Controlling Communication in Distributed Planning Using Irrelevance Reasoning

Michael Wolverton **Marie desJardins**
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
{mjw|marie}@erg.sri.com

Abstract

Efficient and effective distributed planning requires careful control over how much information the planning agents broadcast to one another. Sending too little information could result in incorrect plans, while sending too much information could overtax the distributed planning system's resources (bandwidth and computational power). Ideally, distributed planning systems would have an efficient technique for filtering a large amount of irrelevant information from the message stream while retaining all the relevant messages. This paper describes an approach to controlling information distribution among planning agents using *irrelevance reasoning* (Levy & Sagiv 1993). In this approach, each planning agent maintains a data structure encoding the planning effects that could potentially be relevant to each of the other agents, and uses this structure to decide which of the planning effects that it generates will be sent to other agents. We describe an implementation of this approach within a distributed version of the SIPE-2 planner. Our experiments with this implementation show two important benefits of the approach: first, a noticeable speedup of the distributed planners; second—and, we argue, more importantly—a substantial reduction in message traffic.

The Problem

Real-world plans in modern organizations are often developed by committee, with multiple planners who are each responsible for a particular part of the plan or a specific activity within the planning process, and who cooperate to combine their results into a final integrated plan. There is a growing need for technology that supports this decentralized planning activity by providing the functionality of AI planning systems in a distributed environment. In this model, multiple planning agents are distributed across different processes and locations, perhaps each supporting a user in a mixed-initiative planning session. They cooperate to develop a single integrated plan for a joint planning problem. Each agent is assigned a collection of subgoals from the high-level goal to work on. To plan effectively, the agents must communicate with one another during the planning process. This means that a given agent must send messages to some or all of the other agents about the current state of its subplan development: what conditions are being made true or false in its current partial plan, what resources are being used, what constraints have been generated, and so forth. This communication is necessary so that the planning agents can detect and avoid potential conflicts and potential duplications of work early in the process, instead of having to replan major portions of the joint plan if the conflicts are discovered at the end of the process when the completed subplans are merged.

This need for communication introduces a new problem: what exactly should the agents send to one another? More specifically, how does a planning agent, A, determine whether a given planning decision it has made is important to the problem solving of another agent, B? It cannot answer this last question precisely without actually solving B's subplan, so it must rely on a heuristic answer to a slightly different question: is the planning decision *likely* to be important to B's planning? The consequences for having either an overly permissive or an overly strict answer to that question are serious. If the heuristic allows too many messages to be sent—for example, by broadcasting every planning decision made by every agent to every other agent—it can quickly overburden the resources of the distributed planning system in several ways:

- By overloading the communication links between planners, especially in situations where bandwidth is limited
- By bogging down the planning algorithms with numerous irrelevant predicates and constraints to add to the agents' partial plans
- By overwhelming the human operators at each of the planning nodes with a flood of messages they do not need to see.

On the other hand, a heuristic that sends too little information will suppress messages that reveal conflicts and other interactions between agents' subplans. If even one such message goes unsent, an undetected conflict can lead to extensive replanning when the subplans are

merged, or, depending on the planning algorithm, the inability of the system to find a correct plan at all. One possible approach would be to bypass the heuristic entirely, and instead have a user or set of users select the information that gets passed from agent to agent. This, however, is clearly an impossibly large task in complex, real-world planning domains.

Example Figure 1 shows a simple transportation example demonstrating the problem. In this example, there are two Hierarchical Transition Network (HTN) planning agents cooperating to transfer a payload from TRUCK-A to TRAIN-A. The truck agent is responsible for moving the truck from LOC-A to LOC-C and unloading the payload there. The train agent is responsible for moving the train to LOC-C and loading the payload there. The agents' current partial plans are shown in Figures 1(a) and 1(b). In those displays, rounded rectangles represent primitive actions, and they are shown with all the planning effects they produce. Hexagons represent unexpanded goals. The lighter-shaded goals are the responsibility of the plan's owner, while the darker-shaded goals are the responsibility of the other agent.

