

Multiagent Planning for Agents with Internal Execution Resource Constraints

Haksun Li

The University of Michigan, Ann Arbor
1101 Beal Avenue
Ann Arbor MI 48109-2110, USA
1-734-327-7907

haksunli@engin.umich.edu

Edmund H. Durfee

The University of Michigan, Ann Arbor
1101 Beal Avenue
Ann Arbor MI 48109-2110, USA
1-734-936-1563

durfee@umich.edu

Kang G. Shin

The University of Michigan, Ann Arbor
1301 Beal Avenue
Ann Arbor, MI 48109-2122
1-734-763-0391

kgshin@eecs.umich.edu

ABSTRACT

We study how agents can cooperate to revise their plans as they attempt to ensure that they do not over-utilize their local resource capacities. An agent in a multiagent environment should in principle be prepared for all environmental events as well as all events that could conceivably be caused by other agents' actions. The resource requirements to execute such omnipotent plans are usually overwhelming, however. Thus, an agent must decide which tasks to perform and which to ignore in the multiagent context. Our strategy is to have agents selectively communicate relevant details of their plans so that each gets a sufficiently accurate view of the events others might cause. Reducing uncertainties about the world trajectory improves the agents' resource allocation decisions and decreases their resource consumptions. In fact, our experiments over a sample domain show that, on average, 50% of an agent's initial actions are planned for states it can discover it will never reach. The protocol we develop in this paper thus discovers futile actions and reclaims resources that would otherwise be wasted.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence – coherence and coordination, multiagent systems.

General Terms

Algorithms, Performance, Design, Experimentation

Topics/Keywords

Coordination of multiple agents/activities & Coordination infrastructures, Action selection and Planning for agents

1. INTRODUCTION

In a dynamic, multiagent environment, an agent (e.g., a car driver) needs to be prepared to react to events arising naturally in the world (e.g., rockslides in the road) as well as those due to the actions of other agents (e.g., merging traffic). In this paper, we distinguish between two types of events. A natural event is one that *may* occur regardless of the activities of the agents in the world. We call it an *unconditional event*. In contrast, the other events occur because other agents explicitly choose to take some actions. We call them *conditional events*.

An ideal agent could manage its resources in order to be prepared to respond rapidly and correctly to all events of both types to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'03, July 14-18, 2003, Melbourne, Australia.
Copyright 2003 ACM 1-58113-683-8/03/0007...\$5.00.

guarantee hard real-time performance¹ [14]. We define *execution resources* as those that an agent needs to implement and operate its plan. They include the perceptual, effectual, and reasoning capabilities of the agent during execution. This is different from and complementary to the deliberation resources studied in the anytime literature. For example, given enough deliberation time, an ideal car driver could theoretically know exactly what to do in all possible circumstances (such as with an exhaustive policy of stimulus-response rules), and would be able to enact those reactions instantaneously. Real car drivers, however, cannot monitor their surroundings and react instantaneously. An agent's resources for following a policy can be constrained, perhaps because of sensory limitations (e.g., the driver cannot look between the front, the rear-view mirror, and the dashboard fast enough), or actuator limitations (e.g., the driver cannot steer the car, apply the emergency break, turn on the emergency flashers, and honk the horn all at the same time).² Realistic agents, with execution resource constraints, may have to give up on guaranteeing timely responses to some events so as to concentrate their resources on meeting more important demands (e.g., the driver might focus on traffic ahead at the expense of missing signs for an exit). Agents also might modify their behaviors to elongate reaction times for events (e.g., drive more slowly), adopt restrictions on their behaviors to eliminate some dangerous controllable events (e.g., drive on the right), and share information to help each other know what conditional events to be prepared for (e.g., use directional signals).

Our ongoing research efforts are devoted to studying the strategies by which an agent that has *execution resource constraints*, or a group of such agents, can efficiently make decisions about how to manage their resources so as to perform their tasks effectively despite the uncertainties inherent in the dynamic, multiagent environment. Elsewhere, we have described a strategy that an agent can use to compute the probabilities of various events occurring so as to devote its limited resources to being prepared for the most likely events [9]. The agent prioritizes its use of resources by planning for events in order of their occurrence probabilities, and unlikely events are ignored in case of insufficient resources. In a multiagent environment, the uncertainties about the plans of other agents impair the accuracy of those probability computations, which in turn degrades the quality of resource allocation decisions. Moreover, those

¹ A hard real-time agent is not a fast agent, but is an agent that must reliably meet the deadlines of all its tasks/actions. Otherwise, it is considered to have failed its mission completely.

² Agents might also have *external* resources that are shared with other agents, such as communication channels. In this paper, we will only focus on an agent's *internal* (execution) resources.

uncertainties prevent agents from “envisioning” similar future situations, so they may not agree on their expected interactions.

In this paper, we exploit the fact that many of the events of concern to one agent are conditional on the choices of other agents. Thus judicious communication among the agents can allow them to develop a more coherent view of the global activities. They can recognize which events are most important to prepare for, as well as which events are assuredly not going to occur. Note that we do not assume the agents are collaborating to solve a common problem. Instead, we assume the agents are loosely coupled but cooperative enough that they are willing to share information. They have different goals. Therefore, this paper is not about coordinating multiple agents to work together to achieve shared objectives, but rather it is about what information an agent, facing the uncertainties caused by other agents, should acquire from them to make better local resource allocation decisions as it otherwise works independently. Specifically, the contributions of this paper are that we model the problem of handling execution resource constraints in the context of multiagent planning, and from this context we derive a communication protocol that can be formally analyzed and coupled with heuristics to improve a real-time agent’s performance in dynamic, multiagent domains.

