Appears in Working Notes of the AAAI Fall Symposium on Distributed Continual Planning
Orlando, FL, October 1998

Issues in Distributed Planning for Real-Time Control (Extended Abstract)

David J. Musliner and **Kurt D. Krebsbach** and **Michael Pelican** and
Robert P. Goldman and **Mark S. Boddy**
Automated Reasoning Group
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
{musliner,krebsbac,pelican,goldman,boddy}@htc.honeywell.com

Introduction

We are interested in extending the existing Cooperative Intelligent Real-Time Control Architecture (CIRCA) for real-time planning and control (Musliner, Durfee, & Shin 1993; 1995) into distributed applications such as the control of multiple unmanned aerial vehicles (UAVs). In such coarse-grain distributed applications, we envision multiple autonomous agents, each controlled by a CIRCA system, cooperating to achieve team goals in mission-critical domains. The DARPA-funded Distributed CIRCA project[1] began investigating many of the issues involved in distributed planning for real-time control. In this paper, we survey several of these issues and describe the current status of our continuing efforts to enhance CIRCA.

Background: CIRCA Review

Early work on CIRCA (Musliner, Durfee, & Shin 1993; 1995) focused on building an intelligent control system for a single agent, allowing that agent to provide real-time response guarantees while also using complex planning algorithms. The resulting architecture, illustrated in Figure 1, combines planning and scheduling modules that build guaranteed, executable plans with a real-time execution subsystem for predictably executing and enforcing the planned behavior.

CIRCA's original planning system builds reaction plans based on a world model and a set of formally-defined safety conditions that must be satisfied by feasible plans (Musliner, Durfee, & Shin 1995). CIRCA plans by generating a nondeterministic finite automaton (NFA) from user-supplied transition descriptions that implicitly define the set of reachable states. Beginning from a set of designated start states, the planner enumerates the reachable states and assigns to each state either an action transition or `no-op`. Actions are selected to *preempt* transitions that lead to failure states and to move towards states that satisfy as many goals as possible. System safety is guaranteed by planning action transitions that preempt *all* transitions to failure, making the failure state unreachable (Musliner, Durfee, & Shin 1995). CIRCA's plans are essentially state-space contingency plans that drive the system towards its goals and safely handle anticipated contingencies that cannot be avoided entirely.

Distributed CIRCA Goals

The Distributed CIRCA (D-CIRCA) project extends the concepts of guaranteed safety and predictable performance into multiagent domains such as cooperating teams of autonomous UAVs. D-CIRCA agents will communicate to allocate tasks and build executable real-time plans that achieve overall team goals. While executing their plans, D-CIRCA agents will respond to ongoing events in real-time, invoking safety-preserving reactions and/or triggering dynamic replanning tailored to the current context. D-CIRCA will enforce both the *logical correctness* of coordinated multiagent behaviors and the *timeliness* of those behaviors, ensuring that coordinated actions achieve their goals and preserve overall system safety.

This dual capability is distinctly different from typical distributed AI systems. Most DAI research eval-

[1] DARPA contract DAAK60-94-C-0040-P0006.

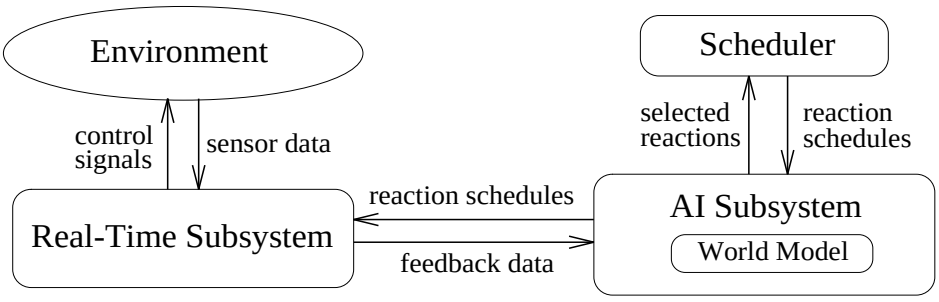

Figure 1: CIRCA combines concurrent planning, scheduling, and real-time execution.

uates collaboration and coordination methods based primarily on logical correctness and solution efficiency, ignoring the issues of behavioral synchronization and reaction timing required for *guaranteed* real-time performance by a multiagent system. Systems based on the D-CIRCA architecture can be applied to mission-critical distributed domains with confidence, and will provide plan-time feedback when the available multiagent resources are insufficient to deal with the anticipated behavior of the domain.