At this point in the distributed planning process, the truck agent has completed its part of the joint plan—the only two unexpanded goals in its display are the responsibility of the other agent—and is ready to broadcast the relevant effects of its actions to the train agent. It could broadcast all the effects its actions produced, including all of the effects having to do with the location of TRUCK-A. However, a cursory glance at the planning operators that the train agent will use to solve its goals reveals that the location of TRUCK-A cannot possibly be relevant to the train agent's planning. For example, LOAD-OP, the operator that the train agent will use to solve its (LOADED ...) goal, is shown (Figure 1(c)). TRUCK-A will not unify with the variable PAYLOAD1, so none of the truck agent's (AT-LOC TRUCK-A ...) effects can possibly threaten or satisfy LOAD-OP's first precondition. Also, if LOAD-OP is to be used to solve the train agent's (LOADED ...) goal, the variable VEHICLE1 will be instantiated to TRAIN-A; since TRUCK-A will not unify with TRAIN-A, the truck-agent's (AT-LOC TRUCK-A ...) effects also will not be relevant to LOAD-OP's second precondition. Those effects are similarly irrelevant to the train agent's other goal.

The only effect that could be relevant is in the last action—(AT-LOC PAYLOAD-A LOC-C)—which unifies with LOAD-OP's first precondition. The truck agent must inform the train agent of that effect only. It can avoid sending all the other effects, if it can get a description of the predicates that might be relevant to the train agent's problem solving (via the train agent or by its own analysis of the train agent's goals and planning operators).

This paper describes an approach to reducing message traffic among distributed HTN planning agents by using *irrelevance reasoning* (Levy & Sagiv 1993). In this approach, agents identify relevant planning effects by constructing a set of *query trees* based on an analysis of their planning operators and assigned goals. These query trees are then used to generate a list of predicates that could possibly be relevant to each agent's planning, and each other agent uses that list to select which of its own planning effects it will send to the agent in question. We have implemented this approach within a distributed version of the SIPE-2[1] planner (Wilkins 1988), called Distributed SIPE. Our experiments with Distributed SIPE show two important benefits of the approach: first, a noticeable speedup of the distributed planners; second—and, we argue, more importantly—a substantial reduction in message traffic.

In the remainder of this paper we first introduce irrelevance reasoning and then describe our application of it to distributed planning. Next, we discuss our implementation of the approach and present experimental results. Last, we discuss some of the issues raised by our project and some of the open research problems in this area.

The Approach

Irrelevance Reasoning

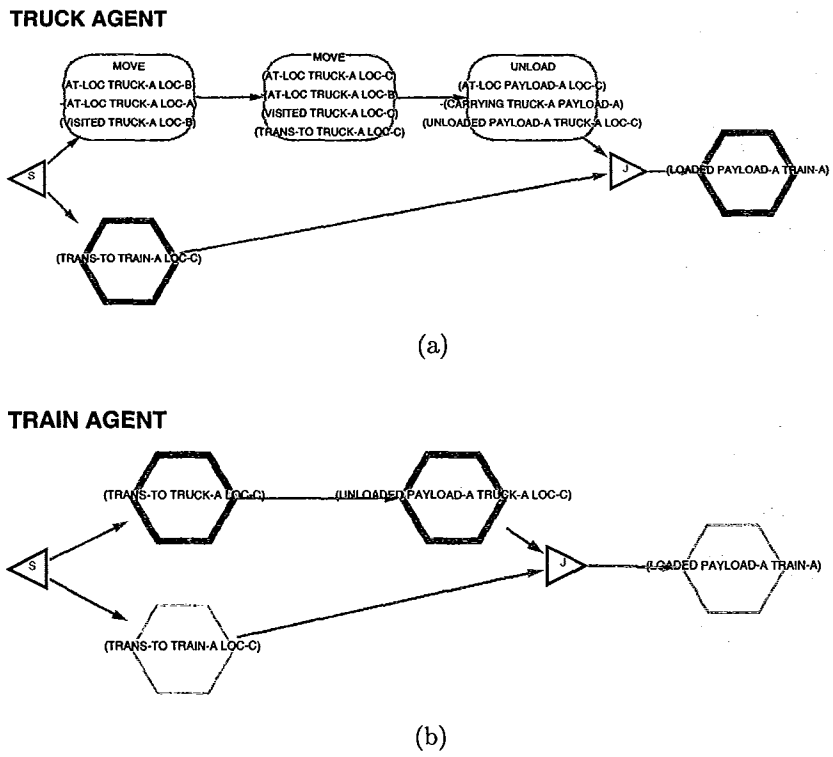
This section presents a very brief introduction to Levy and Sagiv's approach to irrelevance reasoning. For a more thorough explanation, see (Levy & Sagiv 1993) or (Levy 1993).

