefer to sacrifice certainty in the solution, it removes enough actions which generate corroborating support so that it still forms a complete solution (covering all the sensed times) but only with a moderate belief. With only 32 time units to work with (E4), the system removes actions so that the solution covers all the sensed times but with low belief.

Given a deadline of time 30 (E5), the problem solver must employ a larger combination of mechanisms because it still expects to exceed the deadline after it has removed all the corroborating actions. Thus, it sacrifices some completeness by ignoring data for the earliest sensed time (time 1) and generates hypotheses with low belief spanning times 2-8. Finally, given the same deadline of time 30 but told by the experimenter to prefer to sacrifice completeness rather than certainty (E6), the problem solver no longer skips corroborating actions but instead reduces the scope of the solution to span times 4-8. Note that in these environments where each action (knowledge source) takes one time unit, the planner can drop just enough corroborating actions (single knowledge sources) to meet the deadline exactly (E3-E5). When it needs to get high belief and ignores data in the less important sensed times, the planner might not meet deadlines exactly (E6) because processing these data can require several knowledge sources.

Although these experiments represent only a preliminary examination of approximate processing, they do indicate how various mechanisms can allow a problem solver to meet deadlines. Among the important issues they have not explored are how the problem solver can flexibly choose approximations that are right for a particular situation and how the time spent in planning and deciding on approximations affects real-time activities. The first issue is discussed in the following section; the second issue remains an open research question, although initial explorations into the practicality of the control mechanisms (when their benefits justify their overhead) have been performed (Durfee and Lesser 1986; Durfee and Lesser 1988; Durfee 1987). This research indicates that because problem-solving activities (applying domain knowledge to some set of data) are typically expensive, the modest amount of overhead spent planning appropriate problem-solving actions is worthwhile.

A Framework for Approximate Processing

A framework for approximate processing must have mechanisms for recognizing that solution requirements cannot be met, choosing appropriate approximations, and performing approximate processing. We now describe how our framework deals with each of these issues.

Recognizing the Need for Approximate Processing

Recognition of the need for approximate processing comes naturally from planning the problem-solving activities for achieving system goals, estimating the time these activities take to achieve a solution, and monitoring the progress of these plans.[2]

If the estimated time to complete the plan is longer than the solution time required by the objective, then it becomes immediately clear that approximations are necessary. As processing goes on, plan predictions can be adjusted to more accurate values because additional information becomes available. The DVMT planner, for example, uses the time the system has taken to achieve goals in the past to predict the time needed to solve similar goals in the future, and as it pursues its plans, it has a larger selection of past goals to use when making predictions (Durfee and Lesser 1986; Durfee and Lesser 1988). Comparing the adjusted solution time with the given deadline can reveal that the current plan cannot be completed on time. Besides monitoring the plan, the planner needs to monitor the system state because slow progress on a plan can be the result of a changed situation. For example, the amount of input data can increase as a result of busier traffic in the area, or a processor malfunction can impose stronger resource constraints. In either case, in order to satisfy the objective, the planner must make a new, approximate plan that consists of faster or fewer activities and takes a shorter time to complete.

Choosing the Approximation Strategy

When the planner predicts that a solution satisfying the current objective cannot be generated within the allowable time, the system must choose a strategy for generating an approximate solution. One approach is to evaluate

every applicable approximation and then choose an approximation that gives the best quality within the allowable time. The evaluation involves forming a plan with approximate activities (described in Performing Approximate Processing) and allowing the planner to estimate the time and quality of the solution based on its models of problem-solving actions and on past experience. This approach for choosing approximations is costly for two reasons: Evaluating an approximation is a sophisticated and time-consuming process because it can require complete replanning, and the number of applicable approximation alternatives is large in any given situation. Complete evaluation of all approximations would consume a large amount of resources, canceling any speedup that could be obtained by approximate processing. Additionally, the search for the optimal approximation might not be appropriate because of the inherent unreliability of the prediction used by the planner to assess the effects of a plan and also dynamic and unpredictable changes to the problem situation. Thus, a heuristic search for a nearly optimal "satisficing" approximation is appropriate.

