Approximate Processing in Real-Time Problem Solving

Victor R. Lesser, Jasmina Pavlin, and Edmund H. Durfee

COINS Technical Report 87-126

December 1987

Abstract

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 action by forming lower quality solutions within time constraints. This may 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 on 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.

Victor Lesser and Edmund Durfee are with the Department of Computer and Information Science at the University of Massachusetts in Amherst. Jasmina Pavlin is with the Intelligent Sciences Institute (ISI) at the University of Southern California.

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

1 Introduction

Artificial intelligence promises to solve many knowledge-intensive problems which cannot be solved in more 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 [6,8,19]. 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 that 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 [16,18]. If there are more tasks 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 may 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 may be able to form rough estimates about the time needed to perform this heuristic search, a problem solver cannot determine the time needs *precisely* beforehand because the information that influences heuristic decisions may change as search progresses.[1] Moreover, solving some subproblems can affect the importance and time needs of tasks to solve related subproblems. 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 I/O and interrupt handling). Our goal is to develop problem solvers that, when they cannot find optimal solutions in time, will adaptively generate the *most acceptable solutions* that meet deadlines and the users' needs. Our criterion for problem-solving success is thus very similar

[1] In an interpretation system [11], 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 the idea of "satisficing" developed by March and Simon [13,17]. In addition, because a real-time problem solver uses imprecise predictions about its activities, it cannot *guarantee* that it will meet deadlines—just that it will meet deadlines if its predictions do not underestimate the time needs of problem-solving tasks.[2]

This new approach to real-time control requires that the system reasons 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 will be further illustrated below). 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 the activities will take to complete, 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* which trades off the quality of a solution against the time to derive it, where the dimensions of solution quality are domain specific.[3]

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 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 those changes can be costly. In this paper, we therefore concentrate on incorporating approximate processing into plans to efficiently construct acceptable solutions within 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 (due to 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 interpretation problem solving as an exam-

[2]The likelihood of underestimating the time needs of problem-solving tasks can be reduced by forcing the problem solver to leave extra time "just in case", but this may cause it to generate poorer solutions than it had to in its haste. In any realistic scenario, the problem solver will therefore run some risk of exceeding deadlines. In this paper, we do not address issues in recovery from such situations.

[3]In medical diagnosis, for example, an important factor that determines the quality of diagnosis is discomfort caused the patient as a result of diagnostic procedures.

ple 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 which must track vehicles and supply the results to a vehicle coordinator. To insure that vehicles do not collide with buildings or each other, the vehicle coordinator supplies the vehicle monitoring problem solver with the 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:

> The following events and their corresponding features are certain: Vehicle type v_1 located at l_1, moving with speed s_1 and direction d_1. Vehicle type v_2 located at l_2, moving with speed s_1 and direction d_2. There are no other vehicles 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:

> 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 may be present.

This answer is acceptable since it is formed within 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 (since vehicle of type v_2 is missing) and certainty (since it is likely rather than certain). It is also important to note that the different dimensions of solution quality are *interdependent*. For example, reducing the completeness of a solution often reduces its uncertainty as well: an incomplete solution leaves room for doubt about whether the unused information supports or refutes that solution.

Tradeoffs 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 more 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 may 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:

Approximate search strategies explore a smaller portion of the search space than would be the case during normal processing.

Data approximations provide an abstract view of data resulting in a simpler space being searched.

Knowledge approximations simplify the knowledge being applied in the system so that search space can be explored more quickly.

We study these types of approximations in detail in the remainder of this paper, 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 [5,12] has inherent flexibility and can easily be extended to incorporate approximate processing. We therefore motivate our discussion using a particular blackboard-based problem-solving system for vehicle monitoring.

The next section discusses the desired properties of approximate processing and gives and 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: Section 3 covers approximate search strategies, Section 4 covers data approximations, and Section 5 covers 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). Section 6 then describes some preliminary experiments with approximate processing in the DVMT to better illustrate its utility. In Section 7, we outline how the different types of approximate processing can be incorporated into a framework for flexibly meeting real-time constraints. Finally, Section 8 discusses our current and future research directions.

2 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 would satisfy the objective. This can be contrasted with approximations that have less well-defined effects such as eliminating weak input signals or input signals above a certain frequency. The difficulty with these two

approximations is that the quality of the solution does not degrade gracefully as the amount of data being eliminated increases. For example, it may 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 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 while other data are the result of exact processing.

