Multiagent Planning with Internal Resource Constraints

Haksun Li, Edmund Durfee, Kang Shin
EECS Dept., University of Michigan, Ann Arbor MI 48109
{haksunli, durfee, kgshin}@umich.edu

Abstract

This paper studies the causes of over-utilization of the resource capacities of a group of cooperative resource-limited agents and what they can do to reduce their resource consumption. We view the problem of how an agent decides what tasks to execute and what to ignore as a type of multiagent planning problem. An agent in a multiagent setting has to prepare for all states it may reach as a result of its own actions, the environment transitions, as well as the actions potentially executed by other agents. Intuitively, the more information it knows about the plans of the other agents, the better it can allocate its resources for various tasks. Indeed, in our experiments over a particular sample space, on average, 50% of the actions are planned for the states it will never reach when an agent is completely ignorant about the plans of others. We have developed a protocol to allow agents to efficiently find out the relevant information about the plans of others. As the agent increases its level of awareness of the others' plans, it can better identify the unreachable states to avoid spending resources on them.

1. Introduction

Most research on cooperative multiagent planning has been focusing on generating compatible plans to avoid negative interactions among agents. Techniques, such as negotiation (Zhang, Podorozhny, Rodion & Lesser 2000, Shintani, Ito & Sycara, 2000), plan merging (Georgeff 1983, Ephrati, Pollack & Rosenschein 1995), hierarchical planning (Clement 1999), voting (Ephrati 1993), multiagent Markov Decision Processes (Xuan, Ping, Lesser, et al. 2000, Boutilier 1999) and Partial Global Planning (Durfee 1991), have been developed to resolve conflicts. These algorithms and protocols, while focusing on generating compatible plans and resolving contentions over shared resources, assume that each of the agents has sufficient resources to carry out its plan once coordinated.

Our research is looking at the cooperative multiagent planning problem from another angle. We are interested in studying what a resource-limited agent in a multiagent setting can do when its plan violates its own internal resource constraints. For example, when an agent must periodically execute a number of tasks with deadlines, the utilization of each internal resource, e.g. computation time, sensors, actuators, of these tasks must be less than 1 (Krishna & Shin 1997). We view the problem of how an agent decides what to attend to and what to ignore as a type of multiagent agent planning problem. The agent must be ready to react to situations that arise due to the combined influences of the actions it chooses to take, of the dynamic changes to the world caused by the external environment, and of the actions that other agents take that further alter the state of the world. The agent clearly has control over its own actions, but is to a large extent at the mercy of how the environment and other agents impact the state of the world. In this paper, we explore the possibility that, even when an agent cannot change the action choices that another agent might make, knowledge about what those choices can serve to reduce uncertainty and allow an agent to allocate its own limited resources better.

Specifically, in a state-space planner like CIRCA (Musliner 1995, Atkins 1999), an agent has to prepare for all states it may reach as a result of its own actions, the environment transitions, as well as any of the possible actions that other agents are capable of taking. Just because an agent is capable of taking an action, however, 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. Indeed, in our experiments over a particular sample space, on average, 50% of the states that an agent thinks it may reach can in fact be unreachable. In other words, the agent could waste 50% of its resources on watching out for and being prepared to react to states it will never encounter when it is completely unaware of the plans of other agents. Intuitively, as the agent increases its level of awareness of the others' plans, it can better identify the unreachable states to avoid spending resources on them.

Our strategy to solve this resource constraint problem is therefore to have the agents find out the relevant information about the plans of other agents. Based on this additional knowledge, the agents can refine their tentative plans to remove the unnecessary actions planned for the unreachable states. They should continue doing so until they satisfy their resource constraints or until they are fully aware of all relevant planned actions of the other agents. In other words, our principal contribution in this paper is a protocol that allows agents, which start by considering all possibilities, to incrementally exchange enough information to "relax" the initially over-constrained plans by removing those unnecessary tasks.