The first step in our approach is to evaluate only a small fraction of applicable approximations that are most likely to be relevant in the current state of processing. This procedure is accomplished by introducing approximation rules: heuristic situation-action rules that link the state of processing with approximate processing strategies. They state the situation preference for approximations by specifying approximations that are most appropriate in a given situation. This simple approach is used to limit the number of approximation alternatives by generating a relatively small number of approximation candidates. For example, a large amount of data about a single event is a situation in which the "incomplete event processing" approximation should be tried. This heuristic can be formulated as a rule:

> If there is a large amount of data about an event then try the "incomplete event processing" approximation.

An approximation suggested by an approximation rule can be eliminated on the basis of unsatisfied preconditions. Preconditions state the conditions under which the approximation is applicable. For example, the incomplete event processing approximation that ignores less relevant data based on recency of arrival might have preconditions specifying sensed times eligible to be ignored. If no data exist for these times, the approximation must be ruled out. The reason for separating these two filtering steps (candidate generation and precondition testing) is efficiency. The rules provide a fast and inexpensive focusing mechanism because situations are based on global features of the state of processing, whereas precondition tests are based on detailed data attributes.

Another method we use to limit the search for approximations is a simple model of quality loss expected from the approximation. Quality loss represents an estimate for degradation in solution quality (certainty, precision, and completeness) brought about by the approximation. In the incomplete event processing example, the simplest model of expected loss would be to make quality loss proportional to the number of sensed times being ignored. The expected quality loss provides a simple and computationally inexpensive model that is used to control the much more expensive evaluation process. Candidate approximations generated by approximation rules are ordered by increasing losses before they are evaluated by the planner. By controlling the order in which approximations are evaluated, the quality loss controls the quality of the resulting approximation. If the quality loss is an accurate reflection of the solution degradation, then the first approximation that satisfies the objective is also the best one, and no other approximations need to be evaluated. Using the first successful approximation assumes that in most cases a single approximation or combined group of approximations coming from a single situation-action rule leads to a better solution than combinations of approximations. Note, however, that the optimality of the approximation is not greatly affected by possible errors in the loss model, because the loss is used only to produce the ordering, and approximations are selected based on a detailed evaluation which relies on a much more accurate model of processing.

Finally, approximations can be ruled out before evaluation if the speedup expected from approximate processing is insufficient (as specified by the objective). The speedup is similar to quality loss in that its accuracy is not critical (as long as the speedup estimate is conservative enough not to rule out any sufficient approximations) because it is not used to select approximations. A good speedup model does, however, improve the efficiency of the search by avoiding the detailed evaluation of approximations whose effect is in the wrong ballpark in terms of processing time. Typically, the speedup and the quality loss depend on the same factors, and thus, their magnitudes are related. This relationship holds in our incomplete event processing approximation example, where the speedup is greater when additional sensed times are ignored.

Before an approximation can be evaluated or applied, the planner must know how the approximation is performed; that is, how new processing plans are constructed. We call the information that specifies how an approximation is applied the *value of the approximation*. For the approximation that ignores less recent data in DVMT, approximate processing entails ignoring intermediate processing goals corresponding to less recent sensed times.

All information needed to select, evaluate, and apply an approximation is contained in a record that we call the *approximation domain*. Sample information in the approximation domain for DVMT (taken from the incomplete event processing approximation example) is shown in figure 6. This approximation ignores less recent data (data for sensed times < t_n). Preconditions also specify that data on the vehicle level are not ignored (level < vehicle). This precondition is introduced because the speedup that could be achieved by ignoring vehicle data is minimal (little processing is left to do), and much processing that has already been done

would be wasted.

The following steps specify how the planner selects approximations:

1. Use approximation rules to generate candidate approximations.

2. Use preconditions to filter out nonapplicable approximations.

3. Find the least costly approximation with sufficient speedup to meet the specified deadline. If no simple approximation provides a desired speedup, consider combined approximations by adding their respective costs and speedups. The cost is calculated as the ratio of quality loss to speedup.