Irrelevance reasoning is a term describing a class of techniques for identifying those parts of a knowledge base (KB) that are irrelevant to a given query. Levy and Sagiv identify several distinctions that can be made between different definitions of "irrelevant." In this paper we are concerned with what they term *strong irrelevance*: a closed formula ϕ in a KB is strongly irrelevant to a query ψ if ϕ does not appear in any possible derivation of ψ.[2]

To determine which formulas are irrelevant to a given query, Levy and Sagiv construct a *query tree*, an AND-OR tree consisting of goal nodes and rule nodes. The root of the tree is a goal node labeled with the query. Each goal node labeled g is an OR node, and has as its children rule nodes for all the rules whose consequent unifies with g. Each rule node labeled r is an AND node, and has as its children goal nodes for each conjunct in the precedent of r. There are special procedures for handling recursive rules. Query trees provide an efficient means for deriving various types of irrelevance claims about the KB. The result of most concern

[1] System for Interactive Planning and Execution.

[2] This is actually a specialization of Levy and Sagiv's more general definition of strong irrelevance.




```
OPERATOR: LOAD-OP
  Purpose:
    (LOADED PAYLOAD1 VEHICLE1)
  Precondition:
    (AT-LOC PAYLOAD1 LOCATION1)
    (AT-LOC VEHICLE1 LOCATION1)
PLOT:
  Process
    Action:  LOAD
    Arguments:  PAYLOAD1, VEHICLE1,
                LOCATION1
    Effects:
      (CARRYING VEHICLE1 PAYLOAD1)
END PLOT
END OPERATOR
```

(c)

Figure 1: Time slice of a distributed planning session: (a) TRUCK AGENT's completed partial plan; (b) TRAIN AGENT's incomplete partial plan; and (c) planning operator for solving TRAIN AGENT's (LOADED ...) goal.

to us is the following: a ground fact $p(a_1, \ldots, a_n)$ is strongly irrelevant to ψ if and only if $a_1, \ldots, a_n$ does not satisfy the label of any goal node of p in the query tree for ψ.

By removing from the KB formulas that are strongly irrelevant to a query before searching the KB for an answer to the query, irrelevance reasoning can substantially reduce problem solving time. Levy and Sagiv tested their query tree technique on various databases and queries. They found that removing strongly irrelevant facts from the KB produced speedups in problem solving of between 23% and 97%. In most test cases, the time taken to construct the query tree and use it to filter irrelevant facts was insignificant compared to the time needed to find an answer to the query.

Selective Communication

Irrelevance reasoning speeds up problem solving by shrinking the knowledge base—that is, by preventing the problem solver from seeing facts and rules that could not possibly be useful in its problem solving. This technique can also be used to limit messages in distributed planning, provided we can construct a query tree from a planning knowledge base. Fortunately, while the problem solved by HTN planners is not equivalent to that solved by deductive database search engines, HTN operators do provide the same information that is used to construct query trees—knowledge about which rules (operators) can be used to solve a given goal, knowledge about the subgoals that must be solved when a rule (operator) is applied, and knowledge about how variables are constrained when a rule (operator) is unified with a goal.