2. CIRCA BACKGROUND

The Cooperative Intelligent Real-time Control Architecture (CIRCA) models the interactions between actions and (conditional and unconditional) events explicitly, taking into account the real-time aspects of execution [1, 10]. CIRCA selects, schedules, and executes recognition-reactions assuming a resource-limited execution platform. There are two main subsystems in CIRCA, the Artificial Intelligence Subsystem (AIS) and the Real-Time Subsystem (RTS). The RTS executes the real-time control plans (see below) pre-computed by the AIS. Inside the AIS are the probabilistic planner and the scheduler. The planner generates a set of recognition-reactions, formally called Test-Action-Pairs (TAPs), by searching through the state space to determine the appropriate reactions for hazardous states. The period for each TAP is also chosen to ensure a sufficiently rapid response to preempt the emerging hazard.

The recognition test in a TAP is done by actively performing actions to collect data or monitor for the relevant aspects of the world, e.g., watching ahead of the car for looming obstructions. A reaction is only executed if the world matches the state description in the corresponding recognition test. As the progress of the world is uncertain, and dangerous events can happen sporadically, the RTS must check whether the reactions should be executed by continuously looping over the schedule of recognition tests frequently enough. The scheduler, based on the resource constraints of the RTS, schedules the set of recognition-reactions (TAPs) according to their periods chosen by the planner. A CIRCA *control plan*, composed of a scheduled set of recognition-reaction pairs, is therefore a cyclic (periodic) real-time schedule of TAPs. (TAPs are also referred to as “actions” in this paper.)

Because the scheduler is working with a RTS with limited resources, it could be that not all of the desired TAPs found by the AIS can be scheduled. For example, the processor utilization of each TAP is $u = \text{sum of the worst-case testing and execution times for the TAP divided by its period}$. When the sum of the utilizations of all TAPs exceeds 1, no schedule is possible. When this happens, CIRCA computes the probabilities, called state probabilities, of the agent reaching different states based on its

local state diagram. It finds a subset of the TAPs by removing those planned for states with state probabilities below a threshold. It keeps increasing this threshold until a schedulable subset is found. We call this the *unlikely state (cutoff) heuristic*³. It is a greedy search that lets an agent to devote its insufficient resources to responding to events that are most likely to occur [9].⁴

CIRCA attempts to minimize the probability of failure and maximize the probability of reaching goals. The failure probability may increase when the unlikely state heuristic is applied because some TAPs required to preempt possible (though less likely) hazards are removed. Consequently, the utility decreases. We distinguish between necessary and unnecessary actions (TAPs). *Necessary actions* are those that an agent may have to perform during execution to preempt some hazards. *Unnecessary actions* are those that the agent includes in its plan due to its ignorance about the plans of other agents. The agent will test them but never execute the corresponding reactions. However, it does not know this during planning unless the agents communicate. Using our terminology from Section 1, necessary actions are planned for unconditional events and some conditional events, while unnecessary actions are planned for those conditional events that will not arise⁵. Those unnecessary actions can be safely removed from the agent’s plan to free up resources without raising the failure probability. Therefore, an agent should not remove any necessary actions by using the unlikely state heuristic before it gets rid of all unnecessary ones. To determine which actions are necessary and which are unnecessary, however, requires a partially global view of the multiagent activities. The main theme of this paper is about how agents can exchange enough relevant information about their plans so that they can identify and remove enough unnecessary actions. They can also compute more accurately the true state probabilities.

The CIRCA state-space representation of the world is similar to STRIPS. It is constructed from a set of state propositions, called state features, and actions and events, called transitions, included as part of a planner knowledge base (KB). A state consists of a set of state features that describe the different aspects of the world. A state in the world model is created dynamically by applying a transition to its parent state. There are two types of transitions. Action transitions are explicitly controlled by the plan executor in the RTS, and thus only occur when selected during planning. Events outside the system’s control are modeled as temporal transitions, either innocuous temporal transitions (labeled *tt*) or deleterious temporal transitions leading to system failure (labeled *ttf*). A temporal transition, whether it is a *tt* or *ttf*, is described by a precondition, an effect (which we call a postcondition), and a probability function. The probability function describes the probability of a transition happening as a function of the time since it was enabled, independently of other transitions. When there is a *ttf* in a state, CIRCA plans a TAP to preempt the hazard. It also chooses the TAP period, to guarantee completion before

³ This heuristic assumes that all failures are equally bad. Otherwise, we could use the disutilities of the failures discounted by their state probabilities.

⁴ If different types of failure incur different penalties, then this can be coupled with the probability information so that the actions that prevent smaller expected penalties are preferentially removed.

⁵ Some conditional events may occur. Their corresponding actions are thus necessary.

the *tff* occurs, up to a pre-defined probability threshold. Preempting actions are called *guaranteed actions*.

A typical state diagram for planning is shown in Figure 3-1. It is a partial state diagram for an agent named FIGHTER. Action SHOOT-MISSILE-1 is a guaranteed action to preempt the *tff* BEING-ATTACKED. There is another type of action, called *reliable actions* that here we assume are also scheduled with real-time deadlines and thus utilize resources. However, they do not preempt any explicit failures. Action HEAD-TO-LOC1 is a reliable action that steers the fighter toward location LOC1.