4. If there is no approximation or combination of approximations for which the total speedup is sufficient or if all applicable approximations have already been passed to step 5, then exit with failure (or try ill-defined approximations); otherwise, pass this approximation to step 5.

5. Use the value of the approximation to evaluate the approximation found in step 3.

6. If the evaluated approximation satisfies the objective, then exit with success; otherwise, go to step 3.

Figure 7 shows how this selection process fits into the basic DVMT planning loop. The selection of approximations is the elaboration of the box labeled "select best approximation."

Incomplete Event Processing

preconditions	value	quality loss	speedup
time > *n* level < vehicle	ignore goals for time = $t_1 \ldots t_{n-1}$	2 per sensed time	2 per sensed time

Figure 6. Incomplete Event Processing Approximation.

Performing Approximate Processing

A change from exact to approximate processing involves modifications of the search space, problem-solving plans, and the objective and low-level system goals (the last two define characteristics of acceptable results at a global and local level, respectively). In a blackboard-based problem solver, the representation of the search space corresponds to dimensions of the blackboard, such as abstraction level (signal, group, vehicle), the types of events to be recognized at each level, and the time and location range of each event. An example of an approximation that involves search space modification is cluster processing, which can combine hypotheses with identical time and location characteristics regardless of their level on the blackboard. For this approximation, the search space is modified by collapsing the abstraction levels into a single level.

Modifications of problem-solving plans include a choice of which of the alternative plans to follow, the elimination of plan steps, the substitution of regular activities in the plan by approximate activities, and the insertion of special activities into the plan. In Preliminary Experiments is a description of specific plan modifications based on certain approximations. In addition, some general mechanisms are required to initiate approximate processing regardless of the type of approximation. One such mechanism is the ability to redefine low-level system goals (Corkill, Lesser and Hudlická 1982). When processing is changed from exact to approximate, old goals (corresponding to exact processing) should be deactivated and replaced by new goals based on a different goal template. A *goal template* specifies the desired range of attributes in which results should fall—and the desired precision and certainty of the results—as a function of data attributes. Each approximation has a characteristic goal template. When clustering a cloud of data, for example, the location range of the clustering goal template specifies the area where noise was detected as the desired location range, and the location precision is the clustering metric.

Another mechanism needed to implement approximate processing is a generalized view of data. In DVMT, this data structure is called an *approximate hypothesis.* An approximate hypothesis has a range of values for event attributes and, thus, represents a generalization of the regular hypothesis. An approximate hypothesis provides an appropriate representation for data used both in clustering and incomplete event processing. For example, a location cluster is represented by an approximate hypothesis whose location range includes the locations of all data being clustered. If a location range is undefined (or, equivalently, includes all values possible in the domain), then the corresponding approximate hypothesis is a result of incomplete (type only) processing.

To enable processing of approximate hypotheses, an approximate version of every knowledge source needs to be added to the system. Approximate knowledge sources are generalized knowledge source modules, which are able to deal with data that have different levels of precision. Thus, processing of data objects with different precisions could be freely intermixed, and the detailed constraints of precise results can be exploited when they are available.

As part of DVMT, we have developed several of the approximations discussed in Approximate Search Strategies, Data Approximations, and Knowledge Approximations, as described in the following subsections.

Eliminating Corroborating Support-Multiple Types. This approximation was illustrated in the experiments (see Preliminary Experiments). When the planner discovers multiple activities which can contribute to satisfying the same goal, it eliminates enough of them from the plan so that it meets deadlines. Thus, if a vehicle is characterized by two harmonic groups, and data for both are available, data corresponding to one group are ignored.