In this paper, we compare our approach with some related work in Section 2. We describe the context in which this work is done (Section 3) and the manner in which an agent identifies the potential reachable states for which it must be prepared (Section 4). We then present our protocol (Section 5) that allows agents to learn the relevant details of the plans of other agents by means of selective communications so that the agents can prune their state spaces to remove the unreachable states. The efficiency of this protocol will be dependent on the

choices that the agents make about what information to gather from others (Section 6). Through analysis, demonstration (Section 7), and empirical evaluation (Section 8), we show that our protocol is guaranteed to terminate and can dramatically improve the expected performance of the agents that employ it. We conclude with a summary of this work and outline directions for our further research (Section 9).

2. Related Work

Our resource constraint problem can be cast as a constraint satisfaction problem (CSP). The possible value assignments for an agent are its feasible local plans. The constraints for an agent come from two sides. The first one is its own resource capacity which it must not overflow. The other one is the actions executed by other agents. If those actions lead the agent to failure, the agent is bound to spend extra resources to ensure success. The goal is that each agent can find an effective local plan for which its local resource constraints are satisfied.

There is already a large amount of work on CSP such as the asynchronous backtracking algorithm (Yokoo, Durfee, et al. 1992). What distinguishes our research from others is that our approach (Section 5) does not generate new constraints, i.e. nogoods, from the agents' communications. On the contrary, our agent always starts with all possible constraints. It starts with an over-constrained local plan, which factors in all possible constraints from other agents (their possible actions), with a very high utilization. The fact that our agents are trying to come up with plans rather than simple value assignments allows them take into consideration all possible constraints in the beginning. It is not always feasible for other traditional CSPs, e.g. 8-queen problem, to do so. The agent then communicates with the other agents to find out what demands, i.e. their actions, that lead to constraints are in fact void. It strips off these unnecessary constraints until it finds a plan that does not overflow its resource capacity.

The agents perform this demand-driven information exchange to limit their searches for satisfying plans. Interesting enough, our work exhibits a duality with the work of (Conry, et al. 1990). Conry discusses how agents, among which information is distributed, may broaden their searches by requesting additional information from others when the agents are not making any progress. We are instead suggesting that additional information can be used to limit the scope of an agent's search by discovering which part of the search space it does *not* need to consider. The heuristics in this paper and in Conry's serve the same function to speed up the searches by trying to identify the "right" information to ask for.

3. Background

We situate this research in a real-time environment where (1) external events are dynamic[1]; (2) it takes time for agents to gather and process sensory data to recognize the states they are in; (3) it takes time for agents to execute the corresponding (re)actions; (4) and more importantly, there are deadlines associated with actions that the agents cannot miss. When a real-time agent misses a deadline, it can mean catastrophe. The entire plan/mission is considered a failure. The Cooperative Intelligent Real-time Control Architecture (CIRCA) has been developed to model the interactions between actions and external events explicitly, taking into account the real-time restrictions of execution (Musliner 1995). CIRCA selects, schedules and executes recognition-reactions assuming a resource-limited platform. Here, we concentrate only on the relevant features. More complete treatments of the overall CIRCA architecture are presented elsewhere (Atkins 1999).

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) computed by the AIS. Inside the AIS are the probabilistic planner and the scheduler. The AIS constructs real-time control plans that are sets of recognition-reactions, called TAPs,[2] generated by the planner. The reactions are scheduled by the scheduler such that they will be executed before the deadlines whenever emerging system failures are detected. The RTS checks periodically whether the reactions should be executed by examining the corresponding recognition tests. CIRCA's plan is therefore a cyclic (periodic) real-time control plan of scheduled recognition-reaction pairs.