Distributed CIRCA Issues

Extending the single-agent CIRCA model to multiagent applications raises many challenging issues, some common to all distributed AI applications and some uniquely the result of CIRCA's commitment to predictable, guaranteed real-time performance. For example, the problems associated with cooperating concurrent planners have been investigated in other work (e.g., Partial Global Planning (Durfee 1988)), but issues of predictable asynchronous plan execution and performance guarantees across team behaviors have not. Some of the more challenging aspects of these issues are outlined below.

Scalability and Incomplete Knowledge

State space explosion problems are exacerbated in distributed systems, because the distribution adds additional, potentially-significant information to the domain (e.g., which agent knows which facts). Of course, the distributed agents cannot share complete information, or they may as well not be distributed. Hence distribution implies incomplete knowledge, which is problematic for a system trying to make performance guarantees.

Recently, the original CIRCA state-space planning algorithm was enhanced with the addition of Dynamic Abstraction Planning (DAP) (Goldman *et al.* 1997b), a technique for automatically deriving a nonhomogeneous abstract state space representation that helps reduce the state-space explosion inherent in CIRCA's contingency planning. DAP works by starting with a completely abstracted state space representation in which none of the available state features are included, and then it incrementally and nonhomogeneously adds state features (details) back into parts of the state space representation when required. Adding detail (or "splitting" an abstract state) can be required, for example, when the state description is not sufficiently refined to indicate whether a desirable action can, in fact, be executed (e.g., because the abstract state description does not specify values for all of the features in the action's preconditions).

The DAP technique was originally motivated by the need to build guaranteed control plans with incomplete information (due to distribution), as described in (Musliner *et al.* 1997). DAP can be seen as a way for individual planning agents to build interacting plans and decide, automatically, what information they need to share with each other (cf. (Wolverton & desJardins 1998)). The DAP planner may be presented with state features that represent both local and remotely-available information, and the abstraction planning algorithm will dynamically decide which of those features must be considered by the plan. Only those remote features that have been selected by DAP must be actually communicated between agents. Furthermore, because DAP supports nonhomogeneous abstraction, the communication about shared features can be started and stopped at various times during the plan, corresponding to those areas of the state space in which DAP has determined different features are important. DAP derives these information needs in a flexible and adaptive

manner that does not sacrifice any of the performance guarantees that the original CIRCA planner could enforce. Temporal latency requirements on planned actions that involve shared features also determine the frequency with which communications must occur.

Action Reflection

When multiple agents collaborate, mechanisms are required to support group knowledge and action reflection (e.g., one agent may have to consider the possible ramifications of another's actions). This reflection becomes particularly important when the agents must cooperate (or at least not interfere) to achieve mission-critical safety goals. Real-time constraints on action synchronization and the delays associated with inter-agent communication come into play as well, making the problem of having two CIRCA agents behave in a tightly-synchronized manner problematic. Note, however, that it is important to consider the types of domains for which CIRCA is appropriate: it is useful in discrete event control and decision-making, but not for continuous fine-motion control. Tightly-coupled agent activities such as carrying both ends of a long I-beam are better addressed using classical continuous control theory, rather than discrete event synthesis methods such as CIRCA. D-CIRCA should support performance guarantees about coarser-grained discrete synchronized behaviors such as coordinated target attack and mode switches. Reasoning about and planning with action reflection is an ongoing area of D-CIRCA research.

Runtime Communication

D-CIRCA agents must communicate predictably about their ongoing actions, to keep plan execution synchronized. We have modified the CIRCA RTS to allow multiple CIRCA agents to communicate with each other. The next step will be to use the DAP technique to derive what information must be shared between agents and what timing constraints must be imposed on that sharing. Given that information, we will enhance the planner to automatically build appropriate reactions that pass necessary information between agents during plan execution, with the required real-time communication guarantees.

Distributed Planning Paradigms

Multiple concurrent CIRCA planners must communicate, at planning time, to build coordinated plans. There are many ways to design this communication, corresponding to different levels of planner cooperation and synchrony. Currently, we are considering three simple alternatives:

Local Planning then Compare — In this simple approach, each CIRCA agent builds its plan without communicating to others, but perhaps using assumptions or agreements about how the other agents will plan (e.g., assigned roles could be agreed upon before plan generation, allowing an agent to ignore a contingency that is not relevant to its role). After the plans are built, they are combined together (see Section below) to determine if there are harmful interactions. This approach, while relatively simple to implement, has obvious disadvantages if the coordinated multi-agent plan is difficult to find (and requires a lot of backtracking).

Serial Planning — In this alternative, one agent constructs its plan and then sends it down the line to the next agent as a set of constraints on the second agent's planning process. This, of course, is less efficient, because the distributed planning is not in parallel, and also forces an ordering on which agents "get their way" first. On the positive side, this approach may recognize infeasibilities and induce backtracking without expending all of the group effort necessary to produce a complete set of candidate plans for all agents.