Eliminating Corroborating Support-Skipping Sensed Times. General tracking knowledge sources take time resolution as a parameter (for example, exact tracking has a time resolution of 1, tracks are formed with data at every sensed time, but skipping every other sensed time would have a time resolution of 2). Plan modification for this approximation involves

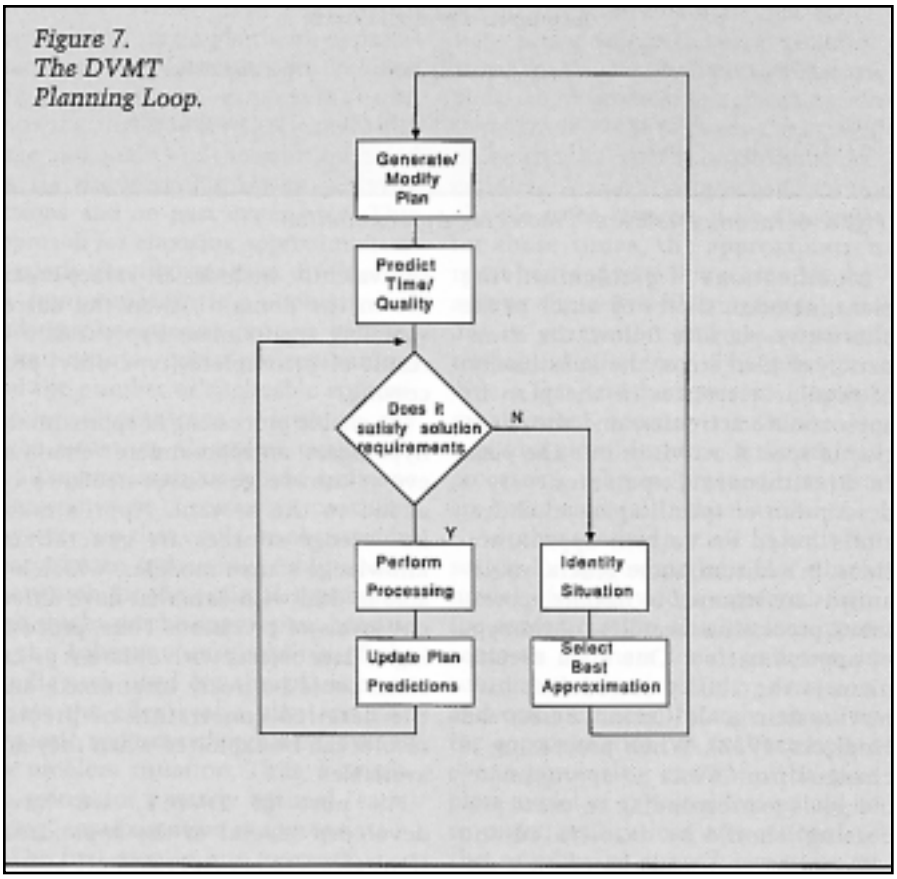

Figure 7. The DVMT Planning Loop.

modifying the resolution parameter of tracking knowledge sources. Processing is modified by deactivating goals corresponding to eliminated data and forming a new tracking goal template with the appropriate time resolution.

Eliminating Competing Interpretations. Before the system returns a solution, it estimates the certainty of potential competing solutions resulting from plans that have not yet been completed. If these solutions are estimated to have much lower certainty than the currently proposed solution the corresponding plans can be ignored (see Preliminary Experiments).

Incomplete Event Processing—Ignoring Attributes. We introduce approximate processing activities that consider a restricted set of data attributes. In the incomplete processing plan, regular activities are replaced by these incomplete activities. An example of an incomplete knowledge source in DVMT is class synthesis, which combines hypotheses on one abstraction level to obtain hypotheses on a higher abstraction level by considering only compatibility of event classes and ignoring their location. The goal templates used for this incomplete event processing specify that the ignored attribute can have a range of all values possible in the domain.

Incomplete Event Processing—Ignoring Least Recent Data. Plan modification involves ignoring the activities which process the least recent data so that less complete solutions are formed (see Preliminary Experiments).

Cluster Processing. The clustering function is performed by the new combine knowledge source. Combine is triggered by a new goal template whose measure of precision is the clustering metric specified in the approximation domain. It forms new data units consisting of the data being clustered and represented by approximate hypotheses. When cluster processing is decided on, the planner replaces regular knowledge sources (operating on single data units in a cluster) with one knowledge source (operating on the cluster).