Since the scheduler is working with a RTS with limited resources, it could be the case that not all of the requested TAPs can be scheduled. When this occurs, CIRCA calculates the probabilities of the agent reaching different states, called state probabilities. It finds a subset of TAPs by removing those states with state probabilities below a threshold. It keeps increasing this threshold until a schedulable subset is found. The idea behind this probability threshold cutoff heuristic is to prioritize the states of the world in any-time manner by their probabilities, so that the agent can devote its resources to respond to events that are more likely to arise over less likely ones (Li, 2001). The failure probability increases whenever this heuristic is used because some TAPs necessary to preempt possible (though less likely) failures are removed. Consequently the utility decreases. CIRCA's utility function is defined using the Cobb-Douglas function, $(1-F)^{\alpha}(G+1)^{(1-\alpha)}$, where F is the failure probability, G the probability of reaching the goals, and $\alpha \in (0,1]$. CIRCA attempts to minimize the probability of

[1] Events can change while agents are deliberating.

[2] An action with a precondition test for state recognition is called a Test-Action-Pair (TAP).

failure and maximize the probability of success (reaching the goals).

The CIRCA state-space representation of the world is constructed from a set of STRIPS-like state features and state transitions (with preconditions and postconditions) included as part of the planner knowledge base. A state in the world model is created dynamically by applying a transition to its parent state with state features matching the preconditions. 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 and natural processes outside the agent's control are modeled as temporal transitions, either "innocuous" temporal transitions (labeled *tt*) or "deleterious" temporal transitions leading to system failures (labeled *ttf*). When there is a *ttf* in a state, CIRCA selects an action to preempt the failure. The action is called a guaranteed action because it is so scheduled that it is guaranteed to be executed before the *ttf*. A typical state diagram for planning is shown in Figure 4-1. It is the state diagram for agent FIGHTER. Action SHOOT-MISSILE-1 is a guaranteed action to preempt the *ttf* BEING-ATTACKED. Another type of action that is also scheduled with real-time deadlines is called a reliable action. Reliable actions, however, do not preempt any failures. Action HEAD-TO-LOC1 is a reliable action that requires scheduling resources.

A CIRCA agent in a multiagent setting builds on the single agent architecture described above. In addition, the agent distinguishes between private/local features and public/shared features of the environment. Private features of an agent are those features that no other agents are interested in but the agent itself, such as current fuel level. They do not show up in the state diagrams of other agents. Public features are those features that more than one agent is concerned about. It is through manipulating the public features that agents impact each other. For example, in Figures 4-1 and 7-1, COMM and ENEMY are public features shared by both BOMBER and FIGHTER, while HEADINGF and LOCF are private features that are accessible only by FIGHTER. [3]

Furthermore, in a multiagent setting, a CIRCA agent includes in its planning the public temporal transitions and public actions of other agents (labeled *ttac*) that affect some public features. Those are the temporal transitions and actions that change the public features other agents care about. Private temporal transitions and actions of another agent do not include in their postconditions any public features but only private features. For example, B:BOMB-1 and B:BOMB-2 are public actions of BOMBER in Figure 4-1, while action HEAD-TO-LOC1 is private for FIGHTER. Temporal transitions FLY-TO-LOC0, FLY-TO-LOC1 and FLY-TO-LOC2 are private for FIGHTER in Figure 4-1. In Figure 7-1, there are transitions of the same names but they are private for BOMBER. There are no public temporal transitions in this example.

The mechanism of private and public features lets an agent model only the relevant features of the world. It does not need to know the entire plans of other agents, but only those affecting 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 others. Because the number of states is exponential in the number of features, this abstraction significantly reduces planning complexity and state diagram sizes.

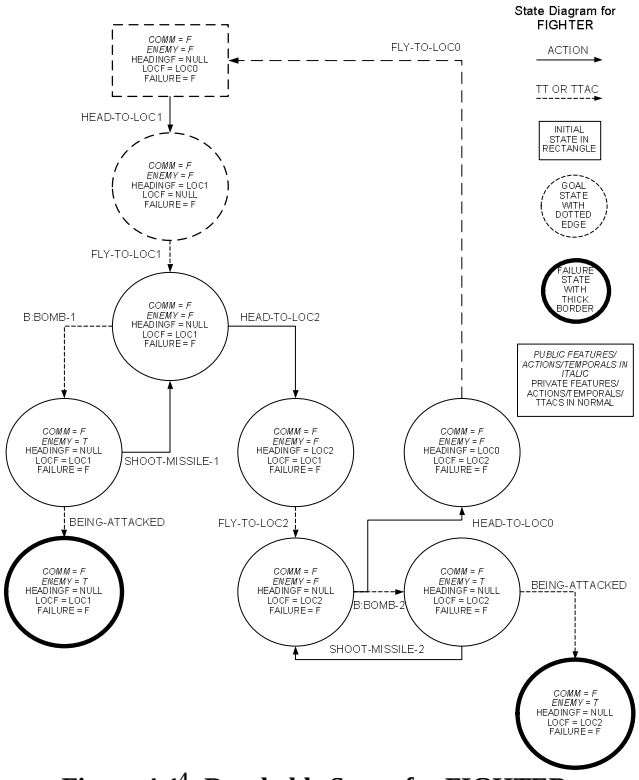

Figure 4-1[4]: Reachable States for FIGHTER

4. Reachability Analysis

In some cases, an agent in a multiagent setting may fail because of uncertainty about the actions of other agents. The other agents cannot be expected to know what constitutes failure for the agent (which could in part be based on private features), nor can they promise not to affect the public features that are relevant to it. The agent therefore has to consider all states that might arise due to the transitions of the environment as well as due to every possible action executed by another agent. The agent has to be prepared for all of the states it *may* reach as a result of these transitions and actions. Regardless of whether a state is reachable from a sequence of environment

[3] FAILURE is always a private feature.

[4] All actions are either guaranteed (e.g. SHOOT-MISSILE-1) or reliable actions (e.g. HEAD-TO-LOC1). This holds true also for Figure 7-1.

transitions (*tts*) or a sequence of others' actions (*ttacs*) or a combination of both, the agent has to expend resources to schedule an action for the state to preempt any imminent failure. We call such an analysis a "reachability analysis". Figure 4-1 shows the state diagram for a fighter hovering to defend two locations for any potential bombing activities by BOMBER based on a reachability analysis. FIGHTER includes in its planning, hence the state diagram, all public actions by BOMBER.

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 simply disregard (be proudly ignorant about) the plans of other agents. Regardless of which of the known possible actions the other agents choose to do, and thus which potential states it may thus encounter, it has the resources to always execute the proper reactions fast enough to preempt failures. Unfortunately, it is possible that the agent cannot schedule all the needed actions for all states it thinks it may encounter based on the reachability analysis due to over-utilization of resources. In terms of CIRCA, the utilization of each planned action (or TAP) is *u = sum of the testing and execution time of the action divided by the period of the action*. When the sum of all utilizations of the actions exceeds 1, no schedule is possible[5]. The agent will not be able to provide real-time guarantees to all TAPs.

5. Convergence to Reduce Resource Requirements

If an agent has sufficient resources to preempt all emergent failures resulting from every temporal transition (*tt*) and temporal-transition-that-is-an-action (labeled *ttac*) of other agents, as based on the reachability analysis described in Section 4, then the agent is done. Because it has more than enough resources relative to the worst-case demand it envisions, it can make all the guarantees it wants to make, even if it ends up devoting some resources to situations that cannot possibly arise. Otherwise, the agent must reduce the number of contingencies it wants to be ready for. It can do so by finding out more precisely which actions other agents are planning to take and, more importantly, which actions they are not planning to take.

We have developed a protocol that allows agents to exchange information about the relevant parts of their plans to reduce resource consumption in cases of insufficient resources. The protocol lets agents identify branches in the state diagrams that cannot be reached during execution and then eliminate the associated actions from their tentative plans. We assume the protocol is run after all agents have locally formed their reachability graphs and planned the actions they would like to take (resources permitting). The protocol is described using C-like code in Protocol 5-1.

```
Inquiring agent () {
    Choose the next branching point that may reduce the
    resource utilization most;  // *
    Ask the agent corresponding to the branching point
    which actions/branches it selects;
    Upon receiving an answer, update its state diagram
    and local plan;
    Loop back to * until either the resource constraints
    are satisfied or all states are examined;
}

Answering agent () {
    When (being asked by another agent about a
    branching point) {
        Identify the corresponding state(s);
        Reply to the agent with the action(s) (or none)
        planned for the state(s);
        Associate the agent with the state(s);
    }
    // **
    If (an action/branch is pruned from its state diagram
    and local plan) {
        Inform all agents associated with the state that
        the state is no longer planned for;
    }
}
```

Protocol 5-1

A branching point is a combination of a state and the set of mutually exclusive *ttacs* in that state corresponding to alternative choices of actions by an agent. Each *ttac* corresponds to a branch in the state diagram that takes up certain resources. In terms of CIRCA, the resource is the utilization of the schedule. Protocol 5-1 inquires about these states in descending order of how much the estimated resource consumption is expected to be reduced if the unplanned actions (*ttacs*) for an agent in the state are pruned.

If all features are public, i.e. there are no private features, then the answering agent can uniquely identify the state (if exists) that the inquiring agent is asking about. It will reply with the only action planned for that state. All but one *ttacs* in the state diagram of the inquiring agent are pruned. If there are public features, the inquiring agent sends a state description instead of all state features. A state description consists of public features and their values. The answering agent may have more than one state in its local state diagram that fits the state description. It must reply with all the actions planned for these states. In other words, the inquiring agent must prepare for more than one *ttac* in that state.

It is important to note that an agent is asking what action another agent *would* execute if a state is reached. The answer can be found simply by looking up the TAPs

[5] A total utilization of less than 1 is only a necessary condition. The set of TAPs can still be unschedulable even if the utilization is less than 1.

in its tentative local plan based on the initial reachability analysis as described in Section 4. If the agent is asking whether another agent *will* execute an action, the other agent may not be able to answer at all because that will depend on its certainty about whether the state will be reached during run-time.

Before running Protocol 5-1, each agent constructs its state diagram based solely on its own plan and its knowledge about how the world may evolve and what other agents may do. Each agent could have a quite different reachability graph from those of the others. In fact, agents typically do not even have the same set of state features. Therefore, some agents may think some states are reachable while others may think otherwise. For the example in Figure 4-1, FIGHTER thinks that enemies will appear at LOCF == LOC2, while BOMBER may not concur. In cases of insufficient resources to handle all states in their respective state spaces, Protocol 5-1 allows agents to gradually discover together which states are indeed reachable by exchanging only the relevant parts of their plans. They are able to refine their individual plans by converging toward a set of commonly agreed-upon reachable states (with public features) until their resource constraints are satisfied. Note that they do not have to agree completely on the set of reachable states, but only need to agree enough such that they can schedule all their required actions after pruning the unnecessary actions from their tentative plans. If an agent fails to schedule for all TAPs after performing the convergence protocol, it can resort to the state probability threshold cutoff heuristic as in the single agent case.

Clearly, Protocol 5-1 terminates.[6] In the worst case, it terminates after an agent examines all states in its initial reachability state diagram. As the protocol examines the states exhaustively, an agent will move toward satisfying its resource constraints by pruning away the unnecessary trajectories. If it starts with sufficient resources to prepare for all that are necessary, then it is guaranteed to find a plan that schedules all the needed actions. The agent's utility is not compromised if it can schedule for all TAPs after running the protocol because the protocol always prunes away TAPs that are assured to be useless. Its utility decreases only when it has to drop some (necessary) TAPs by raising the probability threshold after examining all states in its local state diagram. Therefore, the order of inquiry about the choices made by other agents 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 updated information about that state (by **) before it has to raise the probability threshold.

6. Choice Functions

The part of the algorithm marked (*) requires a heuristic to choose what the next best branching point is. We call such a heuristic a choice function. The more effective the choice function is, the sooner the protocol leads an agent into finding a satisfying plan, and the less computation and communication the agent needs to do. In general, the states that are closer to the initial states and that 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 relying on them. Therefore, they tend to free up more resources if they are pruned.

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 heuristics also 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 descending order of their distances to the initial states. The distance of a state is the minimum 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. However, not all actions have the same utilization.
5. Utilization Choice Function: The agent inquires about the states in descending order of utilization per branch.

A summary of their converging speeds for our sample domains is shown in Section 8.

7. Demonstration

We will now continue our story in Figure 4-1 and demonstrate that our approach is helpful when agents do not have enough resources to schedule for all recognition-reactions. A bomber (BOMBER) is given a mission to destroy one of the two locations, whichever one it chooses. Whenever it bombs a location, the enemies will be alert and endanger the bomber. A fighter (FIGHTER) is assigned to hover around the locations to defend the bomber (Figure 4-1) whenever enemies show up. Also, if FIGHTER requests a response from the bomber, BOMBER has to report its status at LOC2. A complete state diagram for BOMBER, when planning individually is shown in Figure 7-1.

[6] This assumes a finite number of states for the domain.

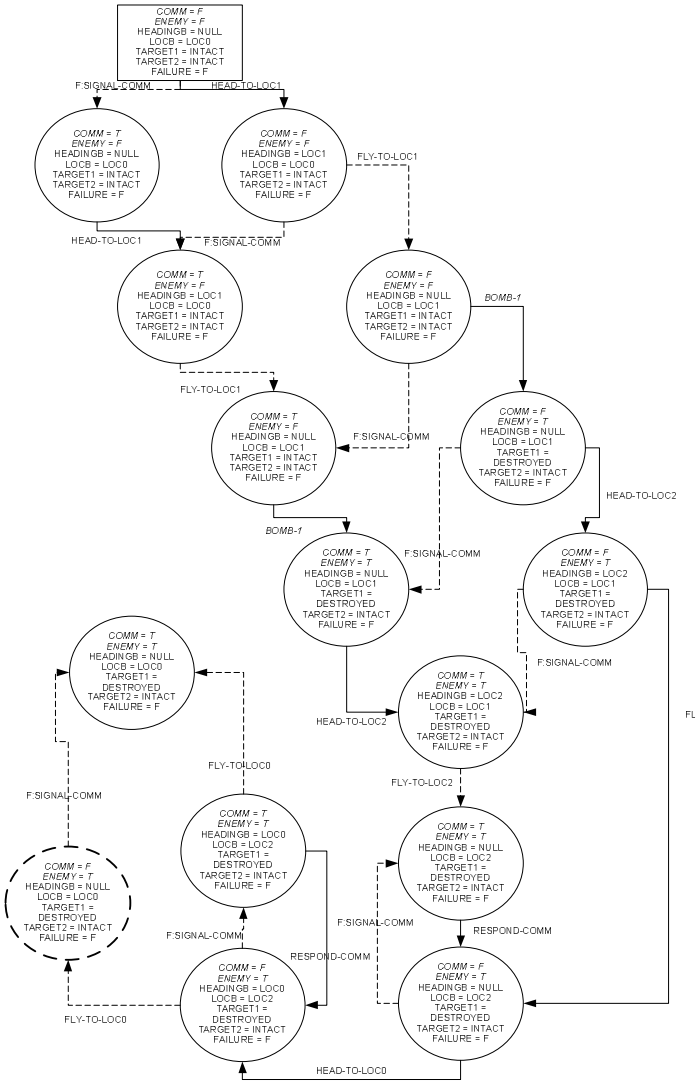

Figure 7-1: Reachable States for BOMBER

Both FIGHTER and BOMBER have 5 actions to schedule if they do not know the plan of the other agent. Suppose the resource constraints are such that each agent can schedule only 4 actions. Both agents exceed their capacities. By running Protocol 5-1 with the sequential choice function, FIGHTER asks BOMBER what actions it plans when ((COMM == F) && (ENEMY == F)).[7] BOMBER replies that it is going to do only BOMB-1.[8] Then FIGHTER can safely remove the children of the *ttac* B1:BOMB-2, hence action SHOOT-MISSILE-2, from its state diagram and tentative plan. Likewise, BOMBER will discover that FIGHTER does not plan to signal BOMBER to respond at LOC2. So, BOMBER can safely remove RESPOND-COMM from its plan. As a result, after communicating using the protocol, both of them have only 4 actions or TAPs left for scheduling and can satisfy their resource constraints. No agent needs to resort to using the probability threshold cutoff heuristic. Thus, no local/global utility is compromised.

8. Evaluation and Experiments

To evaluate the effectiveness of the convergence protocol and the efficiencies of the choice functions, we have generated a set of random domains within certain parameters. In total, there are 287 domains with 1626 agents. Each domain has a random number, chosen uniformly, 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 combined. The number of public features in a domain is random and measures how tightly coupled the agents are for that domain, i.e. how many features they share. Needless to say, all knowledge bases for any given domain have the same number of public features because any public feature is shared by all agents. There are also 15 private and public actions combined, and 7 private and public temporal transitions combined for each agent. The actions and temporal transitions are generated such that they flip a random number of features. In other words, a random number out of the 7 binary features will be inverted by those actions and temporal transitions. All knowledge bases for any domain contain the same set of public temporal transitions and public actions.

We measure the effectiveness of the convergence protocol by what we called "Action Effectiveness" and "State Effectiveness". Action effectiveness is the percentage of unnecessary but planned actions discovered by the convergence protocol. Likewise, state effectiveness is the percentage of states, which cannot be reached given agents' plans, included in an agent's state diagram but removed by the convergence protocol. For our experiments, action effectiveness has an average of 51.74% and standard deviation 35.84. State effectiveness has an average of 66.22% and standard deviation 33.79. The data suggest that more than 50% 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 they think they may encounter are in fact not reachable. Communication is therefore very important for resource-tight agents. Our protocol allows the agents to remove all these states from their state diagrams.

Effectiveness in general depends on many factors, such as the number of agents, the degree of coupling, the number of states, the topologies of state diagrams, and the resource requirements of the recognition-reactions. We have run an Ordinary Least Square regression on state effectiveness against the number of agents and the degree of coupling to determine their empirical relations. The coefficient for the number of agents is 0.86 and that for

[7] Agents communicate only the public features and actions.

[8] BOMB-1, BOMB-2, RESPOND-COMM are the only public actions for BOMBER.

Choice Function Efficiency	Random (Action)	Random (State)	Sequential (Action)	Sequential (State)	Distance (Action)	Distance (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	0.76	4.45
standard derivation	0.48	2.78	4.30	23.86	4.40	24.66	1.65	8.22	2.22	12.23

Table 8-1

Communication Efficiency	Random (Action)	Random (State)	Sequential (Action)	Sequential (State)	Distance (Action)	Distance (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	0.11	0.73
standard derivation	0.09	0.48	0.19	1.06	0.22	1.22	0.98	1.58	0.12	0.68

Table 8-2

the degree of coupling is 10.86. The regression suggests that state effectiveness depends strongly on the degree of coupling. When the degree of coupling increases by 1, state effectiveness increases by 10.86. In other words, on average, 10.86% more states in the state diagrams are unreachable before the convergence protocol for our samples.

We have also measured the efficiencies of the different choice functions. Choice Function Efficiency is measured by the number of actions (states) removed per inquiry. Table 8-1 shows the summary. 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 all). They are the states that have more actions (total actions; not actions per branch) and utilization. The ancestor of a node has at least as many actions as the node itself.

Although choice function efficiency tells us how efficient a choice function is at pruning 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 pruned actions. Even if the agent itself has sufficient resources so that it never needs to ask other agents, it may have to answer a lot of inquiries. We would like to measure how efficient it is for an agent to communicate using the convergence protocol as a member of a group. We call the number of actions (states) pruned per message sent "Communication Efficiency". The communication efficiencies for different choice functions are shown in Table 8-2.

Although our experimental random domains suggest that a lot of (~50%) the states or actions are removed after the agents perform the convergence protocol, the reader should be wary of using these statistics to predict the performance of using the protocol in problem domains whose characteristics are very different from our experimental settings.

