Building a Plan with Real-Time Execution Guarantees

Article · September 1996

Source: CiteSeer

CITATION
1

READS
15

3 authors, including:

Ella M Atkins
University of Michigan
263 PUBLICATIONS **5,882** CITATIONS
SEE PROFILE

Edmund H. Durfee
University of Michigan
345 PUBLICATIONS **9,082** CITATIONS
SEE PROFILE

Building a Plan with Real-Time Execution Guarantees

Ella M. Atkins **Edmund H. Durfee** **Kang G. Shin**

University of Michigan AI Lab
1101 Beal Ave.
Ann Arbor, MI 48109
{marbles, durfee, kgshin}@umich.edu

Abstract

The degree to which a planning system succeeds depends on its ability to meet critical deadlines as well as the correctness and completeness of its models which describe events and actions that change the world state. It is often unrealistic to expect either unlimited execution time or perfect models, so a planner must be able to make appropriate time vs. quality tradeoffs, then detect and respond to states it had not originally planned to handle. In this paper, we consider these issues in the context of the Cooperative Intelligent Real-time Control Architecture (CIRCA), which combines a planner with a separate real-time system so that plans are built, scheduled, and then executed with real-time guarantees. Specifically, we discuss our recent addition of a probabilistic model to help the planner prioritize states for expansion, and present important classes of "unplanned-for" states that we detect and handle in CIRCA. Finally, we describe our current work to improve CIRCA's planner by estimating planning time constraints in advance and incorporating a more intelligent utility function to prioritize states.

1 Introduction

Fully automating complex systems requires the ability to reason about possible world events and react, often quickly, to sensory input. Ideally, a plan could be built to handle all possible situations, as suggested in Universal Planning work (Schoppers 1987), but such computations are often prohibitively complex in many practical situations. Conversely, one might plan for a very small set of "highly-probable" states, but then many situations may not be handled at all.

Automated systems impose time constraints in two ways. First, some restriction must be placed on the planner's deliberation time; otherwise, the planner could take unacceptably long to produce a result. Second, since planning involves selecting actions that must execute in the real world, these actions may need associated execution time constraints. For example, suppose a mobile robot planner selected an action to change directions when detecting an obstacle. That action must be executed before the robot strikes the obstacle. If this robot had numerous tasks to perform, the "change direction" action might be ignored until it was too late. Such problems led to the idea of guaranteeing execution times, particularly when reacting too slowly could result in failure.

We study such problems within the context of CIRCA (Cooperative Intelligent Real-time Control Architecture) (Musliner, Durfee, and Shin 1995), which combines a planner, scheduler, and separate real-time plan execution module to build, schedule, then execute plans with real-time guarantees of system safety. CIRCA differs from an anytime planning approach (Dean et al. 1993) in that it primarily considers *execution time* guarantees in its plans. Thus it would ensure that the robot in the example above would react in time to avoid the obstacle, so long as obstacle avoidance had been planned for. While CIRCA does not presently limit planning time, it does guarantee real-time action via a separate real-time plan execution subsystem. We hope to incorporate planning time limits in future improvements to CIRCA.

Working with CIRCA has illustrated some basic problems involved with taking a complex problem, specifying it in terms of a planner's knowledge base, and trying to make claims of *guaranteed* safety. First, providing the planner with comprehensive knowledge is virtually impossible -- particularly when experts do not yet have a complete understanding of the domain. Thus, if some possible event is missing or misrepresented, or if some system sensor or actuator fails to operate as expected, safety guarantees are lost. We have been working to solve this problem by incorporating state feedback to the planner when an "unexpected" (or "unplanned-for") state is reached, and by building a probabilistic model which allows statistical knowledge base specification and assists both planning and plan scheduling by eliminating highly improbable states (so we relax the 100% safety guarantee to, say, a 99.99% guarantee of safety). Details of CIRCA are presented in

Section 2, followed by a discussion of our recent additions to the architecture in Section 3.