Knowledge Simplification—Level Hopping. Level-hopping knowledge sources combine the knowledge contained in several regular knowledge sources. For example, a level-hopping knowledge source that identifies vehicles based on acoustic signals combines the knowledge about how signals should be grouped and how groups of signals indicate vehicles. When the situation for which this approximation is appropriate is discovered, a sequence of regular activities in the plan is replaced by the level-hopping activity.

Summary and Future Research

We have laid out an approach for real-time problem solving. It is based on the AI problem solver having a sophisticated control component that can plan out its problem-solving activities. Once a system is actively planning the use of its resources, then time is but one such resource. The key aspects of this approach follow.

1. The criterion for successful real-time control behavior for AI systems should be to try to develop the best solution to the overall problem that satisfies the time constraints.

2. A real-time AI problem solver must be able to reason about its criteria for acceptable solutions, its problem-solving state, and its plans for achieving an acceptable solution.

3. A problem solver must estimate the time it needs to complete its plans. Because these estimates are uncertain (problem solving is to some extent unpredictable), the problem solver might fail to meet a deadline exactly.

4. If the best (most complete, precise, and certain) solution is not obtainable within the available time, the problem solver should resort to approximate processing strategies that trade the quality of the solution for speedup. Three classes of approximate processing strategies are detailed: approximate search, data approximations, and knowledge approximations. Each of these contributes to developing a smaller or simpler search space and faster ways of searching the space.

We believe that the additional flexibility provided by approximate processing can be exploited by the problem solver to achieve other ends

besides meeting deadlines. For example, when faced with highly uncertain situations, the problem solver could employ approximate processing to get a handle on these situations without investing large amounts of effort. The problem solver, thus, becomes versatile as a result of its ability to selectively use a wider range of problem-solving techniques.

An important assumption about this framework for real-time control is that it is possible to make a reasonably accurate estimate of the time to carry out the steps of the plan and the quality of the expected solution. The better the prediction, the more quickly it can be recognized that deadlines cannot be met; thus, increased flexibility is given to the system to find the best-quality solution within the remaining time. The work of Durfee and Lesser (1986) and Pavlin (1983) shows how to make reasonable predictions in DVMT, and we must develop corresponding mechanisms for other types of AI systems. We should also explore other important issues such as interrupting knowledge-source processing; allowing for ongoing, low-level sensor monitoring and effector control; and predicting and managing the cost of control decisions (Fehling, Joerger, and Sagalowitz 1986; Pardee and Hayes-Roth 1987).

We considered real-time control in the context of a blackboard-based interpretation system that plans out its problem-solving activities. In this context, we discussed how deadlines can be met by reducing the solution's completeness, precision, certainty, or any combination of these and outlined approximate processing techniques for speeding up problem solving at the cost of solution quality. Our preliminary experiments show that these techniques allow a problem solver to flexibly meet deadlines. Finally, we outlined a computationally efficient framework for choosing and pursuing suitable approximations in a blackboard-based interpretation system.

Acknowledgments

This work was first initiated in 1986 as part of a Masters project by Ravi Bhargava under the supervision of Victor Lesser and Daniel Corkill. The perceptive and probing comments on earlier drafts of this paper by Norman Carver, Paul Cohen, Daniel Corkill, and Jack Stankovic were extremely helpful. Michele Roberts assisted in preparing the manuscript and figures.

This research was supported by the Office of Naval Research under a University Research Initiative Grant, Contract # N00014-86-K-0764, and the Defense Advanced Research Projects Agency monitored by the Office of Naval Research under Contract # N00014-79-C-0439.

References

Corkill, D. D.; Lesser, V. R.; and Hudlická, E. 1982. Unifying Data-Directed and Goal-Directed Control: An Example and Experiments. In Proceedings of the Second National Conference on Artificial Intelligence, 143-147. Los Altos, Calif.: Morgan Kaufmann.

