Working Notes of the AAAI Workshop on Representational Issues for Real-World Planning Systems
Austin, TX, July 2000

Planner Feedback: NIL is Not Enough

(Extended Abstract)

David J. Musliner
Automated Reasoning Group
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
musliner@htc.honeywell.com

Introduction

In the beginning, there was always a plan. Monkeys and bananas, cannibals and missionaries, blocks world... the planner just had to be smart enough, and look hard enough, and there was a plan to be found. Even in "real world" problems, there was always a plan: you can always find a way to get from Princeton to Brown, even if the airport is closed. If the planner couldn't find a plan, either the planner was broken or the domain description was broken. Building complex domain descriptions was still hard, because planners would say little about *why* they couldn't build a plan. But now the problem is much harder.

Now, planners are going to control spacecraft and refineries and autonomous combat aircraft, and there may not always be a plan. Parts break, sensors fail, accidents happen, adversaries thwart intentions. The planner (or at least the overall agent control system) will have to make tradeoffs: if the goal to take high-resolution pictures cannot be satisfied, take low-resolution pictures; if the pump breaks and temperature skyrockets, reduce throughput; if the enemy may shoot you down, fly as carefully as you can.

When there may not be a plan that accomplishes the goals perfectly, two new problems arise. The first obvious problem is "how do we get the planner to make the requisite tradeoffs?" The second problem, which is less obvious but probably at least as important for real-world applications, is "When the planner can't find a perfect plan, how do we know if the domain description is right or not?" One key to addressing both problems is planner feedback.

Context: Self-Adaptive CIRCA

To place this discussion in context, let's consider some examples based on the desired behaviors of the Self-Adaptive CIRCA (SA-CIRCA) system (Musliner *et al.* 1999). Our goal here is not to describe SA-CIRCA in detail; rather, we give a brief overview of the system concepts and refer the interested reader to details available in other papers (Musliner, Durfee, & Shin 1993; 1995; Musliner *et al.* 1999). Roughly illustrated in Figure 1, SA-CIRCA is designed to control systems interacting with adversarial environments that pose hard real-time deadlines: that is, domains in which it is possible to incur catastrophic failures by not acting quickly enough. For the purposes of this discussion, the important point is to understand the roles of the Adaptive Mission Planner (AMP) and the Controller Synthesis Module (CSM).

The basic idea is that the AMP conducts long-term, highly abstract planning and control of the overall mission of the system, and directs the CSM to develop suitable reactive controllers for differ-

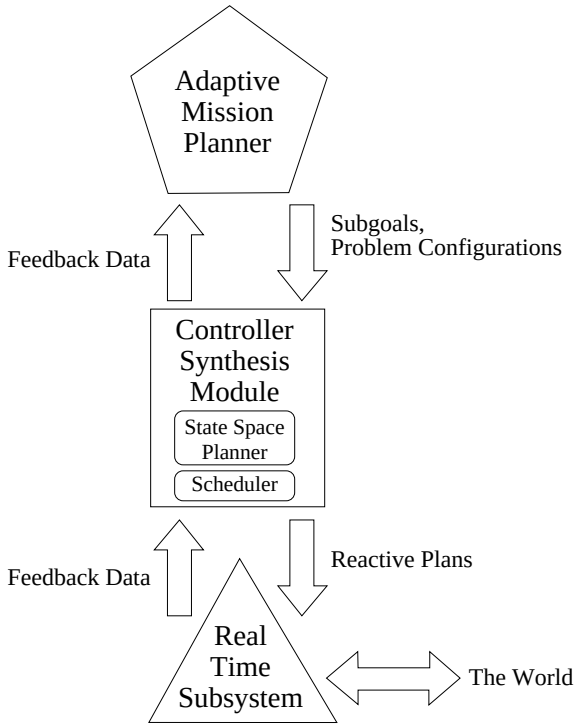

Figure 1: Within SA-CIRCA, the AMP and CSM engage in a negotiation process to derive the best plans using the available deliberation and reaction resources.