3. *Approximations made to 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 more 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 be reasonably 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-behaved approximation is unpredictable: it may lead to a better quality answer in the available time, but it may form a result not consistent with exact processing or whose quality and timeliness does not meet the stated criterion. Although we feel predictability is a very important character of an approximation, the overriding philosophy of our approach is to get the "best" answer given time limitations. Thus, there are situations (e.g., extremely tight time constraints) where the system may choose to abandon well-behaved approximations that are unable to produce a solution with the appropriate level of quality within available time, and instead use ill-behaved approximations. By doing so it risks forming erroneous solutions or no solutions at all, but since it predicted that well-behaved 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 either by having 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 that are inconsistent. Thus, problem solving can be viewed as a constraint aggregation process that involves creation of more encompassing interpretations and elimination of alternatives that 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 of 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 before, is the vehicle monitoring task as implemented in the Distributed Vehicle Monitoring Testbed (DVMT) [11]. A problem solver in the DVMT receives acoustically-sensed data at discrete *sensed times*, and applies simplified signal processing knowledge to that 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 the DVMT the solution criteria, including deadlines and preferences for solution characteristics. The DVMT problem solver applies its knowledge sources (KSs) to its data to extend and refine partial solutions until it forms a solution meeting the 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), by an event class (classifying the frequency or vehicle type) and by a belief (the confidence in the accuracy of the hypothesis).

We provide a 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 [2,3]. The abstract view is built by clustering related data into an *abstraction hierarchy*, and it 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 occupied 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. It 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 [3].

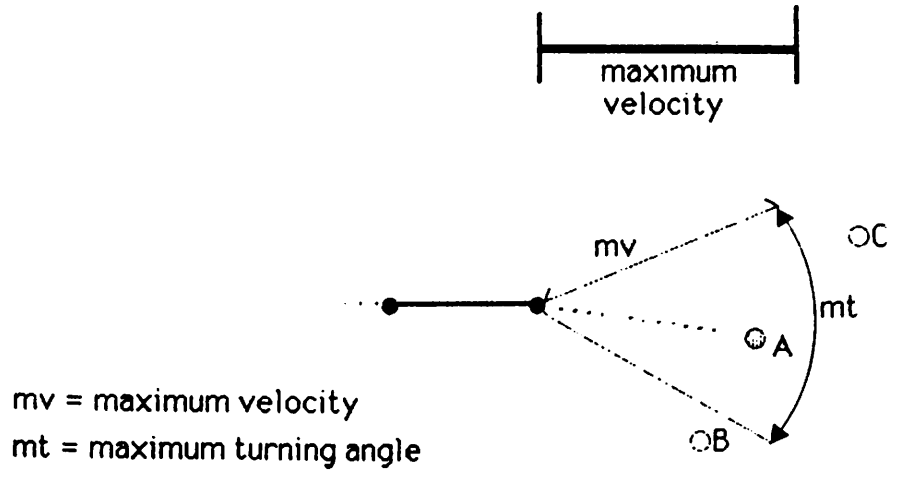


Figure 1: Eliminating Alternatives Based on Movement Constraints.

To illustrate vehicle monitoring, an example problem-solving situation is depicted in Figure 1, where there are three alternative directions to extend the track, labeled A, B and C. However, only alternative A is consistent with vehicle movement constraints. Alternative B can be eliminated since it exceeds the *maximum acceleration* constraint (it turns too sharply), while point C is eliminated because to have gotten there 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, while the *certainty* of a solution is high to the degree that:

- there is substantial support for the solution (it is consistent with much data);
- supporting data have high certainty;
- and 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 by 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 by taking 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 through the simplification of knowledge in the system or by 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 the 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 more complex verification that considers maximum acceleration as well. Considering only velocity is faster, but may let the system hypothesize invalid tracks: the vehicle could not have accelerated to the velocity that it would have needed to have generated that data. This can potentially slow down problem solving because more hypotheses may be formed and pursued due to weakening constraints. To reduce the number of hypotheses in this situation, the weakening of constraints may 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.

3 Approximate Search Strategies

We consider two ways for limiting the number of alternative interpretations that are examined: limiting inputs to knowledge processing activities and limiting outputs 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.

3.1 Eliminating Corroborating Support

Multiple independent sources of evidence for an event increase the certainty of that event. For example, multiple acoustic sources (e.g., 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 is a well-behaved approximation because only corroborating data is ignored. Unlike ill-behaved approximations that ignore data based on their individual attributes (such as signal strength or frequency), this approximation ignores data that has already be classified to some extent (as corroborating).