An HTN operator typically contains four components:

- a *purpose*—a predicate representing the higher-level goal that the operator solves.
- *preconditions*—a collection of predicates representing conditions that must be true in the world state before the operator can be applied.
- a *plot*—a plan fragment that is inserted into the plan in place of the higher-level goal matched by the purpose. The plot consists of a partial ordering of goals and primitive actions.
- *constraints* on the variables that are used as arguments of the other three components. Our present implementation propagates only class constraints, but in principle the approach supports other types of constraints as well.

The construction of a query tree using these kinds of planning operators is similar to the method described in the previous section. The root of the tree is a goal node labeled with an agent's planning goal. Each goal node labeled g is an OR node, and has as its children rule

nodes for all the operators whose purpose unifies with g. The contents of the rule node are the predicates that result in unifying the operator's purpose and its preconditions with g. The label of the rule node is the result of unifying the operator with g. Each rule node labeled Op is an AND node, and has as its children goal nodes for each subgoal contained in the plot of Op.

The use of the query tree for controlling communication in distributed planning works in three steps:

(1) When a planning agent is assigned new goals to solve, it constructs a query tree for *each* goal.

(2) The agent generates a list of relevant predicates for itself by walking the query trees and collecting all the contents of rule nodes.

(3) The agent sends the collected list of relevant predicates to each of the other planning agents.

During the course of distributed planning, whenever an agent, A, is considering sending a planning effect to another agent, B, A checks the effect against B's list of relevant predicates. If a predicate on the list unifies with the effect, it is sent to B. Otherwise, it is not.

A variant on steps (1) and (2) above is to have each agent compute the query trees for each other agent, instead of computing the tree only for itself and broadcasting the results to all other planners. This variant is possible when all planning agents share the same planning knowledge base and when each agent knows the other agents' goals, as is the case in Distributed SIPE. It is desirable when communication bandwidth is at a premium, since it requires no extra communication between agents.

The query tree for the (LOADED ...) goal of the example in Figure 1 is a simple one, with the goal node root and a single child rule node. The rule node represents the operator LOAD-OP unified with the goal, which is already fully instantiated. As a result, only three ground predicates are relevant for that goal:

```
(LOADED PAYLOAD-A TRAIN-A)
(AT-LOC PAYLOAD-A LOC-C)
(AT-LOC TRAIN-A LOC-C)
```

Experimental Results

We have implemented the irrelevance reasoning communication filtering method described above in Distributed SIPE (desJardins & Wolverton 1998), a distributed planning system built on the SIPE-2 planner. In Distributed SIPE, multiple planning agents, running as different processes and usually on different platforms, cooperate to produce a joint plan. Each agent is a running instance of SIPE-2, with the following additional capabilities:

- An agent can assign one or more of its subgoals to another agent.
- An agent can send some or all of the planning effects contained in its current partial plan (i.e., effects created by the actions and goals in its partial plan) to another agent. It can send effects manually, in which case a user selects the effects to be sent, or automatically, in which case the irrelevance reasoning procedure selects the relevant effects from the complete set of effects in the current plan. Along with an effect, the sending agent sends information that allows the receiving agent to know where to place the effect in its plan.
- An agent can submit its completed subplan to a coordinating agent.
- A coordinating agent can merge the subplans that have been submitted to it.