Asynchronous Coordinated Planning — The most complex alternative is to constantly trade partial plans, queries, constraints, and negotiations throughout the distributed planning process. According to our current thought experiments and manual planning simulations, in the D-CIRCA context this approach appears to have the disadvantage that it deteriorates into a lock-step process, in which parallel planners must reason at the same time (synchronously) about each shared state, thus negating many of the advantages of multiagent planning. Without lock-step synchronization, a CIRCA planner is unable to actually make performance guarantees or reason about the temporal aspects of a

particular state it wishes to consider, because the other planning agents may not have made their decisions about that state, and their decisions can alter the world model. However, we are continuing to investigate this approach (as the Holy Grail) in the hope that introspective methods such as DAP and the formal methods described below in Section may address the complex action reflection issues enough to allow asynchronous multiagent planning.

Breaking News on CIRCA: Model Checking for Time

The CIRCA temporal propagation model, discussed in detail in (Musliner, Durfee, & Shin 1995; Goldman *et al.* 1997a), has proven to be quite challenging to implement effectively. The current implementation in the DAP version of CIRCA's state space planner is notably incomplete, in that it eliminates from consideration a fairly large class of plans that are, in fact, safe and sound. Fortunately, we recently found that the CIRCA state/time model corresponds quite closely to models used in the field of formal methods concerned with "timed automata" and "model checking" (Alur, Courcoubetis, & Dill 1993; Alur 1998).

Model checkers are given a high-level automata description of a system, and are able to compare that model against various logical correctness requirements (e.g., unreachability of failure states). Timed automata models are state transition graphs annotated with temporal transitions and constraints associated with a finite set of *clocks*. All of the clocks increment synchronously, but can be independently reset to zero by selected transitions. Transitions themselves are instantaneous, just like event transitions in CIRCA models. Mapping a CIRCA state space model into a timed automata is a fairly simple matter of assigning different clocks to different CIRCA temporal transitions and translating the temporal transition timing constraints into timed automata clock constraints. Once this translation is complete, the timed automata model can be passed to existing, available model-checking code to determine whether failure is reachable and, if so, what path of transitions leads to failure (to guide CIRCA's intelligent backjumping).

We have just completed an automatic interface tying the DAP planner with the KRONOS (Yovine 1997) model-checking tool. Preliminary tests indicate that this hybrid planner is now able to build and verify plans in the general class that the old DAP temporal model implementation incorrectly rejected. Considerable additional work will be required to validate the hybrid planner's results, but we are optimistic that this approach is successfully leveraging existing formal verification methods. Future work will probably involve integrating the efficient temporal model reasoning back into the DAP planner for efficiency and flexibility.

An additional advantage of the model-checking methodology is that it includes efficient techniques for reasoning about the composition of multiple concurrent timed automata. In other words, the model checkers know how to find the (minimal) cross-product state space of multiple CIRCA state-space plans that will be executed on distributed agents. We are investigating whether this functionality can be used to reason, as efficiently as possible, about multi-agent interacting CIRCA plans generated in parallel.

References

Alur, R.; Courcoubetis, C.; and Dill, D. 1993. Model-checking in dense real-time. *Information and Computation* 104(1):2–34.

Alur, R. 1998. Timed automata. In *Working Notes of the NATO-ASI Summer School on Verification of Digital and Hybrid Systems.*

Durfee, E. H. 1988. *Coordination of Distributed Problem Solvers.* Kluwer Academic.

Goldman, R. P.; Musliner, D. J.; Boddy, M. S.; and Krebsbach, K. D. 1997a. The CIRCA model of planning and execution. In *Working Notes of the AAAI Workshop on Robots, Softbots, Immobots: Theories of Action, Planning and Control.*

Goldman, R. P.; Musliner, D. J.; Krebsbach, K. D.; and Boddy, M. S. 1997b. Dynamic abstraction planning. In *Proc. National Conf. on Artificial Intelligence*, 680–686.

Musliner, D. J.; Boddy, M. S.; Goldman, R. P.; and Krebsbach, K. D. 1997. The link between distributed planning and abstraction. In *Working Notes of the AAAI Fall Symposium on Model-Directed Autonomous Systems.*

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1993. CIRCA: a cooperative intelligent real-time control ar-

chitecture. *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.

Wolverton, M., and desJardins, M. 1998. Controlling communication in distributed planning using irrelevance reasoning. In *Proc. National Conf. on Artificial Intelligence*, 868–874.

Yovine, S. 1997. KRONOS: A verification tool for real-time systems. *Springer International Journal of Software Tools for Technology Transfer* 1(1/2).