ent *phases* of the mission. For example, an uninhabited combat aerial vehicle (UCAV) mission may consist of ingress, attack, and egress phases which each require different reactive control plans to adjust the system's behavior in response to different types of threats and failures. The AMP formulates *problem configurations* for each of these phases, and asks the CSM to generate plans that can guarantee to avoid failure and achieve the intermediate goals. In the UCAV example, the ingress phase problem configuration might specify the types of threats expected along the ingress route (potential failures) and the endpoint (goal) of the ingress. The CSM contains a different sort of planner: for each phase, it tries to build a time-constrained reactive plan that is guaranteed to preserve system safety (while achieving other goals) by accounting for the environment, nondeterminism, and external adversarial agents.

We designed SA-CIRCA with the assumption that its application domains will be overconstrained; that is, the system will not always be able to guarantee that it will both keep the system safe (avoid catastrophic failures) and achieve all the goals. In overconstrained situations, we want SA-CIRCA to "self-adapt" by automatically deciding on appropriate tradeoffs between safety and goal-achieving behavior. For example, in the UCAV application we want SA-CIRCA to recognize that in some situations it may not be able to guarantee responses to all the possible types of threats (e.g., different surface-to-air missiles) and so it should account for only the most-probable types, building plans that have a non-zero risk but still achieve the overall goals of attacking some target.

To that end, we are developing a broad array of methods that the AMP can use to adjust and trade off the structure, goals, and safety concerns of the problem configurations sent to the CSM. This behavior characterizes the fundamental difference between the AMP and CSM. The CSM

does not make performance tradeoffs, it tries to build a controller that preserves safety against all the threats it is told about, and meets all the goals it is given. The AMP makes tradeoffs, altering the CSM problem configurations when the CSM is not successful. The AMP can reduce the number of goals the CSM is pursuing, reduce the threat possibilities the CSM must be concerned with, or perform a wide variety of more subtle modifications to the CSM problem configurations to make them easier or harder to solve.

To guide the application of these tradeoff methods, the CSM must provide useful feedback to the AMP. In the current implementation, the CSM either succeeds or fails, but no additional meaningful feedback data is provided. This abstract is meant to describe some concepts for improved CSM feedback, and their utility both for making better automatic tradeoffs and for simplifying user-level debugging of planning models.

Planner Feedback

There are several possible results of running the CSM on a particular problem configuration:

Complete Success — The CSM has successfully built and scheduled a reactive control plan that ensures safety and also achieves all of the goals. Note that in this situation it may still be useful to get feedback more significant than simply "success!" Telling the AMP how difficult the problem was can help the AMP optimize the allocation of reactive resources to the overall mission. The AMP may have initially posed a simplified problem configuration to establish a baseline, and feedback on the level of problem difficulty may help in maximizing the complexity of the next, revised configuration. For example, suppose the AMP has submitted a problem configuration for the entire ingress phase accounting for both ship-to-air threats and ground-to-air threats, since the ingress path flies over a littoral region before going "feet dry." Furthermore, suppose that the SSP was unable to build a safety-preserving plan for that full-complexity configuration, so the AMP has chosen to eliminate the lowest-probability threats (say, the ship-to-air missiles). This simplified configuration results in a successful plan, and the AMP can be satisfied with that. However, it might also get feedback information from the SSP showing that the plan now spends a disproportionate amount of effort *throughout* the ingress phase worrying about threats that are only present in the later, over-land part of the phase. This could trigger an AMP heuristic suggesting that the ingress phase be split into two separate phases, one over land and one over water. The resulting sequence of two smaller, phase-specific plans may yield a much higher probability of mission success. Note that while this example results in a phase partition relying on geometric characteristics, the concept itself is not tied to geometry or other domain-specific characteristics; the AMP can divide the problem space along any dimensions or feature values it chooses.

Partial Success — The CSM has built and scheduled a reactive control plan that ensures safety, but is not capable of achieving all of the goals. Again, feedback can help the AMP assess whether additional goals or complexity can be added to the problem.

Planning Failure — The SSP has examined the given problem and found an unresolvable failure; it cannot find a plan that will guarantee to avoid all of the possible failures. Feedback will help decide what types of tradeoffs should be made.

Scheduling Failure — Although the SSP was able to find a safety-preserving plan, the CSM Scheduler module was unable to schedule the planned reactions to meet all of the specified time constraints. Feedback about this type of failure can point to specific types of problem

modifications (such as relaxing worst-case timing concerns).

Timeout Failure — Either during the planning phase or scheduling phase, the CSM ran out of time for computing a controller, and no finished product was returned. Feedback in this situation can help the AMP understand the root cause of the excess problem complexity that defeated the CSM.

In addition to categorizing its results into these scenarios, the CSM should provide additional feedback to guide problem configuration adjustments. To date, we have identified the following types of feedback as feasible and potentially useful to the AMP.

Chokepoints — A chokepoint or bottleneck structure in the state space is a planned action that leads between two regions of state space having different characteristics (i.e., different likely threats). This structure indicates that breaking the single current problem configuration into two separate phases (with different problem configurations) may help make the smaller phases easier. Our research teammates at the University of Michigan are currently investigating ways of identifying these structures in SSP plans.

Difficult Threats — A single threat that occurs repeatedly throughout a region of the state space may make the problem difficult or infeasible. If the SSP can identify which threats are most difficult to handle, the AMP can use this information to guide restructuring of the phases, perhaps to avoid these particularly difficult threats entirely.

Schedulability Culprits — If the SSP builds a safety-preserving state-space plan but the reactions it requires cannot all be scheduled to execute within their timing constraints, the Scheduler may be able to provide general feedback about the level of overutilization (McVey *et al.* 1997). In addition, the Scheduler may be able to identify specific individual planned reactions or small sets of reactions that are placed under such tight time bounds that they bear primary responsibility for the inability to build a schedule.

As noted earlier, these types of feedback should be useful both for on-line self-adaptation and for helping the user (application engineer) debug domain descriptions during the off-line knowledge acquisition process.

Related Work

The notion that a planner must provide complex and useful feedback is relatively new; most planning work remains situated in fully-autonomous, non-adaptive settings. A notable exception is the work on mixed-initiative planning, which focuses on tying automatic planning to human-generated goals, and collaboratively making decisions in which humans remain involved. Mixed-initiative plan generation requires an ability to solve certain aspects of the planning problem automatically, while deferring other functions to the user. Complex query/response dialogue abilities allow the planner to interact with the user to clearly establish intent and mutual understanding (Allen 1983). Informative feedback from the planner to the human is essential to the mixed-initiative planning process. This suggests that results and approaches from mixed-initiative planning may map directly to our problems of AMP/CSP collaborative planning. And, while originally conceived of as useful for building plans collaboratively, the mixed-initiative concepts may be equally useful for acquiring correct planning domain models collaboratively.

References

Allen, J. 1983. Recognizing intentions from natural language utterances. In Brady, J. M., and Berwick, R. C., eds., *Computational Models of Discourse*. Cambridge, Mass.: M. I. T. Press.

McVey, C.; Durfee, E. H.; Atkins, E. M.; and Shin, K. G. 1997. Development of iterative real-time scheduler to planner feedback. In *Proc. Int'l Joint Conf. on Artificial Intelligence.*

Musliner, D. J.; Goldman, R. P.; Pelican, M. J.; and Krebsbach, K. D. 1999. Self-adaptive software for hard real-time environments. *IEEE Intelligent Systems* 14(4):23–29.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1993. CIRCA: a cooperative intelligent real-time control architecture. *IEEE Trans. Systems, Man, and Cybernetics* 23(6):1561–1574.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1995. World modeling for the dynamic construction of real-time control plans. *Artificial Intelligence* 74(1):83–127.