Domain/Problem			*Unfiltered*	*Filtered*
Name	# ops	# acts	# Effects	# Effects
JMCAP	30	41	45	16
Trans	5	24	37	1

Table 1: Message Traffic Reduction

Our experiments were designed to measure the utility of irrelevance reasoning in distributed planning along two dimensions: how effectively the approach reduces message traffic, and to what extent the reduced message traffic speeds up the planners. We measured these effects in two domains: (1) a real-world maritime planning domain, JMCAP,[3] in which U.S. Navy and Marine Corps planners cooperate to produce a plan for a Noncombatant Evacuation Operation (NEO); and (2) a simple transportation domain from which the example in Figure 1 is drawn.[4] Table 1 gives some details about the two domains and problems used in the experiments, specifically the number of planning operators in the domain ("# ops"), and the number of actions in the bottom level of the final joint plan ("# acts"). In each domain, we partition the top-level subgoals between two agents, and then allow one agent to plan its assigned subgoals to completion. That agent then sends some or all of its planning effects to the other agent, which in turn completes its subplan, resolving any conflicts and capitalizing on any opportunities presented by the effects of first agent's plan. We took measurements in two different modes: using irrelevance reasoning to filter irrelevant effects, and using no filtering (i.e., sending all the agent's effects). We measured the number of messages that were sent in these two modes, along with the CPU time used by query tree construction, filtering effects, sending the effects across a network, and the second agent's planning of its subproblem.

Table 1 shows the reduction in message traffic from applying irrelevance reasoning. Both domains showed substantial reductions in the number of effects broadcast. In the JMCAP domain, where there is significant overlap and interaction between the goals assigned

[3] Joint Maritime Crisis Action Planning.

[4] For the experiments, we solved a larger problem in this domain than the one in Figure 1.

to the two agents, the filtered message passing sent 16 of the total 45 effects. In the transportation domain, which was specifically designed to have very little overlap between the responsibilities of the two agents, irrelevance reasoning reduced the message traffic from 37 effects down to only one.

Table 2 shows the results of the planning time experiments. The "Sending" columns represent the time it takes for an agent to walk through the plan and collect possible effects, filter out the irrelevant ones (if in filtering mode), and send the messages to the other agents.[5] The "Planning" columns represent the time it takes for the second agent to complete its planning after it has received all the effects sent by the first agent. The "QT Gen." column represents the time it takes to generate a query tree in filtered mode. Each "Total" column represents the sum of the previous numbers; it adds the times of all activities in a distributed planning session that are part of or are effected by message filtering. Each number represents an average over 20 runs of the system. The results show mild speedup in the maritime domain, and substantial speedup in the transportation domain. In JMCAP, sending and planning combined were 5% faster in the filtered mode, even when query tree construction time was added. In the transportation domain, the speedup was much more significant: combined sending and planning (and QT construction) was 35% faster in filtered mode. The result is better for the transportation domain primarily because the effects sent by the first agent represent a relatively small part of the total planning problem of the second agent in JMCAP—that is, the second agent in JMCAP has a larger problem to solve than the first. The total time difference between the filtered and unfiltered approaches is statistically significant for $t > 0.99$ in both domains.

It is important to note that, for many real-world domains, reducing message traffic can be a much higher priority than reducing planning time at the agent nodes. Our application runs on a cluster of Sun SPARC Ultra workstations linked by high-bandwidth connections, but for many of the domains that drive this work—for example, the maritime planning domain of JMCAP—it is unreasonable to assume such high-bandwidth connections between agents. It is more likely that agents will be communicating over low-bandwidth, wireless channels, and in these situations, the indiscriminate broadcasting of all planning decisions made by all agents is unacceptable. A second factor motivating the importance of controlling message traffic is the need for avoiding information overload of the human operator(s) at each agent. In many real-world domains, the distributed planning agents described here will not be operating autonomously, but will each be serving as a tool for a human planner or planning team. In these situations, even if the automated planning software may be able to handle many irrelevant effects with a tolerable increase in planning time, the human operator can more easily be overloaded by being sent numerous facts irrelevant to his own tasks. Irrelevance reasoning can help avoid overloading human planners as well as computer planning systems.

[5] Observant readers may put Table 1 and Table 2 together and notice that it takes less than twice as much time to send 16 effects in the JMCAP domain as it takes to send one in the transportation domain, and that it takes less than three times as much time to send 37 effects as it takes to send one in the transportation domain. This is because walking through the plan collecting the effects represents a large portion of the time reported in the "Sending" column.