We briefly assess CIRCA's strengths and weaknesses in Section 4. A major strength of CIRCA is its careful consideration of the timings associated with planned action execution. With our recent additions, we feel we have significantly enhanced its capability to limit its search through state-space and compensate for unexpected occurrences. However, one of CIRCA's main weaknesses is that the planner currently has no imposed execution time limit. We realize this is unrealistic for many situations, particularly when replanning to handle "unexpected" states. Although we have no finalized algorithms, we present our ideas for limiting planning time and more accurately prioritizing states in Section 5, followed by a brief conclusion (Section 6).

2 CIRCA Background

Figure 1 shows a block diagram of CIRCA. The AI subsystem (AIS) contains both the planner and the scheduler. The "shell" around all AIS operations consists of meta-rules controlling a set of knowledge areas, similar to the PRS architecture (Ingrand and Georgeff 1990). Working memory contains tasks that are ready to be executed. These tasks include planning, downloading plans from the AIS to the real-time subsystem (RTS), and reading/processing feedback data from the RTS.

The CIRCA knowledge base specifies a list of goals which, when achieved in order, will enable the system to successfully reach its final goal. CIRCA executes a planning cycle for each new goal in this list. To minimize domain knowledge complexity, the CIRCA world model is created incrementally based on the initial state set and a group of temporal and action state transitions. Each transition has a name, precondition set, and postcondition set. Action transitions correspond to commands that are explicitly executed by the CIRCA RTS, while temporal transitions correspond to state changes that are not initiated by CIRCA.[1] The planner currently selects actions based on simple criteria, including number of goal features achieved and proximity to failure, and eventually backtracks if a selected action does not ultimately help achieve a goal or avoid failure. CIRCA minimizes its use of memory and time by expanding only states explicitly produced by transitions from initial states or their descendants.

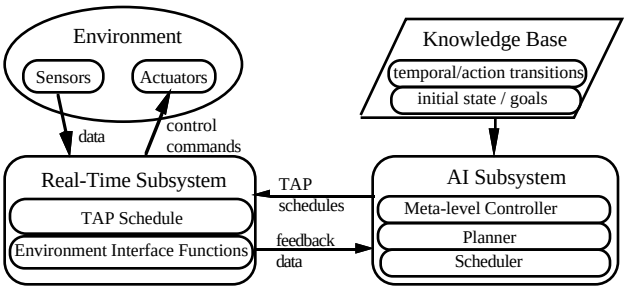

Figure 1: CIRCA Architecture

Figure 2 shows a typical state set expanded during a planning cycle. CIRCA begins planning by selecting one of the initial states and building a list of descendants resulting from temporal transitions (tt). If a tt leads to a state that potentially violates the agent's safe operating envelope, then the state is labelled "failure" and the tt is labelled TTF -- temporal transition to failure. In this case, CIRCA selects an action and associated execution deadline to guarantee avoidance of the TTF. Otherwise, CIRCA may select an action that moves the system closer to the goal. CIRCA continues state expansion for all other initial states and their reachable descendants until at least one goal state is found and all reachable TTFs are guaranteed to be avoided. Note that the planner is minimally satisfied with only one goal path due to tradeoffs between completeness and schedulability (Musliner, Durfee, and Shin 1995). Thus, as shown in the figure, some reachable states (labeled "deadend") do not lead to the goal. These states are "safe" because all TTFs are preempted by actions, but the system has no chance of achieving its goals from those states. Replanning for goal achievement when a deadend state is encountered is discussed in (Atkins, Durfee, and Shin 1996b).

CIRCA's control plans are represented as cyclic schedules of test-action pairs (TAPs). Tests involve reading sensors; actions involve sending actuator commands or transferring data between CIRCA modules. When the AIS planner creates a TAP, it stores an associated worst-case execution time and execution deadline to enable safety guarantees. These TAP attributes are then used by a deadline-driven scheduler (Liu and Layland 1973) to create a periodic TAP schedule. If the scheduler is unable to create a schedule that supports all deadlines, the AIS backtracks to the planner, which then selects different actions. This backtrack-(select actions) cycle repeats until either the scheduler succeeds or until the planner can find no actions that avoid failure and reach the goal.