A CIRCA agent in a multiagent environment builds on the single agent architecture. It is augmented with the ability to distinguish between private (local) features and public (shared) features. An agent's private features are those that no other agents are interested in, such as its current fuel level. Private features do not appear in the state diagrams of other agents. Public features are those features that more than one agent is interested in. An agent includes in its feature set only the public features that it cares about. Thus, different agents may have different sets of public features. It is through manipulating the public features that agents impact each other. For example, in Figures 3-1 and 5-1, COMM and ENEMY are public features shared by both BOMBER and FIGHTER, while HEADINGF and LOCF are private features that are accessible only to FIGHTER.⁷

Furthermore, a CIRCA agent in multiagent applications includes in its knowledge base some public temporal transitions and public action transitions of other agents (labeled *ttac*). Those transitions are the temporal transitions and actions that affect the public features it cares about. From the perspective of the agent, both the temporal transitions (*ts*) in the environment and the actions of other agents (*ttacs*) are alike. Both are beyond the control of the agent. In contrast, private temporal transitions and private actions of another agent do not include in their postconditions any public features but only private features. For instance, B:BOMB-1 and B:BOMB-2 are public actions of BOMBER in Figure 3-1, while the action HEAD-TO-LOC1 is private for FIGHTER. The temporal transitions FLY-TO-LOC0, FLY-TO-LOC1, and FLY-TO-LOC2 are private for FIGHTER. In Figure 5-1, there are temporal transitions of the same names but they are private for BOMBER. There is no public temporal transition in this example. To date, CIRCA treats transitions independently, modeling concurrent activities only as a series of separate transitions. It does not model concurrent activities whose effects when taken in concert differ from their additive effects.

Differentiating between private and public features lets an agent model only the relevant aspects of the world. The agent does not need to know the entire plans of other agents, but only those parts affecting the public features. Since a state represented by an agent does not include the private features of other agents, it may correspond to a group of states represented by those agents. As the number of states is exponential in the number of features, this abstraction significantly reduces planning complexity and state diagram sizes.

3. REACHABILITY ANALYSIS

To succeed in a multiagent environment, a rational agent needs to envisage what actions other agents might take, and choose its own actions based on these predictions. Other agents cannot be expected to know what constitutes failure for the agent (which

⁷ The special feature "FAILURE" for an agent is always private.

could in part be based on its private features), nor can they promise not to affect negatively the public features. To play it most safe, the agent must consider and plan for all states that it foresees due to the transitions in the environment (unconditional events) as well as all possible actions executed by other agents (conditional events). Regardless of whether a state is reachable by a sequence of temporal transitions (*ts*) or a sequence of others' actions (*ttacs*) or a combination of both, the agent needs to expend resources to schedule an action to preempt any hazards in the state. We call such an analysis a *reachability analysis*. A state diagram generated by a reachability analysis is called a *reachability graph*. Figure 3-1 is the reachability graph of a fighter patrolling nearby two locations after any potential activities there by BOMBER. It includes in its planning, hence the state diagram, all public actions by BOMBER.

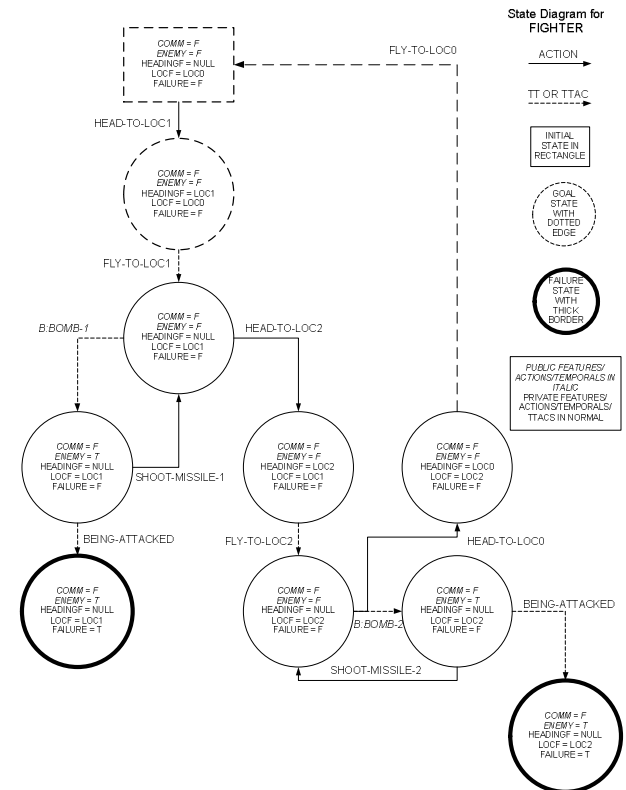


Figure 3-1: The (Partial)⁹ Reachability Graph for FIGHTER¹⁰

However, just because another agent is capable of taking an action does not mean that it will take that action, meaning that anticipating all possible actions on the part of other agents requires an agent to prepare for states that might never arise. We call those states *unreachable states*. In contrast, *reachable states* are those states that the agent *may* reach during execution. Unreachable states are included in a reachability graph only because of ignorance. The actions planned for the unreachable states are unnecessary actions and can be removed if they can be recognized as such.

⁹ For space reasons, this graph omits some B:BOMB-1 and B:BOMB-2 transitions. They do not affect the final plan.

¹⁰ All actions are either guaranteed (e.g. SHOOT-MISSILE-1) or reliable actions (e.g. HEAD-TO-LOC1). This holds true also for Figure 5-1.

Forming control plans based on such a straightforward reachability analysis suffices only when an agent has enough resources to handle all contingencies. In this ideal case, it can disregard (and be proudly ignorant about) the plans of other agents. Regardless of which of the known possible actions the other agents choose, and which potential states it may thus encounter, it can always execute the proper reactions fast enough to avoid failures. On the other hand, the agent may be unable to schedule all the actions for all the states that it thinks it may encounter due to over-utilization of resources.

This paper views the problem of how an agent decides how much it needs to know about the plans of other agents as a type of multiagent planning problem. Although an agent might not be able to change the actions chosen by another agent, knowledge about those choices can reduce uncertainties for its own planning, thus allowing better resource allocation decisions to be made. Our strategy to solve the limited resource allocation problem for a group of independent but communicating agents is therefore to have them first construct the worst-case resource consumption scenarios, i.e., the reachability graphs. These can inevitably include states that are reachable conditional upon the execution of certain actions by other agents. The agents can then incrementally exchange details about their plans to each form more precise views of the future. While an agent must expend resources to prepare for all unconditional events, it can avoid wasting resources on conditional ones it learns for sure will not happen.

4. CONVERGENCE PROTOCOL

We have developed a protocol that allows agents to exchange information about the intersecting parts of their plans. The agents can reduce resource consumptions by lowering the uncertainties in modeling the events caused by each other, in case of insufficient resources. Agents identify unreachable states in the state diagrams and eliminate the associated actions from their tentative plans. For instance, in Figure 3-1, when FIGHTER is unable to guarantee real-time performance to both SHOOT-MISSILE-1 and SHOOT-MISSILE-2, knowing which location BOMBER is not going to removes the corresponding shoot-missile action from its plan. We assume that the agents perform the protocol after they have locally formed their reachability graphs and have selected all actions they would like to take (as if there were no resource constraints). The protocol is described using C-like code in Figure 4-1.

An uncertain point of interest in the graph is a combination of a state and a set of mutually exclusive *ttacs*. A set of *ttacs* in that state corresponds to the alternative action choices of another agent out of which it will choose one or none. As multiple agents can plan for a state, there could be multiple sets of *ttacs*. When choosing an uncertain point, the agent picks a state as well as another agent planning for the state, i.e., a particular set of *ttacs*. Each *ttac*, if planned by another agent, is a *realizable* branch in the state diagram that consumes certain resources. A branch from a state is a subgraph from a transition in that state. On the other hand, if a *ttac* is not planned, the branch is not realizable during execution. It can safely be pruned from the state diagram. All actions planned for the descendant states of the branch are subsequently removed (unless the states can be reached via other viable transition paths), making scheduling easier.

The Convergence Protocol inquires about the states in descending order of the estimated resource consumption reduction *if* the unplanned actions of another agent in a state are removed. We postpone the discussion of picking the next uncertain point (labeled *) until Section 6.

```
Inquiring agent () {
    Choose the uncertain point that gives the biggest estimated
    utilization reduction; /*
    Ask the corresponding agent which action(s) it will take;
    Upon receiving an answer, update the state diagram and drop
    unnecessary actions from the local plan;
    Loop until either the resource constraints are satisfied or all
    uncertain points are examined;
}
Answering agent () {
    When (being asked by another agent about an uncertain point) {
        Identify the corresponding state(s) in the local graph;
        Reply with the action(s) (or none) planned for the state(s);
        Record the agent's name with the state(s);
    }
    If (an action is removed from its state diagram/plan) { /**
        Inform all agents with names recorded with the state that the
        action is no longer planned for that state;
    }
}
```

Figure 4-1: The Convergence Protocol

If all features are public, then all agents have the same features. There is a one-to-one correspondence between the states in any two agents' state diagrams. The answering agent can uniquely identify the state (if it exists) that the inquiring agent is asking about. As there can only be one planned action in a state for a CIRCA agent, it will reply with the single action planned (if it exists) for the inquired state. All but one *ttac* in that state can be eliminated in the state diagram of the inquiring agent. However, agents that distinguish between public and private features (Section 2) do not usually share all features. Instead of sending all its state features, the inquiring agent sends a more abstract state description, consisting of only public features and their values. The answering agent may have more than one state in its local state diagram that matches the abstract description. It must reply with all the actions planned for those matching states. Accordingly, the inquiring agent must plan for more than one *ttac* in that state.

It is important to point out that an agent is asking which action another agent *would* execute if that other agent reaches a state. The answer can be found simply by looking up the TAPs in its reachability graph. If the agent is asking whether another agent *will* execute an action, that other agent may not be able to answer at all. The answer depends on its certainty about which states it will encounter during execution. It cannot be certain of the reachability without the complete plans from all other agents. As the agents are refining their plans concurrently and asynchronously, there are no such complete plans before the multiagent planning process finishes. We are in the process of extending the protocol so the agents can start exchanging useful details to let other agents prune their search spaces earlier on.

Moreover, the reachability graphs of agents can be very different. Some states may be in some graphs but not in others. In fact, agents typically do not even have the same sets of state features. Some agents may think that certain states are reachable while others may think otherwise. For the example in Figure 3-1, FIGHTER thinks that enemies appear at $LOCF == LOC2$, while BOMBER may not concur. In case of insufficient resources for

agents to handle all states in their respective state spaces, the Convergence Protocol allows them to gradually and cooperatively discover which states are indeed reachable by exchanging only the relevant details of their plans. They are able to refine their individual plans by converging toward a set of commonly agreed-upon reachable states until their resource constraints are satisfied. Note that they do not need to agree completely on the set of reachable states so all agents have the same state diagrams. The beauty of our protocol is that they only have to use it until they can schedule all their remaining actions after pruning enough unnecessary ones. Even if an agent ends up allocating some resources to situations that cannot possibly arise due to incomplete knowledge, it is still acceptable if the agent has enough resources relative to all the demands. If we require the agents to have identical state diagrams, other (more costly) protocols, such as [6], might work better.

If an agent fails to schedule for all remaining actions after completing the Convergence Protocol, it resorts to the unlikely state heuristic (Section 2) to remove the most unlikely (but possibly necessary) actions. Even so, the agent still benefits from engaging in the partial plan exchange by having a more informed model of the plans of other agents. Otherwise, it might have removed more necessary actions in order to satisfy its resource constraints yielding an even lower agent utility.

The part of the protocol marked by (**) speeds up the pruning process of other agents by broadcasting to those who have previously inquired about (and therefore have shown interest in) a state if an action is no longer planned for that state.

Clearly, the Convergence Protocol terminates. In the worst case, it terminates after all agents examine all uncertain points in their reachability graphs. As the protocol examines the states exhaustively, an agent will move toward satisfying its resource constraints by pruning the unnecessary temporal trajectories. If the agent starts with sufficient resources to prepare for all that are necessary, then it is guaranteed to find a plan that schedules all the actions. An agent's utility is not compromised if it can schedule for all actions after running the protocol because the protocol always removes actions that are assured to be unnecessary. Its utility decreases only when it drops some necessary actions by raising the probability threshold. Therefore, the order of inquiries agents make about the action choices of others is irrelevant to its utility. Similarly, the order of communication among agents is also irrelevant to the utility. Regardless of when an agent asks about a state, it will always get the most recent information about that state (by **) before it has to raise the probability threshold. In essence, what the agents ultimately end up finding as the definitely reachable states will be unaffected by the order in which states not in the intersection are pruned.

5. DEMONSTRATION

We now continue our story in Figure 3-1. We demonstrate how our strategy lets agents cooperatively create a more coherent picture of the global activities. By reducing uncertainties in modeling other agents' plans in case of insufficient resources, agents may decrease their resource consumptions. A bomber (BOMBER) is given a mission to attack one of the two locations. After it attacks a location, enemy aircraft could arrive to retaliate. A fighter (FIGHTER) is assigned to patrol around the locations to repel the enemy aircraft whenever they arrive. Also, if FIGHTER requests a response from BOMBER, BOMBER has to report its status at LOC2. The complete state diagram (reachability graph) for BOMBER is shown in Figure 5-1.

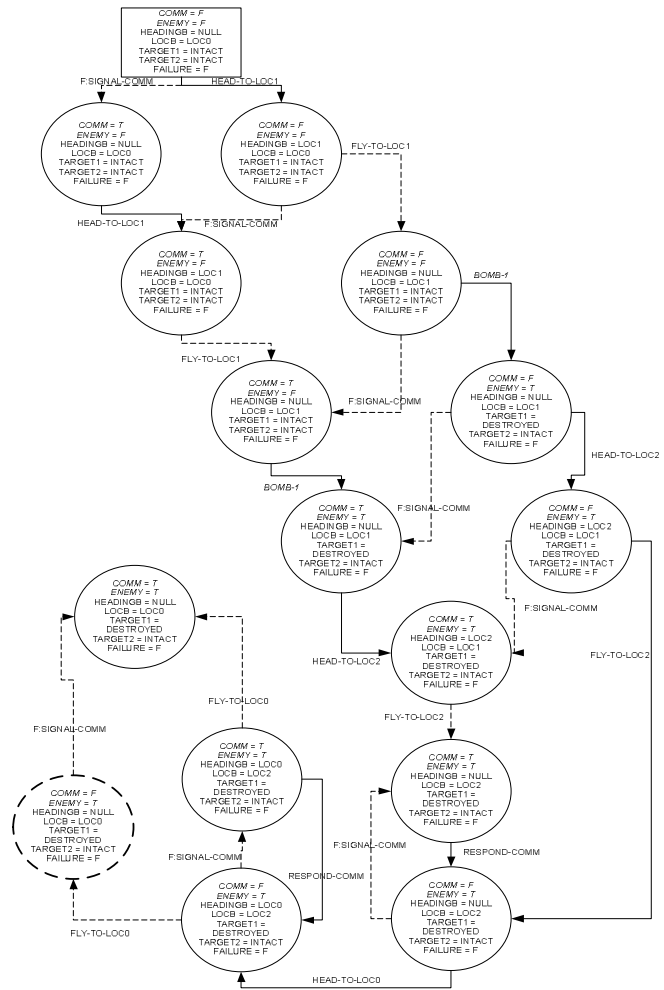


Figure 5-1: The Reachability Graph for BOMBER

Both FIGHTER and BOMBER (Figures 3-1, 5-1) have 5 actions to schedule if they do not know about the plan of the other agent. Suppose the resource constraints are simplified such that each agent can schedule only 4 TAPs. Both agents exceed their capacities. By running the Convergence Protocol, FIGHTER asks BOMBER what actions it plans when ((COMM == F) && (ENEMY == F)).¹¹ BOMBER replies that it is going to do only BOMB-1.¹² Then FIGHTER can safely remove the children of the *ttac* B1:BOMB-2, hence the action SHOOT-MISSILE-2, from its state diagram and tentative plan. Likewise, BOMBER will discover that FIGHTER does not signal BOMBER to respond at LOC2. So, BOMBER can safely remove RESPOND-COMM from its plan. As a result of communication using the Convergence Protocol, both of them have only 4 TAPs left for scheduling. Both agents in this case can satisfy their local resource constraints: neither resorts to using the unlikely state heuristic to drop actions. So, no agent's utility is compromised.

6. CHOICE FUNCTIONS

One implicit assumption in our work is that exchanging information costs something. Otherwise, agents would simply

¹¹ Recall agents communicate only public features and actions.
¹² BOMB-1, BOMB-2, and RESPOND-COMM are the only public actions for BOMBER.

dump their entire plans or reachability graphs to other agents. Also, not only does an agent need to identify what to communicate but it also has to answer inquiries made by other agents. Therefore, agents should reduce their planning costs by minimizing the number of inquiries made.

The part of the Convergence Protocol marked by (*) applies a heuristic *choice function* to choose what the next best uncertain point is. The more effective a choice function is, the sooner the protocol leads an agent into finding a satisfying plan, and the less computation and communication the agent does. In general, the states that are closer to the initial states and have more *ttacs* (alternative actions of one or more other agents) should be examined with priority. These states tend to have more downstream children, hence more planned actions taking up resources. So, they tend to free up more resources *if* removed.

We have considered the following heuristics. They are listed in order of increasing complexity.

1. Random Choice Function: The agent inquires about a random state. This heuristic serves as a benchmark.
2. Sequential Choice Function: The agent inquires about the states in the order of their expansions during planning. If the state expansion is a breadth-first search, then the states closer to the initial states tend to be examined before others.
3. Distance Choice Function: The agent inquires about the states in ascending order of their distances to the initial states. The distance of a state is the minimal number of transitions that take the agent from any of the initial states to the state.
4. Load Choice Function: The agent inquires about the states in descending order of their numbers of actions per branch. The idea is to prune actions as fast as possible. Not all actions have the same utilizations.
5. Utilization Choice Function: The agent inquires about the states in descending order of utilization per branch.

Each of these choice functions allows an agent to find satisfying plans at different converging speeds. Figure 6-1 shows a sample relation between the utility¹³ and the communication cost using the Convergence Protocol with the simplest choice functions (repeated experiments). The communication cost could be computational, e.g., time, power, or bandwidth, and/or non-computational, e.g., the risk of eavesdropping. In any case, we simply measure cost as the number of inquiries exchanged. Note that these curves are monotonic non-decreasing functions, meaning that the agent never loses utility by decreasing uncertainties about the global activities. Although we typically use the Convergence Protocol offline, its anytime nature allows it to be employed online as well. In that case, the choice function becomes particularly important. Adopting an appropriate choice function can dramatically reduce the costs. We discuss the converging rates of our candidate functions in Section 8.

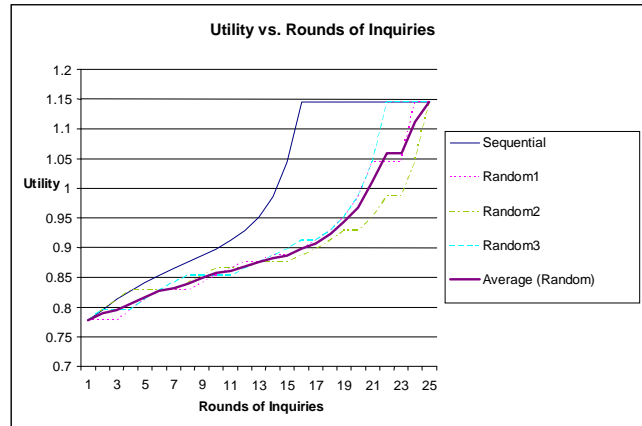


Figure 6-1: Varying Converging Speeds¹⁴

7. EVALUATION

To evaluate the merit of the Convergence Protocol, we have generated a set of random domains. Each domain has a random number of agents from 2 up to a maximum of 10. Each agent has its own knowledge base. The knowledge base has 7 private and public binary features (T/F) total. The number of public features in a domain is random. It measures how tightly coupled the agents are for that domain, i.e., how many features they have in common. As any public feature is shared by all agents, the knowledge bases for any given domain have the same number of public features. There are 15 private and public actions combined, and 7 private and public temporal transitions combined for each agent. The compositions are random. The actions and temporal transitions are generated such that they invert the values of a random number of features. All knowledge bases for a domain contain the same set of public temporal transitions and public actions.

We have generated 1126 agents (KBs) for 402 domains with which we perform our experiments. The number of agents that are able to schedule for all their TAPs *before* running the Convergence Protocol is 202 (12.42%). The number of agents that are able to schedule for all their TAPs *after* running the protocol is 704 (43.30%). In other words, 502 (30.87%) agents *become* able to schedule for all their TAPs. For those agents that still fail to schedule for all actions, they, nonetheless, drop fewer necessary actions (by raising the probability threshold) than they otherwise would have needed to. For our experiments, the reduction in the number of necessary actions dropped is on average 59.55% with a standard deviation 39.85%. As a result, the agents' utilities are not as compromised as they would otherwise be without running the Convergence Protocol.

Furthermore, we measure how effective the Convergence Protocol is by what we call action effectiveness and state effectiveness. *Action effectiveness* is the percentage of unnecessary actions removed by the protocol. Likewise, *state effectiveness* is the percentage of unreachable states included in an agent's reachability graph but removed by the protocol. The average effectiveness is 53.74% and the standard deviation is 29.45%.

¹³ The utility is a weighted function of the probability of the agent achieving its goals and the probability of remaining safe.

¹⁴ We graph the results for only the sequential and the random heuristics. For this simplified illustrative problem, the more sophisticated heuristics are overkill and give the same results as the sequential heuristic. But, they usually perform differently depending on the domains. We show this in Section 8.

Table 8-1: Choice Function Efficiencies

Choice Function Efficiency	Random (Action)	Random (State)	Sequential (Action)	Sequential (State)	Distance (Action)	Distance (State)	Load (Action)	Load (State)	-Load (Action)	-Load (State)	Utilization (Action)	Utilization (State)
average	0.26	1.72	2.14	12.19	2.40	13.88	0.59	3.39	0.17	1.16	0.76	4.45
standard derivation	0.48	2.78	4.30	23.86	4.40	24.66	1.65	8.22	0.16	0.73	2.22	12.23

Table 8-2: Communication Efficiencies

Communication Efficiency	Random (Action)	Random (State)	Sequential (Action)	Sequential (State)	Distance (Action)	Distance (State)	Load (Action)	Load (State)	-Load (Action)	-Load (State)	Utilization (Action)	Utilization (State)
average	0.09	0.61	0.15	0.98	0.17	1.13	0.13	0.75	0.07	0.53	0.11	0.73
standard derivation	0.09	0.48	0.19	1.06	0.22	1.22	0.98	1.58	0.06	0.35	0.12	0.68

Similarly, action effectiveness has an average of 51.74% and standard deviation of 35.84%. The data suggest¹⁵ that more than half of the resources, at least in our sample domains, are often wasted when an agent is ignorant about the plans of other agents. When the agents include in their planning all conceivable interactions based on all public actions of other agents, very often more than 50% of the states that they think they may encounter are in fact not reachable. Communication is therefore very important for resource-limited agents. Our protocol allows the agents to remove those states from their state diagrams.

8. CHOICE FUNCTION EFFICIENCIES

To study the convergence speeds and overheads of various choice functions, we define choice function efficiency and communication efficiency. *Choice Function Efficiency* is measured by the number of actions (states) removed per inquiry. Based on the same set of experimental domains described in Section 7, we get the statistics in Table 8-1.

As we had expected, inquiry priority measured in terms of utilization (Utilization) is more efficient than in terms of number of actions (Load). Surprisingly, the efficiency of a choice function does not necessarily increase with its complexity. The Load and Utilization choice functions are both worse than Sequential and Distance choice functions, but are still better than Random. The reason for the poorer performances of both Load and Utilization heuristics is that they are considering the number of actions and utilization *per branch*. Even if a state has many actions in its children, those actions could concentrate in one of the many branches of the state. The inquiry priority of the state, i.e., utilization per branch, is therefore “diluted” by those branches with fewer actions.

Moreover, the data confirm our hypothesis that the states closer to the initial states should be inquired about with higher priority (Distance is the best among the 5 heuristics). They are the states that have more actions (total number of actions rather than actions per branch) and higher utilizations. An ancestor of a state has at least as many actions as the state itself. As the choice functions have different computation and communication costs, we can trade computation time with communication overheads.

Although choice function efficiency tells us how efficient a choice function is at getting rid of unnecessary actions (states), it does not tell us how costly communication is using the Convergence

Protocol. Not only does an agent have to send one message per inquiry, it also has to send messages to answer all inquiries from other agents and update them of any removed actions. Even if the agent itself has sufficient resources so that it never asks questions, it may still have to answer a lot of inquiries made by other agents. We would like to measure the overhead for an agent to communicate using the Convergence Protocol as a member in a group. We define *communication efficiency* as the number of actions (states) removed per message sent. The communication efficiencies for different choice functions are shown in Table 8-2.

Both tables show that action efficiencies are lower than state efficiencies. This is because an action is often planned for more than one state for CIRCA. Only when all these states are pruned from the state diagram can the action be safely removed from the plan. Also, some pruned states have no associated actions. These states do not change the utilizations of plans. However, they give the agents better ideas of what states they may visit and how the world may evolve. The agents can also more accurately compute state probabilities. Accordingly, they can better apply the unlikely state heuristic to drop the least likely necessary actions in case of very stringent resource requirements. As discussed in Section 4, all choice functions give equal performance in terms of agent utility and generate equivalent final plans for agents.

9. RELATED WORK

Most existing research on cooperative multiagent planning has been focusing on generating compatible plans to avoid negative interactions among agents. Techniques, such as negotiation [11], plan merging [7], multiagent MDPs [2] and (Generic) Partial Global Planning [5], have been developed to resolve conflicts. These algorithms and protocols focus on how the plans of some agents constrain the plans of other agents, but assume that each agent has sufficient resources to carry out its plan once coordinated. Our work does not make this assumption.

An important consideration in multiagent planning centers around how much an agent knows ahead of time about the other agents. At one extreme, it might know nothing; for many of the approaches listed above, agents must exchange plan information to begin to model each other. At the other extreme, coordination approaches such as Social Laws [12] assume (in the offline case) that agents know everything they need to know about each other right from the beginning: any plan an agent makes that adheres to the social laws is assured to be coordinated with the other law-abiding agents. In the middle are approaches where agents have some knowledge (for example, the organizational roles of others) ahead of time, but need to exchange information to acquire more situation-specific details [4]. Our work resides in this middle

¹⁵ We say “suggest” because these statistics are gathered over a sample domain and have large variances. They do not necessarily apply to other unknown applications.

ground, because we assume that our agents know everything about what others might do under different eventualities. But, in contrast with work where subsequent communication is to add details to enlarge an agent's view of how the world might unfold, we instead view the purpose of communication as helping agents rule out some of the choices that they had anticipated others might make.

A wide variety of agent communication protocols have been developed over the years, ranging from the early days of the Contract Net [13] to recent work on conversation policies [8]. The protocol we employ involves simple query-response interactions. Of particular relevance is the work of Conry [3], where the protocol was coupled with a decision-making strategy such that an agent would decide to send a query when it got "stuck" in terms of not being able to make further headway on solving its problem without more input. Our protocol works in a similar vein, by assuming that an agent will only request information from others when it cannot make local action choices that are guaranteed to achieve its objectives without violating its execution resource constraints. In other words, each of its action choices utilizes some fraction of its internal execution resources to carry out, and the principal constraint an agent faces is that the set of actions must be *schedulable* (Section 2). The choice functions in this paper and in Conry's serve the same role: to speed up the searches by trying to identify the "right" information to acquire.

10. CONCLUSIONS

We see in the demonstration from Section 5 how our strategy helps agents find satisfying plans in a distributed manner. The agents first construct their reachability graphs and then iteratively refine their plans using the Convergence Protocol. The agents cooperatively determine the set of states for which they need to react by exchanging partial plans to generate more coherent views of their activities.

Our experiments suggest that it is often worthwhile for agents to exchange partial details of their plans if they have inadequate internal execution resources. More than 50% of the resources may be wasted when they are ignorant about the plans of others. Our data show the tradeoffs between complexity and efficiency of various choice functions. When communication is costly but computation time is not, we can adopt a choice function with a higher efficiency to minimize the overhead, and vice versa.

One major drawback of this approach is that it requires the agents to construct the entire reachability graphs before they start to talk. A reachability graph is an explicit representation of the complete search space, not only of the agent itself but extended with its knowledge of external events that might happen. It could be too demanding on the part of the agent if the entire graph is to be generated and kept in memory. We are in the process of improving the Convergence Protocol so that the agents start communication earlier as they construct their reachability graphs. This can cut down on time spent on searching unreachable states, and space storing the unused portions of their plans. Moreover, as the agents interleave plan/reachability graph construction and communication, we expect that they can generate better plans than they do now.

11. ACKNOWLEDGMENTS

This research was supported in part by DARPA/AFRL Contract F30602-00-C-0017. We also thank Dmitri Dolgov, Dave Musliner, and the anonymous reviewers for their helpful comments.

12. REFERENCES

- [1] Atkins, E. M., Abdelzaher, T. F., Shin, K. G. and Durfee, E. H. Mar-Apr 2001. Planning and Resource Allocation for Hard Real-time, Fault-Tolerant Plan Execution. *Journal of Autonomous Agents and Multi-Agent Systems*.
- [2] Boutilier, C. 1999. Sequential Optimality and Coordination in Multiagent Systems. *IJCAI-99*.
- [3] Conry, S. E., MacIntosh, D. J., and Meyer, R.A. 1990. DARES: A Distributed Automated Reasoning System. *Proceedings of AAI-90*, pp. 78-85.
- [4] Decker, K. and Lesser, V. 1995. Designing a Family of Coordination Algorithms. *ICMAS-95*.
- [5] Durfee, E. H. and Lesser, V. R. September 1991. Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation. *IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on Distributed Sensor Networks*, SMC-21(5):1167-1183.
- [6] Ephrati, E., Pollack, M., and Rosenschein, J. S. 1995. A tractable heuristic that maximizes global utility through local plan combination. In Lesser, V., editor, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 94-101, San Francisco, CA. MIT Press.
- [7] Georgeff, M. 1983. Communication and Interaction in multi-agent planning. *Proc. of the Third National Conference on Artificial Intelligence (AAAI-83)*, pp. 123 – 129.
- [8] Kumar, S., Huber, M. J., Cohen, P. R., and McGee, D. R. 2002. Toward a Formalism for Conversation Protocols Using Joint Intention Theory. *Computational Intelligence Journal (Special Issue on Agent Communication Language)*, Brahim Chaib-draa and Frank Dignum (Guest Editors), Vol. 18, No. 2, pages 174-228.
- [9] Li, H, Atkins, E., Durfee, E. H. and Shin, K. G. August 2001. Practical State Probability Approximation for a Resource-Limited Real-Time Agent. *Proceedings of the IJCAI-01 Workshop on Planning with Resources*.
- [10] Musliner, D. J., Durfee, E. H., and Shin, K. G. 1995. World Modeling for the Dynamic Construction of Real-Time Control Plans. *Artificial Intelligence*, vol. 74, pp. 83-127.
- [11] Shintani, T, Ito, T., and Sycara, K. 2000. Multiple Negotiations among Agents for a Distributed Meeting Scheduler. *In Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS'2000)*.
- [12] Shoham, Y. and Tennenholtz, M. February 1995. On Social Laws for Artificial Agent Societies: Off-Line Design. *Artificial Intelligence*, Vol. 73, Numbers 1-2, pp. 231-252.
- [13] Smith, R. G. December 1980. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104-1113.
- [14] Stankovic, J. A. October 1988. Misconceptions about Real-Time Computing: A Serious Problem for Next-Generation Systems. *IEEE Computer*, vol. 21, no. 10, pp.10-19.