9. Conclusions

We see from Section 7 that our approach helps agents finalize their tentative plans by reducing the resource consumption iteratively until no resource constraint is violated. Our framework allows agents to collaboratively generate plans that satisfy their individual resource constraints in a distributed manner. Our experiments suggest that it is quite worth the effort for agents to communicate if they do not have sufficient resources, because more than 50% of the resources are wasted when they are ignorant about the plans of others. The larger the number of agents and the higher the degree of coupling among agents, the better justified is the cost to run the convergence protocol. Different choice functions allow us to trade computation time with communication overheads.

Although our problem is along the lines of CSPs, the very fact that we are finding plans rather than simple value assignments allows us to exploit this idea: our agents start with over-constrained plans and then gradually "relax" the constraints from the other agents until they find the solutions.

We do not yet allow agents to change their actions after the reachability analysis phase. Intuitively, if the agents are allowed to change their actions after they know more about the plans of other agents, rather than merely pruning mutually exclusive trajectories, they can search a larger space of potential plans before they have to raise probability thresholds. Our future research will therefore incorporate negotiation techniques to let agents make mutually beneficial agreements to reduce their resource consumption further.

10. Acknowledgements

The authors thank the reviewers' careful and constructive comments. This research was supported in part by DARPA/AFRL Contract F30602-00-C-0017.

11. References

Atkins, E. M., Plan Generation and Hard Real-Time Execution with Application to Safe, Autonomous Flight. Ph.D. Thesis, the University Of Michigan, 1999.

Boutilier. C. Sequential Optimality and Coordination in Multiagent Systems. IJCAI-99, 1999.

Clement. B. J. and Durfee, E. H. Theory for Coordinating Concurrent Hierarchical Planning Agents Using Summary Information. Proceedings of the Sixteenth National Conference on Artificial Intelligence, 495-502, 1999.

Conry, S. E., MacIntosh, D. J., and Meyer, R.A. DARES: A Distributed Automated Reasoning System. Proceedings of AAAI-90, pp. 78-85, 1990.

Durfee, E. H. and Lesser, V. R. 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, September 1991.

Ephrati, E. and Rosenschein, J. S.. Multi-Agent Planning as a Dynamic Search for Social Consensus. The Thirteenth International Joint Conference on Artificial Intelligence, Chambery, France, August 1993, pages 423-429.

Ephrati, E., Pollack, M. E. and Rosenchein, J. S. A Tractable Heuristic that Maximizes Global Utility through Local Plan Combination. The First International Conference on Multi-Agent Systems, 1995.

Georgeff. M. Communication and Interaction in multi-agent planning. Proceedings of the Third National Conference on Artificial Intelligence (AAAI-83), pp. 123 – 129, 1983.

Krishna, C. M. and Shin, K. G. Real-time Systems. McGraw-Hill. 1997.

Li, H, Atkins, E., Durfee, E. H. and Shin, K. G. Practical State Probability Approximation for a Resource-Limited Real-Time Agent. Proceedings of the IJCAI-01 Workshop on Planning with Resources, August 2001.

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.

Shintani, T, Ito, T., and Sycara, K. Multiple Negotiations among Agents for a Distributed Meeting Scheduler. In Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS'2000), 2000.

Xuan, P., Lesser, V., and Zilberstein, S. Communication in Multi-agent Markov Decision Processes. In Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS-00), Poster Session, 2000.

Yokoo, M., Durfee, E. H., Ishida, T., and Kuwabara, K. Distributed constraint satisfaction for formalizing distributed problem solving. Proceedings of the 12th IEEE International Conference on Distributed Computing Systems, pp. 614-621, 1992.

Zhang, Podorozhny, Rodion, and Lesser. Cooperative, MultiStep Negotiation Over a Multi-Dimensional Utility Function. In Proceeding of the IASTED International Conference, Artificial Intelligence and Soft Computing (ASC 2000), 136-142, Banff, Canada, July, 2000, IASTED/ACTA Press.