Durfee, E. H. 1987. A Unified Approach to Dynamic Coordination: Planning Actions and Interactions in a Distributed Problem Solving Network. Ph.D. diss., Dept. of Computer and Information Science, Univ. of Massachusetts.

Durfee, E. H., and Lesser, V. R. 1988. Incremental Planning to Control a Time-Constrained, Blackboard-Based Problem Solver. *IEEE Transactions on Aerospace and Electronic Systems*. Forthcoming. (Also Technical Report 87-07, Dept. of Computer and Information Science, Univ. of Massachusetts.)

Durfee, E. H., and Lesser, V. R. 1986. Incremental Planning to Control a Blackboard-Based Problem Solver. In Proceedings of the Fifth National Conference on Artificial Intelligence, 58-64. Los Altos, Calif.: Morgan Kaufmann.

Erman, L. D.; Hayes-Roth, F.; Lesser, V. R.; and Reddy, D. R. 1980. The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys* 12: 213-253.

Fehling, M. R.; Joerger, K.; and Sagalowitz, D. 1986. Knowledge Systems for Process Management. In Proceedings of the Instrument Society of America-86 Conference, 41(3): 1509-1526. Research Triangle Park, N.C.: Instrument Society of America.

Hayes-Roth, B. 1987a. A Multi-Processor Interrupt-Driven Architecture for Adaptive Intelligent Control, Technical Report, KSL 87-31, Dept. of Computer Science, Stanford Univ.

Hayes-Roth, B. 1987b. Dynamic Control Planning in Adaptive Intelligent Systems. In Proceedings of the DARPA Knowledge-Based Planning Workshop, 41-47.

Hudlická E., and Lesser, V. R. 1984. Meta-Level Control through Fault Detection and Diagnosis. In Proceedings of the Fourth National Conference on Artificial Intelligence, 153-161. Los Altos, Calif.: Morgan Kaufmann.

Lesser, V. R., and Corkill, D. D. 1983. The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks. *AI Magazine* 4(3): 15-33.

Lesser, V. R., and Erman, L. D. 1977. A Retrospective View of the Hearsay-II Architecture. In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 790-800. Los Altos, Calif.: Morgan Kaufmann.

March, J. G., and Simon, H. A. 1958. *ORGANIZATIONS*. New York: Wiley.

Nii, H. P. 1986. Blackboard Systems: Part One. *AI Magazine* 7(2): 38-64. Blackboard Systems: Part Two. *AI Magazine* 7(3): 82-106.

Pardee, W. J., and Hayes-Roth, B. 1987. Intelligent Real-Time Control of Material Processing, Technical Report #1, Rockwell International Science Center, Palo Alto Laboratory.

Pavlin, J. 1983. Predicting the Performance of Distributed Knowledge-Based Systems: A Modeling Approach. In Proceedings of the Third National Conference on Artificial Intelligence, 314-319. Los Altos, Calif.: Morgan Kaufmann.

Ramamritham, K., and Stankovic, J. A. 1984. Dynamic Task Scheduling in Distributed Hard Real-Time Systems. *IEEE Software* 1(3):65-75.

Simon, H. A. 1969. *The Sciences of the Artificial*. Cambridge, Mass.: MIT Press.

Stankovic, J. A.; Ramamritham, K.; and Cheng, S. 1985. Evaluation of a Flexible Task Scheduling Algorithm for Distributed Hard Real-Time Systems. *IEEE Transactions on Computers* C-34(12): 1130-1143.

Notes

1. It is possible to have opportunism within the combined knowledge source, but this opportunism is more difficult to do correctly because the decision to drop processing of intermediate alternatives is only a local decision within the context of a single knowledge source application.

2. An alternative way to recognize the need for approximate processing is to have some simple measure to indicate appropriate problem-solving progress. For additional details of this approach, see work by Hudlická and Lesser (1984). This simpler but less precise method could also be used in conjunction with the techniques developed in Choosing the Approximate Strategy except that the final step, which is the verification of the new objective and control strategy by constructing a detailed plan, would be eliminated.