Related Work

COLLAGE (Lansky 1994; Lansky & Getoor 1995) uses a technique called *localization* to decompose a planning problem into subproblems called *regions*. An agenda-based mechanism generates a subplan for each region (corresponding to the distributed agents in Distributed SIPE), and a consistency agenda is used to propagate changes between regions (corresponding to message passing in Distributed SIPE). This approach improves overall planning efficiency by permitting COLLAGE to solve the subproblems in each region (relatively) independently.

Localizations in COLLAGE can be automatically generated based on abstraction levels or scope. COLLAGE uses a set of heuristics to find regions that minimize the number of interactions that can occur between regions. This approach focuses on the problem of assigning parts of the planning problem to regions or agents, in contrast to our work, which uses the notion of irrelevance (similar to COLLAGE's notion of scope) to manage the information flow among the distributed agents. The specification of the regions, and of which constraints need to be propagated between regions, is based on a set of action and constraint types rather than a reachability analysis of the planning operators as described in this paper. However, a modified version of COLLAGE's techniques could be applied to Distributed SIPE to distribute the planning problem in such a way that the interaction among the agents is minimized.

Corkill (1979) describes a distributed version of NOAH, a nonlinear hierarchical planner from which SIPE is conceptually descended. Corkill's distributed NOAH allocates subgoals to distributed planners, and applies distributed versions of NOAH's plan critics to identify interactions among the subplans. Corkill's implementation does interplanner constraint management at a very fine grain, and communicates all constraints that are possibly related to another planner without doing any relevance filtering, so the system entails heavy message traffic between the planners.

DIPART (Pollack 1996) is a distributed, real-time planning system that focuses on issues arising in higher-level control of distributed planning (such as the load balancing of tasks). Communication among agents is primarily at this higher, task-based, level, with some support for incrementally merging subplans. However,

Domain	*Unfiltered*			*Filtered*			
	Sending	*Planning*	*Total*	*QT Gen.*	*Sending*	*Planning*	*Total*
JMCAP	459	15237	15696	994	231	13631	14856
Transportation	406	877	1283	18	175	646	839

Table 2: Effect on Planning Time (in miliseconds)

to our knowledge, DIPART does not include a method for automatically controlling message passing between distributed planning agents.

STEAM (Tambe 1997) is a distributed agent architecture that builds on joint intentions theory (Levesque, Cohen, & Nunes 1990) to enable coordinating agents to maintain a coherent view of the team's goals and plans. STEAM uses *team operators* to represent shared goals and to identify achieved, unachievable, or irrelevant goals; these in turn determine which information is potentially relevant to other agents. This approach is extended with a decision-theoretic framework that incorporates the costs and benefits of communication, as well as the probability that other agents already have the information in question. STEAM's notion of relevance is rather different from Distributed SIPE's—rather than determining whether a piece of information is (ir)relevant to another agent's subgoals, the team operators are used to determine whether a piece of information directly contributes to the success or failure of a joint goal. However, the decision-theoretic approach, which is also used in (Gmytrasiewicz, Durfee, & Wehe 1991), represents an interesting extension that could be layered on top of Distributed SIPE's irrelevance computation to incorporate additional decision-making factors in deciding whether and when to communicate information between planning agents.

Huber and Hadley (1997) describe a multiagent system in which reactive agents coordinate as teammates in playing an internet game called Netrek. They have applied plan-recognition techniques for agents to implicitly coordinate (by understanding each others' activities), and developed a limited communication scheme for the agents to inform each other about their activities. The key to the success of this approach is that it uses an a priori analysis of the domain (performed manually by the designers) to identify what information would be useful to communicate.

Discussion

We have described a technique for using irrelevance reasoning to reduce message traffic in a distributed planning system. The current implementation controls the sending of effects generated by agents' subplans. The technique is general, and could be applied to other types of message traffic in distributed planning systems, or to other types of distributed problem solving system. For example, irrelevance reasoning could be used to determine whether to send requests for other agents to solve subgoals, or to achieve or maintain preconditions. Another application of irrelevance reasoning would be to use the query trees to allocate subgoals among agents, based on some notion of locality, to minimize interactions among subplans (similar to the localization process in COLLAGE).