[1] Previously (Musliner, Durfee, and Shin 1995), CIRCA contained three transition types: action, temporal, and event. "Events" can occur instantaneously while "temporals" have a non-zero delay. We now model events and temporals as "temporal transitions", with differences specified using transition probabilities.

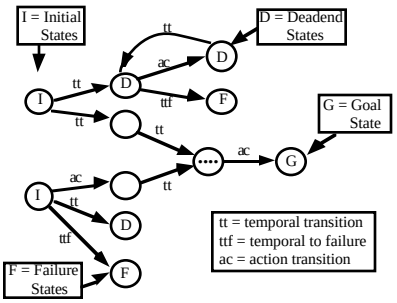

Figure 2. States Expanded during Planning.

3 CIRCA Enhancements

We have recently improved CIRCA in two respects. First, we sought to relax the absolute 100% safety guarantee requirement, at least when planning and scheduling become prohibitively difficult. To do this, we have implemented a model in which transition probabilities (specified in the domain knowledge base as any functions of time) are used to approximately compute state probabilities as described in (Atkins, Durfee, and Shin 1996a). These probabilities are then used by the planner so that state expansion occurs in decreasing order of probability. This provides two advantages: (1) highly-probable goal paths are identified, saving the planner time by not requiring expansion of all potentially goal-reaching states, and (2) highly-improbable states may be removed from consideration when scheduling is difficult or impossible. This probabilistic model has been tested using an aircraft simulator and is discussed in (Atkins, Durfee, and Shin 1996b).

We also have incorporated and tested algorithms for detecting and handling certain classes of "unplanned-for" states. Figure 3 characterizes the relationships between subclasses of all possible world states for any domain. At the top level, states are either "modeled" or "unmodeled". Modeled states are those whose distinguishing features and values are represented in the planner's knowledge base. Because the planner cannot consider unmodeled states without the addition of a feature discovery algorithm, unmodeled states cannot be considered. Within the modeled set, the "planned-for" states include those the planner has expanded. This set is divided into two parts: "handled" states which avoid failure and can reach the goal, and "deadend" states which avoid failure but cannot reach the goal with the current plan.

Aside from the "planned-for" states, a variety of other states are modelable by the planner. Such states include those identified as reachable, but which have been "removed" because attending to them along with the "planned-for" states exceeds the system's capabilities. Other modeled states include those that indicate "imminent failure;" if the system enters these states, it is likely to fail shortly thereafter. Note that some states might be both "removed" and "imminent-failure", as illustrated in Figure 3. Finally, some modeled states might not fall into any of these categories, such as the states the planner considered unreachable from the initial states but that are not immediately dangerous.

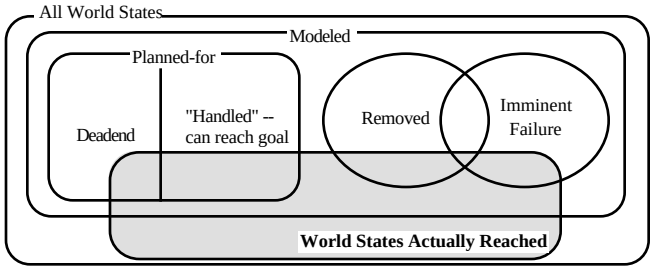

Figure 3. World State Classification Diagram.

The shaded region in Figure 3 illustrates a state set reached during plan execution. To assure reaching the desired goal, the set should be empty except for where it overlaps the "handled" region. To assure safety, the set should only have elements that are in the "planned-for" region. When the set has elements outside these regions, safety and performance depend on detecting the new state and responding appropriately.

A critical premise in our work is that a planner, or system in which it is embedded, cannot be expected to somehow just "know" when it has deviated from plans---it must explicitly plan actions and allocate resources to detect such deviations. For example, to make real-time guarantees, CIRCA's AI planner must specify all TAPs the RTS will execute, including any to detect and react to unhandled states. In our implementation, after the planner builds its normal plan, it builds special TAPs to detect each of the deadend, removed, and imminent-failure unhandled state classes. First, the planner builds separate lists of all states that fall into each unhandled state class. The TAP tests for an unhandled state class *could* include an explicit test for every set of state features in that unhandled state class list, but these tests would be repeated frequently during plan execution and may be unnecessarily time-consuming with long state lists. Thus, once each list is completed, the planner calls the ID3 test minimization algorithm (Quinlan 1986) with that unhandled state list as the set of positive examples and a subset of the reachable states (depending on the unhandled state type) as the set of negative examples. ID3 returns what it considers a minimal test set, which is then used as the special TAP test to detect that unhandled state class.

When one of these special TAPs detects an unhandled state, CIRCA's RTS feeds back all state feature information

to the AIS. The AIS then reads the uplinked state feature values and selects a subgoal (from the knowledge base list) that is closest to the final goal but with preconditions matching the uplinked state features. Next, the AIS runs the state expansion part of the planner, using the state feature feedback as the initial state, all temporal transitions, and the executing plan's test-action (TAP) transitions. The state list returned from this state expansion routine contains all possible states the RTS could reach *while* the AIS is replanning, thus each is a possible initial state when the new plan begins executing. The AIS then replans using this potentially large initial state set and the selected subgoal. This new plan is downloaded to the RTS which will then have the ability to react to the previously unhandled state and its descendants.

4 Assessment of CIRCA

The main motivation driving the original CIRCA development was the ability to specify real-time guarantees of safety via explicit scheduling of critical planned actions. As a first approximation, worst-case times were assumed for action execution and temporal transitions to failure during scheduling. In this manner, guarantees were provided with 100% certainty (so long as the model was correct), but there was no sense of compromise, so the planning and scheduling was slow, and scheduling sometimes failed completely. Additionally, it was assumed that an executing plan could keep the system safe indefinitely, thus there were no restrictions placed on planning and scheduling execution time.

Because of the emphasis placed on execution guarantees, the real strength of CIRCA was its combination of "off-the-shelf" algorithms from the AI and real-time systems fields. Although tests used relatively simple knowledge bases, researchers demonstrated that it could plan, build schedules, and successfully reach goals while avoiding failure in several situations, including a simulated assembly line processing task with time-critical reaction to aperiodic events (Musliner, Durfee, and Shin 1995). CIRCA's modular architecture provides a good platform for introducing state-of-the-art planning and real-time systems technology, especially due to the meta-level architecture controlling AIS execution.

We feel we have enhanced CIRCA's capabilities by relaxing the restrictions on model precision -- handling important "unplanned-for" states and computing approximate state probabilities. Due to the generic nature of our planner, we feel our algorithms to compute probabilities and detect "unplanned-for" states are not CIRCA-specific, but applicable to any planner which considers execution times and imprecise models.

Improvements are still needed, particularly to our probabilistic model (Atkins, Durfee, and Shin 1996a). CIRCA's planner allows cycles in its state diagrams, a realistic representation of many real-world situations, such as executing a holding pattern in an aircraft. Such cycles may introduce significant inaccuracies in our approximate state probability calculations. Also, we currently require the user to specify all time-dependent probability functions for temporal transitions in the domain knowledge base -- a daunting task because these functions must account for probabilistic dependencies when multiple transitions match the same state.

5 Future Work -- Limiting Planning Time

Researchers generally agree that one must restrict planner execution time, thus meta-level controlling mechanisms, such as anytime (Dean et al. 1993) and design-to-time (Garvey, Humphrey, and Lesser 1993) algorithms, have been added to impose limits on planning execution time, and learning mechanisms, such as chunking in SOAR (Rosenbloom, Laird, and Newell 1993), have been added to increase planning execution speed. However, use of an anytime algorithm alone does not guarantee good-quality results unless the planner has had time to complete at least an approximate plan, and learned chunks do not even exist until the situation has been encountered at least once. The tradeoff between planner execution time and plan quality has been documented from many perspectives, but in proposed solutions, either the time or quality must be compromised when new, complex problems are encountered.

CIRCA's reaction to unhandled states is coincidentally real-time, and was successful during our relatively simple flight simulation tests. This may not always be the case, particularly when unhandled states quickly lead to failure. Timely reactions may be achieved either by bounding replanning execution time or by building reactions in advance. As planning technology progresses, more architectures employ methods for computing and adhering to planner execution time bounds, as discussed in (Dean et al. 1993), (Hendler and Agrawala 1990), (Horwitz 1988), (Ingrand and Georgeff 1990), (Musliner, Durfee, and Shin 1995), and (Zilberstein 1994). We feel a combination of these and other innovative ideas is required before achieving a near-optimal balance between planning time and plan quality. We would ideally like to use the planner's initial state(s) to quickly compute an initial estimate of a planning time limit, then modify this estimate based on environment changes during planning. Assuming an algorithm to compute a planning time limit, we could use a type of design-to-time algorithm. We are now expanding states in inverse order of probability, so the most probable states will be handled first. Whenever the planner's execution time

expires, the probability of the next state to be expanded will be the maximum probability of reaching any single unplanned-for state.

There are still potential quality problems with this algorithm. Perhaps most important, with a short planning deadline, high probability states may remain unexpanded. The planner may make use of its knowledge about probabilities and temporal delays, as well as knowledge about relative "importance" among states, to make tradeoffs that may improve planning. State expansion may be ordered by decreasing utility u(s), as shown in equation (1), where p(s) = probability of reaching state s, t(s) = minimum time before the system can reach state s, pf(s,n) = probability of reaching failure in n (or fewer) steps from state s, and a, b, and c are (as yet undetermined) scaling constants. By expanding states in this order, we hope to plan for the most "important" states, achieving a balance between state probability, system safety (i.e., prioritizing expansion to handle states that can reach failure), and the time horizon considered by the planner (i.e., near-term states are handled; far-term states will be handled by subsequent plans). Unfortunately, even the best u(t) will not guarantee a high-quality plan when the planner must execute quickly, so we continue to search for ways to achieve the ever-elusive balance between quality and planner deliberation time.

$$u(s) = a * p(s) + b * t(s) + c * pf(s, n) \qquad (1)$$

6 Conclusion

This paper presents a discussion of real-time challenges associated with building and executing plans in the context of CIRCA, an architecture which separates planning from plan execution to allow hard real-time guarantees without truncating planning. Unfortunately, incomplete knowledge and bounded resources may allow reachable world states to be unhandled during planning. Recently, we incorporated a probabilistic model which is used by the planner to select highly-probable goal paths and to remove improbable states from consideration when planning or scheduling difficulties arise. We introduce a classification hierarchy of possible world states, and identify three important unhandled state classes -- deadend, removed, and imminent-failure. New algorithms implemented within CIRCA can detect these unhandled states when reached and subsequently replan to handle them.

We have concentrated on timeliness issues associated with plan execution, not planning itself. However, our work directly addresses one important issue in time-limited planning -- pruning the search space by expanding only reachable states and ignoring low-probability states. As domain complexity increases, we recognize that we may need to impose explicit restrictions on planning time, particularly when replanning for an unhandled state that may lead to failure. We hope to take advantage of our probabilistic model and temporal knowledge to expand states in order of decreasing utility, then terminate planning before time expires with only low-utility states remaining unexpanded. We are still working to build algorithms that will help the planner consistently achieve an appropriate tradeoff between time and result quality.

One of our major long-term research goals is to help achieve safe, fully-automated aircraft flight -- a challenging task due to the multitude of possible events (e.g., sensor or actuator failures, traffic, weather, etc.) and tight restrictions on response time and quality, since the only absolutely safe state is when the aircraft sits motionless on the ground. Additionally, aircraft control is not a "solved problem" – uncertain events such as collision-course traffic may occur. In fact, most emergency situations are now handled by pilots, and even the best pilots may select suboptimal responses. Therefore, fully-automated aircraft flight will require approximate knowledge, and quite a large knowledge base will be necessary. Limiting planning time and carefully scheduling planned actions will thus be crucial for safe, fully-automated flight in which all expected time-critical operations are part of a scheduled plan and have near 100% chance of executing properly. Emergency situations must be handled by some combination of pre-planned reflex actions and fast replanning. We have begun to investigate methods to build plans to handle such emergency situations, including necessary feature-value specifications for aircraft, types of time-dependent temporal transition probabilities to use, and available sources of statistical data (e.g., NTSB (National Transportation Safety Board) aircraft accident reports). We have tested simple situations such as "collision-course traffic" and "landing gear fails on final approach", but will need to significantly enhance CIRCA's planning and temporal reasoning capabilities before a full-scale aircraft control knowledge base could be handled with sufficient speed and accuracy to be trusted during emergencies.

7 Acknowledgements

This work was supported under NSF grant IRI-9209031.

8 References

Atkins, E. M., Durfee, E. H., and Shin, K. G., "Plan Development in CIRCA using Local Probabilistic Models," to appear in *Uncertainty in Artificial Intelligence: Proceedings of the Twelfth Conference*, August 1996a.

E. M. Atkins, E. H. Durfee, and K. G. Shin, "Expecting the Unexpected: Detecting and Reacting to Unplanned-for World States," to appear in *Proceedings of AAAI Workshop on Theories of Action and Planning: Bridging the Gap,* August 1996b.

Dean, T., Kaelbling, L. P., Kirman, J, and Nicholson, A., "Planning with Deadlines in Stochastic Domains," *Proceedings AAAI-93,* AAAI, pp. 574-579, 1993.

Garvey, A., Humphrey, M., and Lesser, V., "Task Interdependencies in Design-to-time Real-time Scheduling," *Proceedings AAAI-93*, AAAI, pp. 580-585, 1993.

Hendler, J. and Agrawala, A., "Mission Critical Planning: AI on the MARUTI Real-Time Operating System," in *Proc. Workshop on Innovative Approaches to Planning, Scheduling and Control*, pp. 77-84, November 1990.

Horvitz, E. J., "Reasoning under Varying and Uncertain Resource Constraints", *Proceedings AAAI-88*, AAAI, pp. 111-116, 1988.

Ingrand, F. F. and Georgeff, M. P., "Managing Deliberation and Reasoning in Real-Time AI Systems," in *Proc. Workshop on Innovative Approaches to Planning, Scheduling and Control*, pp. 284-291, November 1990.

Liu, C. L., and Layland, J. W., "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of the ACM,* vol. 20, no. 1, pp. 46-61, January 1973.

J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, pp. 81-106, 1986.

Rosenbloom, P. S., Laird, J. E., and Newell, A., (eds), *The Soar Papers: Research on Integrated Intelligence,* MIT Press, 1993.

Musliner, D. J., Durfee, E. H., and Shin, K. G., "World Modeling for the Dynamic Construction of Real-Time Control Plans", *Artificial Intelligence*, vol. 74, pp. 83-127, 1995.

Schoppers, M. J., "Universal Plans for Reactive Robots in Unpredictable Environments," in *Proc. Int'l Joint Conf. on Artificial Intelligence,* pp. 1039-1046, 1987.

Zilberstein, S., "Real-Time Robot Deliberation by Compilation and Monitoring of Anytime Algorithms," *AAAI Conferenc*e, pp. 799-809, 1994.