The least relevant support—the support that contributes least to the event identification—should be eliminated first. In the above 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 corroborating data are eliminated from further consideration during the problem-solving process, the more time could be saved. Thus, not acquiring certain input data would be most beneficial, providing that 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 certainty of supporting data is high. The effect of the elimination of 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 (i.e., 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 the 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 characteristics they have in common, 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 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 example above, 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 those 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 7 sensed times). Figure 2b is an incomplete solution (times 2, 4, and 6 are missing). Figure 2c is an imprecise solution, where there is a range of possible vehicle locations at times 2, 4 and 6.

3.2 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 may 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

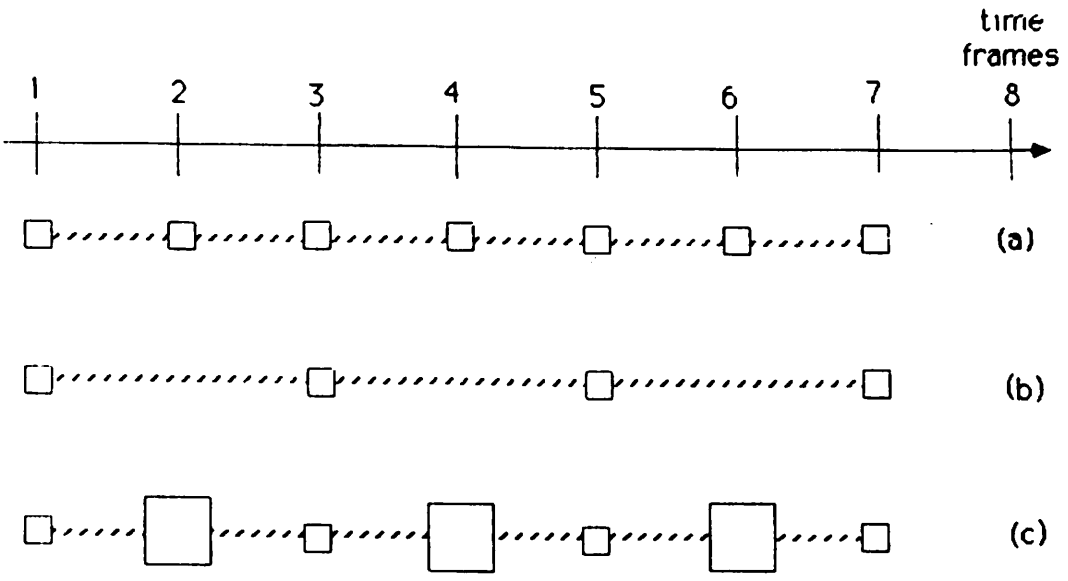


Figure 2: Some Effects of Approximation.

make an alternative more certain than the best alternative, then it can eliminate that alternative. However, because predictions may be imperfect, the problem solver runs the risk of missing better alternatives. To avoid this, 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, while the less conservative the planner is, the more depth-first the search becomes.

Eliminating competing interpretations is a well-behaved approximation even though it may look similar to thresholding of input data (an ill-behaved approximation). The approximations differ in two important ways: elimination is based on competition, and certainty of high-level interpretations is highly correlated with their correctness. Competition is important since it establishes that only one of a group of interpretations can be true. This fact cannot be established for input data, since 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 may not be the case with low-level interpretations (e.g, 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 above and input data thresholding: it is well-behaved when used in 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, since potentially more alternatives are generated. Another way to lower this risk is to allow the activity to act as a "generator function" [10], initially creating only the most certain output alternatives, but having the ability to generate more outputs if needed (e.g., if lack of problem-solving progress is discovered.) The disadvantage of this scheme is that the solution time becomes more difficult to predict.

4 Data Approximations

There are two ways 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 of incomplete events is faster because the constraints concerning the ignored information do not need to be considered, while clustering speeds up processing by reducing the size of the search space.

4.1 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-behaved since the ignored information can later be used to refine the solution. The degradation of the solution quality is proportional to the significance of the information being ignored. If there is a possibility 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 is an appropriate approximation.

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 may 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, there is no need 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 since considering both sounds and movements simultaneously during interpretation might be more efficient (generate fewer alternatives) then considering one first and then going back for the other.

As another example, when faced with tight deadlines the problem solver may 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 Section 3.1), the problem solver might decide to develop an incomplete solution that only covers the most recent sensed times but with high certainty (since 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 are more important than a complete map of a vehicles movements over all sensed times. This approximation is well-behaved, however, since if it finds that it has more time a problem solver can later develop the information for earlier sensed times.

4.2 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 since 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-behaved since the cluster will satisfy any domain constraints that any of its encompassed data will, so no possible solutions will be 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 together, 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.

5 Knowledge Approximations

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

5.1 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 which 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 insure that the approximations are well-behaved, clusters must satisfy constraints if any combination of the data they encompass satisfy those 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 more uncertainty in solutions. Further data abstraction can reduce the number of alternatives or increase certainty, but will 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.

5.2 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 they 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 by 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 may be lost. A more subtle negative effect is in loss of opportunism in processing, resulting in potentially more search. To understand this phenomenon, consider the example depicted in Figure 3. There is a large "fanout" of interpretations after two successive activities, act_1 and act_2, are performed. If two activities are combined, then all output alternatives have to be generated, as shown in Figure 3b. However, opportunism is lost since if the two steps were scheduled independently, it may have been possible to pursue only certain

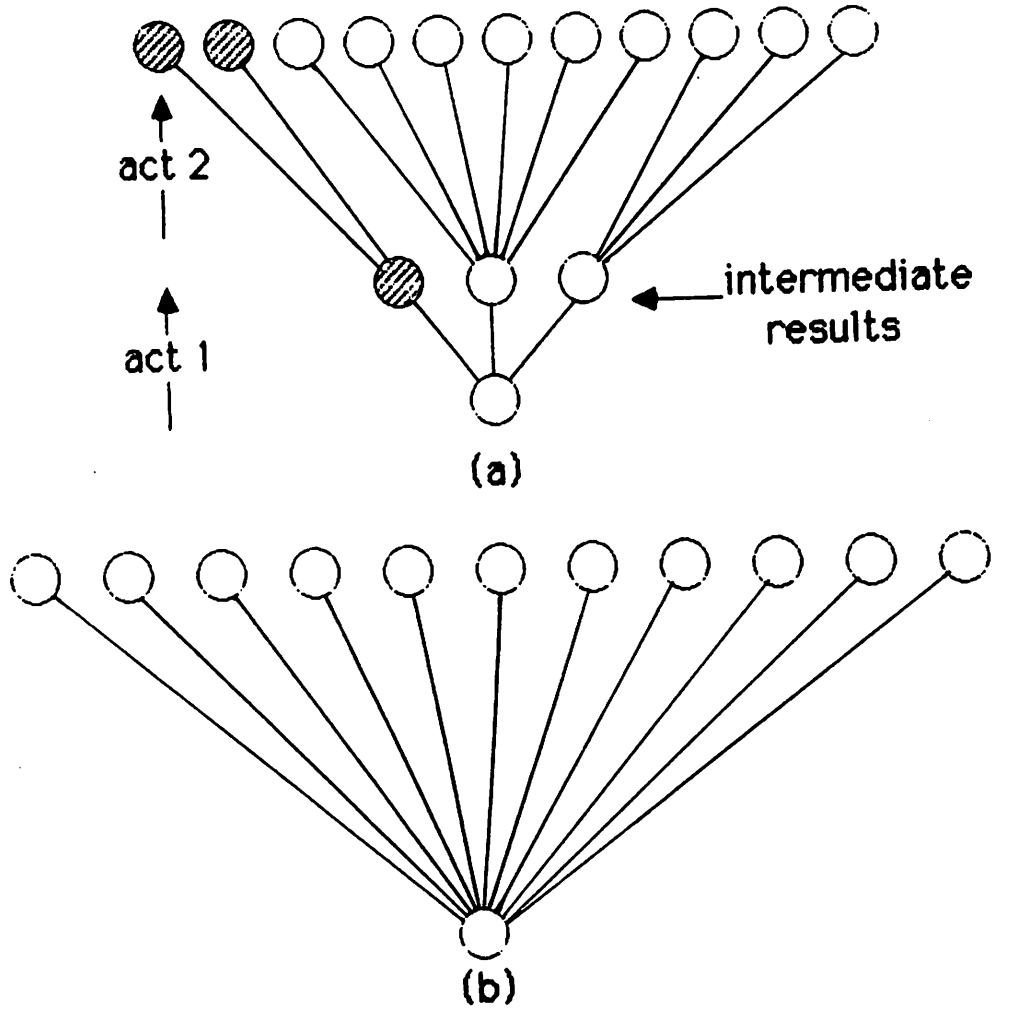


Figure 3: Eliminating Intermediate Processing Steps.

intermediate interpretations[4] (e.g., a shaded intermediate result in Figure 3a). If the certainty of this intermediate result is very high, then exploring the other two (unshaded) intermediate results will be unnecessary. In this case, the presence of the intermediate step has reduced the amount of search.

Approximations which combine processing steps are well-behaved and are appropriate when the speedup is sufficient to balance off 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

[4]It is possible to have opportunism within the combined KS, but this 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 KS application.

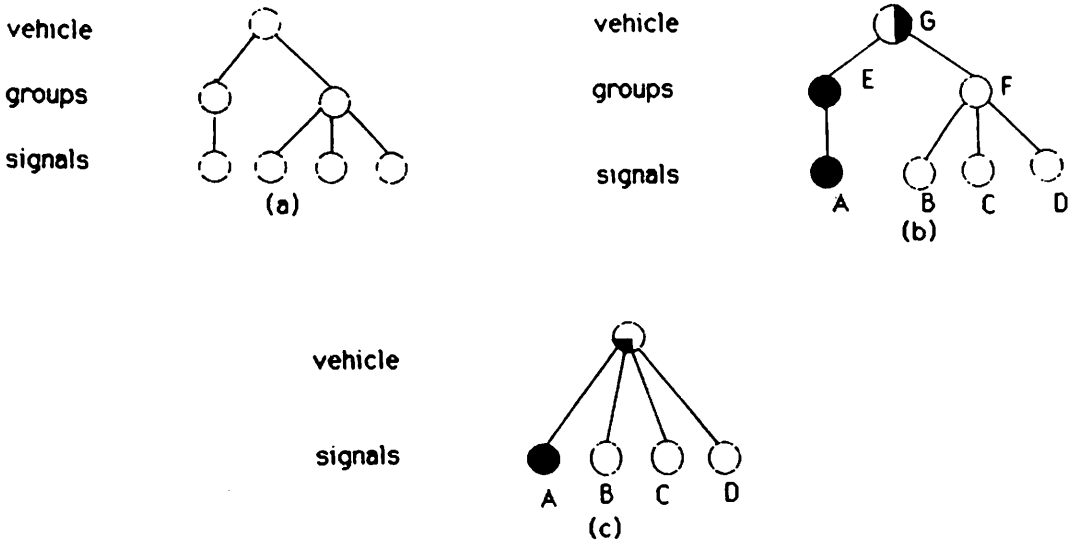


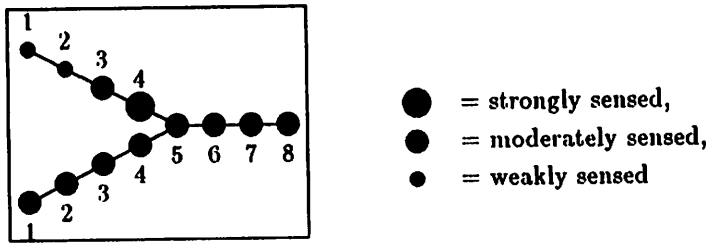
Figure 4: Level Hopping.

grouped, and 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, while 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, since signal A provides 50% 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, since it is not known which group the signal belongs to.

6 Preliminary Experiments

To develop an initial understanding of how approximate processing affects problem solving, we have implemented three approximation mechanisms as part of the DVMT. These mechanisms reduce problem-solving time to better meet deadlines in different ways. The first mechanism eliminates competing interpretations (Section 3.2), and we have 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 solu-



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

Figure 5: The Experimental Problem Situation.

tions by incompletely processing the data (Section 4.1): 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.[5] The third mechanism eliminates corroborating support (Section 3.1) by removing planned actions whose only purpose is to increase the certainty in an interpretation. For example, rather than looking for all of the different sounds a particular type of vehicle makes, the problem solver will conclude (with lower certainty) that the vehicle is present based on a few sounds.

The user specifies the objectives to the 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. There are two competing solutions 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, 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 data for the upper extension was perhaps due to echoes in the environment.

[5] Ignoring this data does not significantly affect certainty in the incomplete solution (see Section 4.1).

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 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 KS 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 very 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, it will always explore 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 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 while E2 predicts that 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 that there is insufficient time to form the best solution and thus revises its plan to meet the deadline. Told by the experimenter (the consumer of the solution) to prefer to sacrifice certainty in the solution, it removes enough actions that generate corroborating support so that it still forms a complete solution (covering all the sensed times) but with only 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 since it still expects to exceed the deadline after it has removed all the corroborating actions. It thus sacrifices some completeness by ignoring data for 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 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 (KS) takes one time unit, the planner can drop just enough corroborating actions (single KSs) to meet the deadline exactly (E3–E5). When it needs to get high belief and ignores data less important sensed times, the planner may not meet deadlines exactly (E6) since processing that data may require several KSs.

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 can the problem solver flexibly choose approximations that are right for a particular situation; and how does the time spent in planning and deciding on approximations affect real-time activities. The first issue will be discussed below; 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 [2,3,4]. This research indicates that, because problem-solving activities (applying domain knowledge to some set of data) are typically very expensive, the modest amount of overhead spent planning appropriate problem-solving actions is worthwhile.

7 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.

7.1 Recognizing the Need for Approximate Processing

Recognition of the need for approximate processing comes naturally from planning out the problem-solving activities for achieving system goals, estimating the time these activities will take to achieve a solution, and monitoring the progress of these plans.[6]

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, since more 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 [2,3]. Comparing the adjusted solution time with the given deadline may reveal that the current plan cannot be completed on time. Besides monitoring the plan, the planner needs to monitor the system state, since slow progress on a plan may be the result of a changed situation. For example, the amount of input data can increase due to busier traffic in the area, or a processor malfunction may impose stronger resource constraints. In either case, in order to satisfy the objective the planner must make a new, approximate plan which consists of faster or fewer activities and takes a shorter time to complete.

7.2 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 which gives the best quality within the allowable time. The evaluation involves forming a plan with *approximate activities* (described in Section 7.3) 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 may 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

[6] An alternative way for recognizing the need for approximate processing is to have some simple measure to indicate appropriate problem-solving progress. For more details of this approach see work by Hudlická and Lesser [9]. This simpler but less precise method could also be used in conjunction with the techniques developed in Section 7.2, except that the final step which is the verification of the new objective and control strategy by constructing a detailed plan would be eliminated.

the "optimal" approximation may 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 very small fraction of applicable approximations which are the ones most likely to be relevant in the current state of processing. This is accomplished by introducing **approximation rules**: heuristic situation–action rules which link the state of processing with approximate processing strategies. They state the situation preference for approximations by specifying approximations which 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 which ignores less relevant data based on recency of arrival may have preconditions specifying sensed times eligible to be ignored. If there are no data 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 since situations are based on global features of the state of processing, while precondition tests are based on more detailed data attributes.

Another method that 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 which 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 which satisfies the objective is also the best one, and no other approximations need to be

Incomplete Event Processing

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

Figure 6: Incomplete Event Processing Approximation.

evaluated.[7] Note, however, that the optimality of the approximation is not greatly affected by possible errors in the loss model, since 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), since 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 is the case in our "incomplete event processing" approximation example, where the speedup is greater when more 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 which specifies how an approximation is applied the **value** of the approximation. For the approximation which ignore less recent data in the 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 which we call the **approximation domain**. Sample information in the approximation domain for the 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 is 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:

[7] This assumes that in most cases a single approximation or combined group of approximations coming from a single situation-action rule will lead to a better solution than combinations of approximations.

1. Use approximation rules to generate candidate approximations.

2. Use preconditions to filter out non-applicable 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 quality loss to speedup ratio.

4. **If** there is no approximation or a 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-behaved approximations)
   else pass this approximation to step 5.

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

6. **If** the evaluated approximation satisfies the objective
   then exit with success
   else 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."

7.3 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 like the DVMT, 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 which 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, elimination of plan steps, substitution of regular activities in the plan by approximate activities, and inserting special activities in the plan. Section 6 described

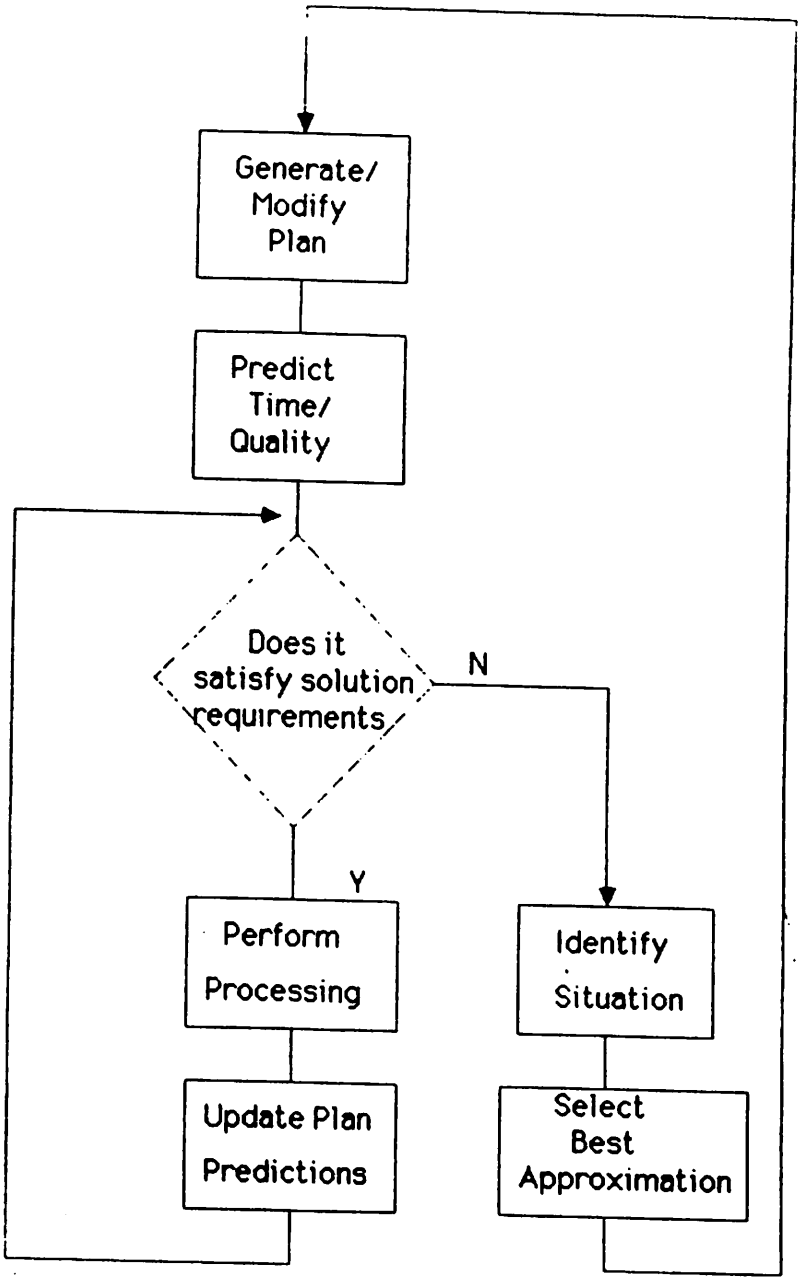


Figure 7: The DVMT Planning Loop.

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 [1]. 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 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 will specify the area where noise was detected as the desired location range, and the location precision will be the clustering metric.

Another mechanism needed to implement approximate processing is a generalized view of data. In the the 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 which have different levels of precision. Thus, processing of data objects with different precisions could be freely intermixed, and the more detailed constraints of more precise results can be exploited when they are available.

As part of the DVMT, we have developed several of the approximations discussed in Sections 3, 4, and 5:

Eliminating corroborating support – multiple types:

> This approximation was illustrated in the experiments (Section 6). 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:

> More general tracking knowledge sources take time resolution as a parameter (e.g., exact tracking has a time resolution of 1, tracks are formed with data at every

sensed time, while skipping every other sensed time would have a time resolution of 2). Plan modification for this approximation involves 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 Section 6).

Incomplete event processing – ignoring attributes:

We introduce approximate processing activities which 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 the 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 Section 6).

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 upon, 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.

8 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 are:

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* which 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. Since these estimates are uncertain (problem solving is to some extent unpredictable) the problem solver may 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 which trade off 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 those situations without investing large amounts of effort. The problem solver thus becomes more versatile due to 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, more flexibility is given to the system to find the best quality solution within the remaining time. The work of Durfee [2] and Pavlin [15] shows how to make reasonable predictions in the DVMT, and we must

develop corresponding mechanisms for other types of AI systems. We should also explore other important issues such as interrupting KS processing, allowing for ongoing, low-level sensor monitoring and effector control, and predicting and managing the cost of control decisions [6,14].

We have considered real-time control in the context of a blackboard-based interpretation system which plans out its problem-solving activities. In this context, we have discussed how deadlines can be met by reducing the solution's completeness, precision, certainty, or any combination of these, and have 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 have outlined a computationally efficient framework for choosing and pursuing suitable approximations in a blackboard-based interpretation system.

9 Acknowledgments

This work was first initiated in 1986 in 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.

REFERENCES

[1] Daniel D. Corkill, Victor R. Lesser, and Eva Hudlická.
"Unifying Data-Directed and Goal-Directed Control: An Example and Experiments." *Proceedings of the National Conference on Artificial Intelligence*, pages 143–147, Pittsburgh, Pennsylvania, August 1982.

[2] Edmund H. Durfee and Victor R. Lesser.
"Incremental Planning to Control a Blackboard-Based Problem Solver." *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 58–64, August 1986.

[3] Edmund H. Durfee and Victor R. Lesser.
"Incremental Planning to Control a Time-Constrained, Blackboard-Based Problem Solver." *IEEE Transactions on Aerospace and Electronic Systems*, special issue on space telerobotics, in press. (Also published as Technical Report 87-07, COINS Department, University of Massachusetts, Amherst, 1987.)

[4] Edmund Howell Durfee.
A Unified Approach to Dynamic Coordination: Planning Actions and Interactions in

a Distributed Problem Solving Network. Ph.D. Thesis, University of Massachusetts, September 1987.

[5] 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*, Vol. 12 pages 213–253, June 1980.

[6] Michael R. Fehling, Kurt Joerger, and Daniel Sagalowitz.
"Knowledge Systems for Process Management." *Proceedings of the ISA-86*, Houston, Texas, October, 1986.

[7] Barbara Hayes-Roth.
"A Blackboard Architecture for Control." *Artificial Intelligence*, Volume 26, pages 251–321, 1985.

[8] Barbara Hayes-Roth.
"A Multi-Processor Interrupt-Driven Architecture for Adaptive Intelligent Control." Technical Report KSL 87-31, Knowledge Systems Laboratory, Stanford University, June 1987.

[9] Eva Hudlická and Victor R. Lesser.
"Meta-Level Control Through Fault Detection and Diagnosis." *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 153–161, 1984.

[10] Victor R. Lesser and Lee D. Erman.
"A Retrospective View of the Hearsay-II Architecture." *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, USSR, pages 790–800, 1977.

[11] Victor R. Lesser and Daniel D. Corkill.
"The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks." *AI Magazine*, Vol. 4, No. 3, pages 15–33, 1983.

[12] H. Penny Nii.
"Blackboard Systems." *AI Magazine*, Part 1 in Vol. 7 No. 2 pages 38–64, Part 2 in Vol. 7 No. 3 pages 82–106, 1986.

[13] James G. March and Herbert A. Simon.
ORGANIZATIONS, Wiley, 1958.

[14] William J. Pardee and Barbara Hayes-Roth.
"Intelligent Real-Time Control of Material Processing." Rockwell International, Science Center, Palo Alto Laboratory, Research Report #1, February 1987.

[15] Jasmina Pavlin.
"Predicting the Performance of Distributed Knowledge-based Systems: A Modeling Approach." **Proceedings of the Third National Conference on Artificial Intelligence**, pages 314–319, Washington, DC, August 1983.

[16] Krithivasan Ramamritham and John A. Stankovic.
"Dynamic Task Scheduling in Distributed Hard Real-Time Systems." *IEEE Software*, Vol. 1, No. 3, May 1984.

[17] Herbert A. Simon.
The Sciences of the Artificial The M.I.T. Press, Cambridge, Mass., 1969.

[18] John A. Stankovic, Krithivasan Ramamritham, and ShengChang Cheng.
"Evaluation of a Flexible Task Scheduling Algorithm for Distributed Hard Real-Time Systems." *IEEE Transactions on Computers*, Vol. C-34, No. 12, December 1985.

[19] *IEEE Expert*, Issue on Real-time Control of Autonomous Mobile Robots, Vol. 2 No. 4, Winter 1987.

[20] *Workshop on Real-Time Control at the Sixth National Conference on AI*, Seattle, Washington, July 1987.