The irrelevance reasoning approach could also be extended by incorporating notions of probability and utility: how *likely* is it that an agent will in fact need to know a piece of information, and how *useful* will it be to the agent to know it? Also, at some cost in bandwidth, query trees could be dynamically pruned as subplans are generated (e.g., when an alternative planning strategy has been ruled out, the associated preconditions would no longer be relevant).

This paper addresses only the problem of managing information flow by selecting which messages agents should send each other. There are many other significant challenges in developing distributed planning systems, including how to synchronize the activities of the planning agents (who should do what, and when), how to resolve conflicts when they arise, and how to merge multiple subplans. Although we do not discuss these issues in this paper, many of them are being addressed in our ongoing research (desJardins & Wolverton 1998).

In summary, we have developed a novel method for using irrelevance reasoning to control information flow in a distributed planning system. We have implemented the method in Distributed SIPE, and have demonstrated its usefulness in a real domain. Irrelevance reasoning has previously been used successfully in deduction; this work extends the evaluation of this technique into a new application area. In addition, this work represents an investigation of irrelevance reasoning in a fundamentally different context, one in which irrelevant facts have costs beyond simply increasing KB size—for example, consuming valuable bandwidth. Limiting information flow has not been addressed by most previous work in distributed planning and problem solving, and the irrelevance reasoning approach will reduce the message traffic in such systems substantially.

Acknowledgments

This research was funded by the Office of Naval Research and the Space and Naval Warfare Systems Command (SPAWAR) under contract N66001-97-C-8515. The JMCAP domain was constructed based on expertise provided by Mr. Bob Rubin of Program Advancement Group, Inc. The authors would like to thank

Mr. Dave Swanson, JMCAP Project Manager, for valuable guidance on our work. Thanks also to Karen Myers, David Wilkins, and the anonymous AAAI reviewers for insightful comments on an earlier version of this paper.

References

Corkill, D. D. 1979. Hierarchical planning in a distributed environment. In *IJCAI-79.*

desJardins, M., and Wolverton, M. J. 1998. Coordinating planning activity and information flow in a distributed planning system. In desJardins, M., ed., *AAAI Fall Symposium on Distributed Continual Planning.* AAAI Press Technical Report (forthcoming).

Gmytrasiewicz, P. J.; Durfee, E. H.; and Wehe, D. K. 1991. The utility of communication in coordinating intelligent agents. In *AAAI-91*, 166–172.

Huber, M. J., and Hadley, T. 1997. Multiple roles, multiple teams, dynamic environment: Autonomous Netrek agents. In *Autonomous Agents '97.*

Lansky, A. L., and Getoor, L. C. 1995. Scope and abstraction: Two criteria for localized planning. In *IJCAI-95.*

Lansky, A. L. 1994. Located planning with diverse plan construction methods. Technical Report FIA-TR-9405, NASA Ames Research Center.

Levesque, H. J.; Cohen, P. R.; and Nunes, J. 1990. On acting together. In *AAAI-90.*

Levy, A. Y., and Sagiv, Y. 1993. Exploiting irrelevance reasoning to guide problem solving. In *IJCAI-93.*

Levy, A. Y. 1993. *Irrelevance Reasoning in Knowledge Based Systems.* Ph.D. Dissertation, Computer Science Department, Stanford University.

Pollack, M. E. 1996. Planning in dynamic environments: The "DIPART" system. In Tate, A., ed., *Advanced Planning Technology: Technology Achievements of the ARPA/Rome Laboratory Planning Initiative.* Morgan Kaufmann.

Tambe, M. 1997. Agent architectures for flexible, practical teamwork. In *AAAI-97.*

Wilkins, D. E. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann.