Using Observation to Select Complementary Goals

Maria Gordin and Sandip Sen
Department of Mathematical & Computer Sciences
The University of Tulsa
e-mail:[gordin,sandip]@euler.mcs.utulsa.edu

Abstract

We present a goal-recognition based goal ordering strategy for an agent in a communication-free cooperative environment. Agents acting in such environments can improve coordination through observation and inference about the intentions of other agents. The central component of the strategy proposed in this paper is observation based goal recognition, that helps an agent understand what others are planning to do and accordingly select its goal to complement the behavior of the rest of the group. In cooperative as well as competitive environments an agent often needs to pursue goals not being pursued by others. Goal recognition and replanning are usually time consuming, thus they should be used only when it is necessary. Some of the questions we are addressing are: how often and when an agent needs to call the goal recognizer; how far ahead should an agent plan in a dynamic environment and at what level of detail should this plan be generated; when does it needs to replan? We also investigated planning and plan recognition on multiple levels of abstraction.

Introduction

We present a goal-recognition based goal ordering strategy for an agent in a multi-agent environment. Agents acting in such environments need to adapt to dynamic changes in the environment, adjust to other agents and predict actions of other agents based on what they have observed so far. Communication between agents might be an effective aid for coordinating behavior in some domains, but generally, communication is costly, may involve ambiguity and can detract from problem solving (Huber & Durfee 1995). For the latter domains, time and effort spent on communication can be used more effectively in performing problem solving activities. Our research focuses on communication-free cooperative environments to study how far agents can succeed in making successful coordination decisions based purely on observation. We believe that it will help us identify situations when communication is absolutely necessary to further improve performance. This will enable us to build efficient systems with more appropriate usage of available computational resources.

The central component of the strategy proposed in this paper is observation based goal recognition that helps an agent understand what others are planning to do and accordingly select its goal to complement the behavior of the rest of the group. To accurately pin down the exact goal being pursued by another agent may require observing that agent's actions for a significant period of time (Durfee 1995). On the contrary, it is often possible to infer goals not being pursued by an agent by observing only a few actions. In cooperative as well as competitive environments an agent often needs to pursue goals not being pursued by others. Since the latter can be identified quickly, an agent can act instead of waiting until its partner's goal can be identified with a high accuracy. Instead of building an exact model of opponent behavior, we are trying to build a model just rich enough to help adopt complementary behavior. This allows agents to move earlier than later and can save redundancy in efforts.

A model of another agent (partner) can be built by observing its actions using some assumptions about agent strategies (e.g., the agent being modeled is similar to the modeling agent.) To continuously update the model of a partner, an agent may have to call its goal recognition routine after every observation, e.g., at every time step. One of our goals in this research is to find out how to reduce the cost incurred in updating for their models. To make the model updating process economical, we would like to do goal recognition and replanning only when it is necessary. Again, our goal is not to maintain a completely accurate model, but to incrementally update such models only when necessary. The questions we are addressing are:

- how often and when does the observer need to update a model of the observee (by calling its goal rec-

ognizer)?

- how often the observer needs to recalculate its own set of active goals?
- how far ahead should an agent plan in a dynamic environment and at what level of detail should this plan be generated?
- when does the observer needs to replan?

Our Approach

In our approach, an agent observes the actions of a partner agent and tries to narrow down the set of goals that the partner is likely to pursue. This also enables the observing agent to choose its own goals from the set of goals not likely being pursued by the observed agent. Though calling the goal recognizer after every observation enables an agent to build the most up-to-date model of the partner, much computation time can be saved if the agent calls the goal recognizer only when necessary. We will present the conditions to be satisfied before the goal recognizer is called.

Let $\mathcal{G}$ be the total set of goals to be achieved, G be the outstanding set of goals to be achieved ($\mathcal{G} - G$ goals have already been achieved), G_p be the set of goals that the partner is likely to pursue given its set of actions. We propose that the observing agent chooses its set of goals, G_o as follows:

$$\begin{aligned} G_o &= G \text{ if } G = G_p \\ &= G - G_p \text{ otherwise.} \end{aligned}$$

The motivation here is to work on complimentary goals with respect to the partner. If, however, observations cannot eliminate any outstanding goals from the partner's possible goal set, the observing agent adopts the entire outstanding goal set as its set of goals. At any point in time an agent pursues the easiest goal from its set of chosen goals.

To further improve performance, our agents plan on abstract and detailed levels and do action monitoring for the abstract level plan. At the abstract level, the agent decides on the order in which it will achieve the set of goals in G_o. But the agent calculates a detailed level plan for achieving only the first few of these goals. The assumption here is that in a dynamically changing world cohabited by other agents a detailed level plan for goals to be achieved in the not-so-near future will become outdated by the time the agent is ready to pursue those goals. For example, the other agent may have changed its plans and already achieved some of those goals. So, an agent plans at the detailed level only up to a certain number of goals in its abstract level plan.

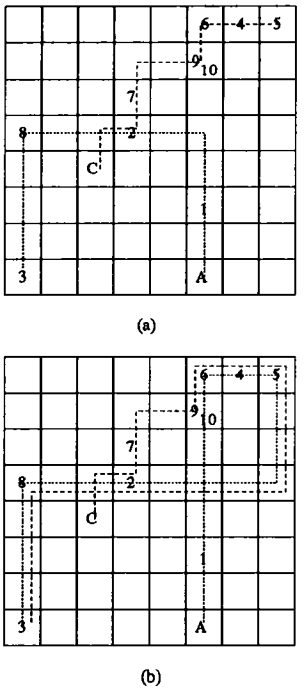


Figure 1: (a) Agent A uses goal recognition, agent C uses greedy strategy. Total number of actions 32. (b) Both agents A and C use greedy strategy. Total number of actions 62.

We have implemented our approach in a room cleaning domain where two robots are cleaning wastebaskets. A set of current goals is then a set of baskets in the room that has to be cleaned. We will compare our approach of agents using goal recognition with those of agents using a greedy strategy of always pursuing the easiest remaining goal (cleaning the next closest waste basket) after achieving its current goal. Figure 1 illustrates traces of agent movements in this domain.

For this domain it is easy to see that selecting complementary goals means picking goals away from the area where the other agent is working. The source for improvement in this domain is the ability of agents to coordinate their efforts by splitting the area into nonoverlaping subareas. For the trace in Figure 1 (b), the agents did not use goal recognition, and hence at some point started pursuing the same goals. In the case on Figure 1 (a) agent A used the goal recognition. It was able to pick the subarea that did not overlap with the subarea on which agent C was working. This effect is the key to the improvement of the agents' performance. We present a more detailed analysis of this effect in the Section .

We also implemented our approach for a more general cleaning domain with multiple rooms. For this domain we used planning and goal recognition at three

levels of abstraction. Our approach can be seen as hierarchical behavior representation as defined by E. Durfee and Montgomery (Durfee & Montgomery 1991) as "a representation that can express different dimensions of behavior at different levels of detail." We believe that our approach can be generalized for domains with more levels of abstraction. As described in Chapter , the use of multiple levels of abstraction can decrease the total number of times goal recognition needs to be performed. This saves a significant amount of computation time. Also, intuitively, goal recognition at higher levels of abstraction is less expensive than goal recognition on lower levels of abstraction, at least because there are fewer alternative goals at the higher level of abstraction that at the lower level.

Goal Recognition

The observer builds a model of its partner in which it records every observed action of its partner. When the goal recognizer is called, it uses the actions recorded in the model to calculate a set of goals that the partner might be pursuing (G_p). Together with updating the model with this goal set, the goal recognizer empties the set of actions it used in its calculations. So it never looks at the same actions twice. After the goal recognizer updates the model of the partner, the observer sets its own set of goals (G_o) as stated in the last section. If there was a change in G_o, the observer replans at the abstract level.

Figure 2 shows the process of goal recognition for part of the trace on Figure 1 (a). For the room cleaning domain the goal recognition process can be illustrated geometrically (for more complex domains, we can adapt more sophisticated goal recognizers (Lesh & Etzioni 1995)). We represent the room as a grid; the portion of the room to which the partner is headed can be approximated from its past movements. This portion of the room is called an "active area" for the partner, and all the waste baskets located in this area comprise G_p. The active area of the partner is marked with dashes in Figure 2. The closest goal for agent A (the observer) in Figure 2 (b) is basket 7, but this goal is in G_p. Hence, agent A moves toward basket 8, the nearest goal in $G_o = G - G_p$.

There are two special cases in finding active goals of the observing agent.

1. The active area of the partner includes all the current goals. For the room cleaning domain, it means that all the waste-baskets are inside the active area of the partner. Example of this situation is illustrated on Figure 3. This is the case when $Gp = G$. We set the active area of the observer to the whole grid. It means that for this case, the ob-

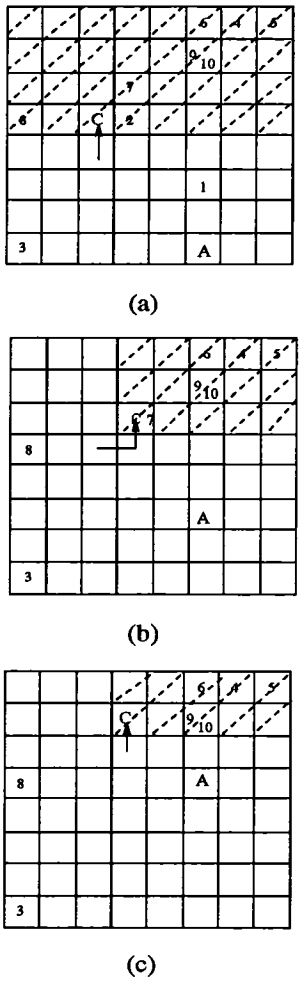


Figure 2: Goal recognition process. Agent A uses goal recognition, agent C uses greedy strategy. (a) Gp=(8,2,7,9,10,6,4,5), Go=(1,3); the next goal of agent A is 1. (b) Gp=(7,9,10,6,4,5), Go=(8,3); agent A's next goal is 8. (If A was not using goal recognition, its next goal would have been the closest goal: 7.) (c) Gp=(9,10,6,4,5), Go=(8,3) Although, there is a change in active areas, the set of active goals remains the same. Agent A does not replan.

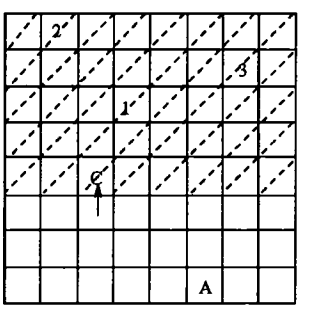


Figure 3: G=(1,2,3), Gp=(1,2,3). All goals fall within the partner's active area.

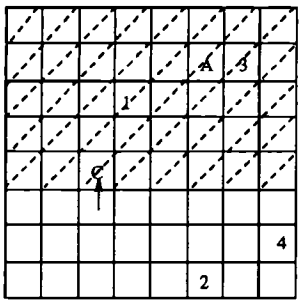


Figure 4: G=(1,2,3,4), Gp=(1,2,3), Go=(1,2,3,4). The observer is inside the active area of its partner. The active area of the observer is set to the whole grid.

server defaults to using the greedy strategy. We believe that this is better than waiting until G_p becomes different from G.

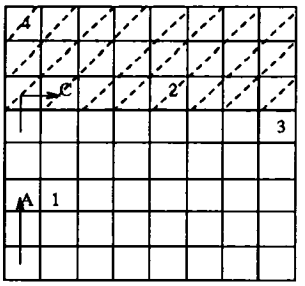
2. The observer is inside the active area of the partner. See Figure 4 for an example. Although there might be goals outside active area of the partner (as in Figure 4), it may not be worse for the observer to move over there when there might be goals right next to it. As in the first exceptional case, we set the active area of the observer to the whole grid. This means that the observer uses a greedy strategy for goal selection until it finds itself outside the active area of the partner.

Goal recognition process is usually time consuming, and often takes exponential time. Computation time spent on goal recognition can be spent on problem solving. Thus, the goal recognizer should be called only when necessary. We suggest calling the goal recognizer when there has been or might be a change in the next goal of the observer or the partner. The time steps in which the goal recognizer is called to update G_p are those in which any of the following conditions is satisfied:

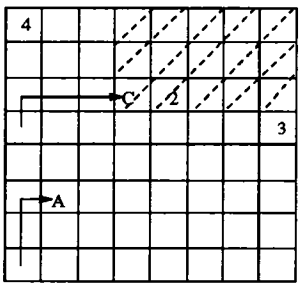
- None of the outstanding goals can be eliminated from the possible set of goals being pursued by the partner, i.e., $G_p = G$ (see Figure 3).
- The observer achieves a goal. Before planning in detail for its next goal, the observer needs to update G_o, and hence G_p (see Figure 5).
- The partner makes its first move after achieving a goal. Such a move may lead to a change in G_p (see Figure 6).

Abstract and Detailed Planning

In our first implementation, agents plan at two levels of abstraction: the abstract level and the detailed level.

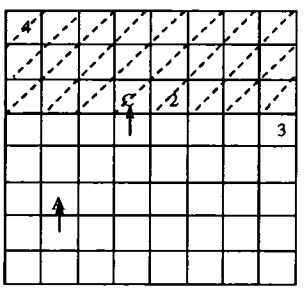


(a)

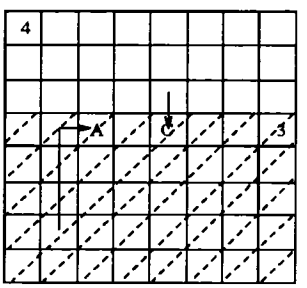


(b)

Figure 5: (a) Before observer (A) achieves goal: Gp=(2,4), Go=(1,3); observer's current goal is 1. (b) After A achieves goal 1: Gp=(2), Go=(4,3); Current observer's goal becomes 4 as it is closer than 3.

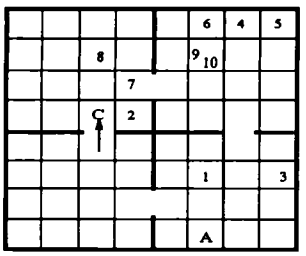


(a)

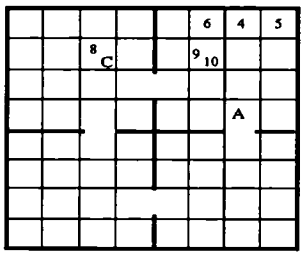


(b)

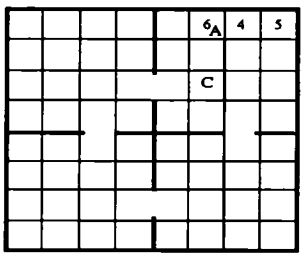
Figure 6: (a) Before the partner (C) achieves its next goal : Gp=(2,4), Go=(3); observer's current goal is 3. (b) Following the first move by C after it achieves goal 2: Gp=(3), Go=(4); observer's goal changes to 4.



(a)



(b)



(c)

Figure 7: Using three levels of abstraction for cleaning domain with multiple rooms.

In the abstract level an agent plans for major goals. An agent generates detailed level plan to execute only the next abstract action in the abstract plan. We believe that in a dynamic environment, the detailed plan is useful only for the immediate step in the abstract plan. Our agents do action monitoring for the abstract level plan after completing every abstract action. In dynamic environments, by the time an agent has finished executing its detailed plan for the current abstract action, the next abstract action may become obsolete. If action monitoring component determines that the next abstract action will fail because of a change in the environment, an agent replans only for its abstract level goals.

In this domain, abstract plan consists of go-to and pick-up actions. For example, robot A plans to empty basket 1 first, basket 2 second, etc. By the time our robot empties basket 1, basket 2 may have already been emptied by another agent. It is not prudent to plan how to get from basket 1 to basket 2 in advance. So, robot A does detailed planning on how to get from its current position to basket 1, and postpones detailed planning for the rest of the abstract actions. After it picks up basket 1, it does action monitoring to determine if the next abstract action, "pick up basket 2" is consistent. If not, it replans at the abstract level, to decide on which set of the remaining baskets it should clean, and in what order.

Two levels of abstraction seem to be natural for the room cleaning domain described in Section . But it is easy to imagine even simple domains with more than two levels of abstraction. Consider, a variant of the room cleaning domain with several rooms as shown in Figure 7. To get from one room to another, an agent needs to use the connecting door way. Two levels of abstraction will still work for this domain, if changes are made to take into account locations of the doors. However, it is more natural to plan on three levels of abstraction for this domain:

1. first level: rooms to clean,
2. second level: waste-baskets inside the room,
3. third level: steps to get from one waste-basket to another and pick it up.

The goal recognizer needs to work on several levels of abstraction as well. When agents are working on different goals of the first level of abstraction, it is guaranteed that their second level goals do not overlap; therefore, the goal recognition needs to be done only at the first level of abstraction. In our domain, when the observing agent is planning for the next room to clean, it needs to avoid rooms that another agent is cleaning or is going to clean. The agent uses the goal recognizer which works with the goals on the first level. The agents work in different rooms until there is only one uncleaned room left. Then, the agents start acting as in our first problem; they try to split the room by calling goal recognizer on the second level of abstraction.

Results

We experimented with three scenarios in the room cleaning domain: both agents using greedy behavior, one agent using goal recognition while the other uses greedy behavior, and both agents using goal recognition. The size of the grid that represent a room was 8x8 in all the experiments. Positions of the waste baskets and initial positions of the agents were generated randomly for each of the experiments. For each above combination of strategies we ran 1000 experiments for each of the following number of waste-baskets in the

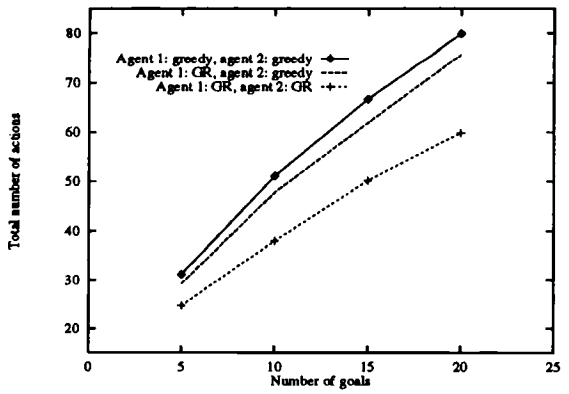


Figure 8: Experiments using different combinations of strategies and different number of goals.

room: 5, 10, 15, and 20. Figure 8 shows the results averaged over 1000 runs. The graphs show number of total actions taken by both agents to complete different number of goals. The difference in performance between the graphs are significant as measured by the Wilcoxon matched-pair signed-rank test. Two greedy agents take the longest time to clean up all the waste baskets. The performance improves in approximately one third of the cases if one of the agents use goal recognition, and the performance drastically improves in the majority of cases if both of the agents use goal recognition.

Our examination of solution traces shows that the performance improves due to the ability of agents to coordinate their efforts by splitting the area into subareas, so that their working areas do not overlap. Even if both agents use the greedy strategy, they may follow nonoverlaping paths in some limited cases. But if anywhere during execution agents get close to each other, they pursue same goals thereafter. This results in redundant, wasted efforts. The goal-recognition based strategy prevents agents from pursuing the same goals. In the solution on Figure 9 (1a) agents using the goal-recognition based strategy successfully split the room into two subareas, and pursue different goals at each point of time. For the same room setting on Figure 9 (1b) , the agents using the greedy strategy start pursuing the same goal early in the trace, thus wasting a lot of time and effort. Agent A cleans up a waste-basket right before agent C gets near it. A similar situation is illustrated on Figure 9 (2). The agents were initially placed far from each other; thus it took some time before they got close to each other. They started pursuing the same goals later in the solution. Figure 9 (3) illustrates the rare case where both strategies are equally efficient. The greedy strategy coincidentally splits the room into two subareas. This particular phe-

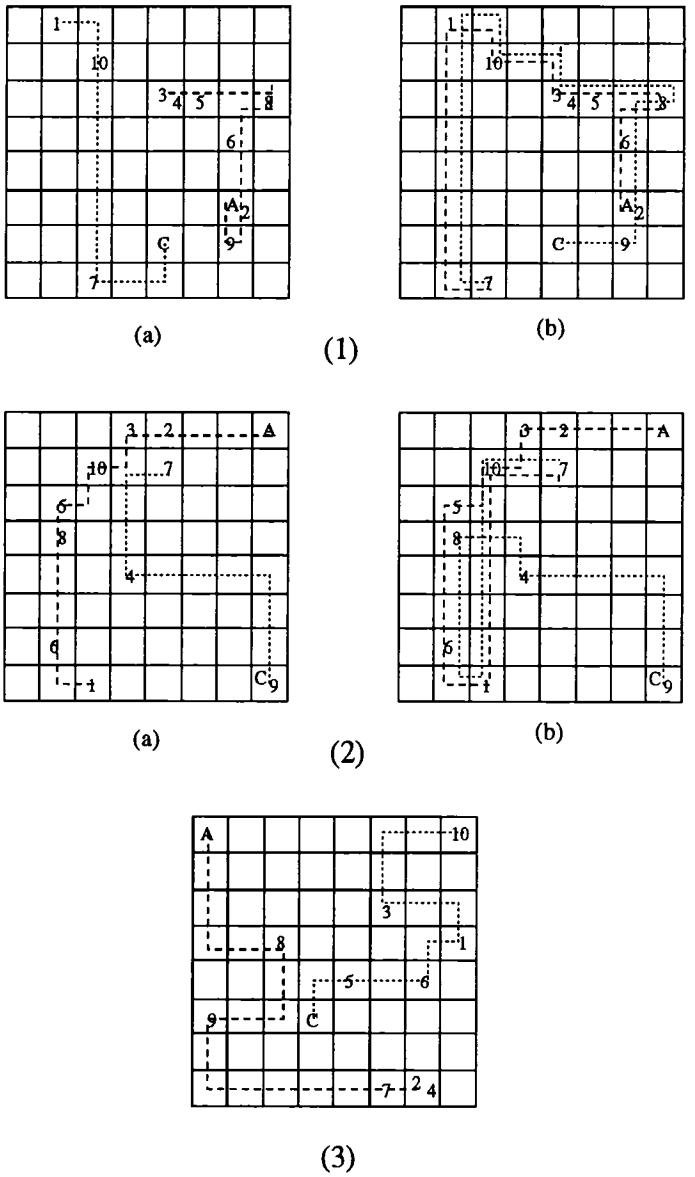


Figure 9: Comparison of paths followed by the agents using goal-recognition based strategy with the paths followed by the agents using greedy strategy. A and C are starting positions of two agents. (1) (a) The agents using goal-recognition based strategy were able to split the room into two subareas. (b) The agents using greedy strategy follow each other most of the route. (2) (a) The agents using goal-recognition based strategy were able to avoid each other on their paths. (b) Greedy agents followed each other once they have met somewhere in the middle of their roots. (3) Greedy strategy worked as well as goal-recognition strategy. In this rare case greedy strategy was able to split the room into subareas. Paths used by greedy and goal recognizing agents are identical.

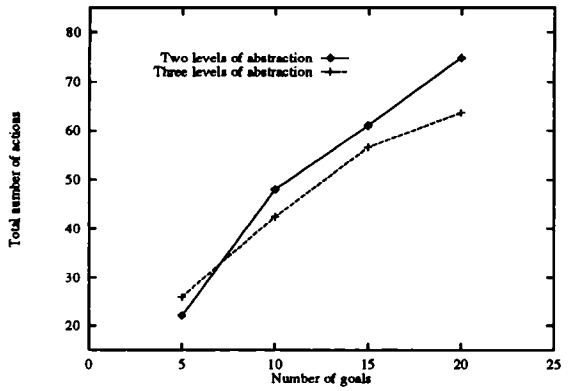


Figure 10: Comparison of numbers of actions using two and three levels of abstraction

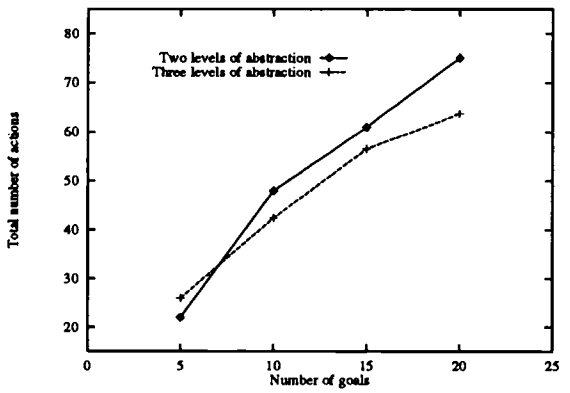


Figure 11: Comparison of numbers of goals recognitions using two and three levels of abstraction.

nomenon has been witnessed in only rare cases in our experiments. In general, the sooner the greedy agents start pursuing the same goals, the less efficient is their solution. The goal recognition based strategy, on the other hand, prevents agents from pursuing the same goals in most cases.

The results for the multiple-rooms domain is shown in Figure 10 and Figure 11. The total number of actions slightly decreases with the use of more levels of abstraction (see Figure 10). The total number of goal recognitions decreases significantly with more levels of abstraction. For example, when agents are working on different goals of the first level of abstraction, it is guaranteed that their second level goals do not overlap. Therefore, the goal recognition needs to be done only at the first level of abstraction. Goal recognition on the second level of abstraction needs to be done only when the agents start working on the same first level goal, that happens at the very end of the cleaning process when there is only one room left for cleaning. There are fewer first level goals than second level goals, so the goal recognition on the second level does not happen as often. Also, the goal recognition on the first level is likely to be less expensive than the goal recognition on the second level. This occurs because there are fewer alternative goals on the first level than on the second level.

Related Work

A variety of approaches have been developed to reach coordination through multi-agent planning. These strategies allow the agent to consider activities of other agents. They include collective multi-agent plans, that are constructed before agents start to pursue their goals (Corkill 1979) (Ephrati & Rosenschein 1996) (Durfee & Montgomery 1991); partial collective plans that are constructed during the execution (Durfee & Lesser 1991). These planning approaches use communication between agents to resolve conflicts, exchange partial information, etc. We use distributed planning in our research, where each of the agents constructs its individual plan based on its model of the current goals of other agents. Thus agents can coordinate without using communication. Huber and Darfee (Huber & Durfee 1996) used plan recognition to anticipate what other agents do. They use probabilistic goal recognition based on specifically instantiated belief networks. This work is the closest to our research. But instead of requiring the goal recognizer to return the most probable goal, we are using the goal recognition to identify likely goals that are then eliminated from this agents' goal set for now. Some of the best known formal works in goal recognition area are those of Kautz (Kautz 1987), and Pollack (Pollack 1990). They define goal recognition as the task of inferring actor's plans and goals given a partial view of their behavior. Carberry (Carberry 1990), and Charniak and Goldman (Charniak & Goldman 1991) used probabilistic models for goal recognition. Probabilistic goal recognizers typically run in exponential time and require a complete plan library as input. It makes them impossible to incorporate into real world applications. The more practical goal recognizer was built by Leash and Etzioni using generalization technique borrowed from machine learning (Lesh & Etzioni 1995). Our research concentrated on making the task of goal recognition easier by relaxing this requirement to return a single most probable goal as the result. We believe that any goal recognizer can be incorporated a system that uses our approach.

Conclusions and Future Work

In this thesis we have shown that agents using our goal-recognition based strategy significantly improve per-

formance over greedy agents that ignore the observed behavior of other agents. The performance improves even if only one of the agents uses goal recognition, and the performance improve drastically if both of the agents use goal-recognition. We also generalized our approach to incorporate planning and goal recognition at multiple levels of abstraction. We showed that increasing the levels of abstraction further improves performance and reduces efforts spent on goal recognition and planning.

The approach to goal recognition described in this thesis can be used with observing agent that uses a traditional goal recognizer. The goal recognizer builds a plan library and eliminates plans and corresponding goals from the library as it observes the actions of the other agent. These eliminated goals become the active goals of the observing agent. Thus, the approach described in this thesis is general enough to be used with different kinds of goal recognizers, and hence can be used in other domains. It also can be generalized to work for both cooperative and competitive domains.

We plan to try our approach for conceptually different domains with possibly more than three levels of abstraction. We also plan to implement our approach for a competitive environment. For example, the room cleaning domain can be changed so that the agents get rewarded for picking up each waste-basket. In this case the agents compete for the waste-baskets. We plan to investigate the usefulness of pursuing complementary goals in this domain and the strategy for calling the goal recognizer.

We plan to analyze the tradeoff of computational cost savings and performance between our proposed approach and that of calling the goal recognizer at every time step. We also plan to scale up the experiments to investigate the effects of more than two agents in the environment.

References

Carberry, S. 1990. Incorporating default inferences into plan recognition. In *Eighth National Conference on Artificial Intelligence*, 423–429. AAAI Press/The MIT Press.

Charniak, E., and Goldman, R. 1991. A probabilistic model of plan recognition. In *Ninth National Conference on Artificial Intelligence*, 160–165. AAAI Press/The MIT Press.

Corkill, D. 1979. Hierarchical planning in a distributed environment. In *Sixth International Joint Conference on Artificial Intelligence*, 168–175. Tokyo, Japan: AAAI Press/The MIT Press.

Durfee, E., and Lesser, V. 1991. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics 21(5) (Special Issue on Distributed Sensor Networks)* 1167–1183.

Durfee, E., and Montgomery, T. 1991. Coordination as distributed search in a hierarchical behavior space. *IEEE Transactions on Systems, Man, and Cybernetics 21(6)* 1363–1378.

Durfee, E. 1995. Blissful ignorance: Knowing just enough to coordinate well. In *First International Conference on Multiagent Systems*, 406–413. Menlo Park, CA: AAAI Press/The MIT Press.

Ephrati, E., and Rosenschein, J. 1996. Multi-agent planning as a dynamic search for social consensus. In *Thirteenth International Joint Conference on Artificial Intelligence*, 423–429. AAAI Press/The MIT Press.

Huber, M., and Durfee, E. 1995. Deciding when to commit to action during observation-based coordination. In *First International Conference on Multiagent Systems*, 163–170. Menlo Park, CA: AAAI Press/The MIT Press.

Huber, M., and Durfee, E. 1996. An initial assessment of plan-recognition-based coordination for multi-agent teams. In *Second International Conference on Multiagent Systems*, 126–133. Menlo Park, CA: AAAI Press/MIT Press.

Kautz, H. 1987. *A Formal Theory Of Plan Recognition.* Ph.D. Dissertation, University of Rochester.

Lesh, N., and Etzioni, O. 1995. A sound and fast goal recognizer. In *Fourteenth International Joint Conference on Artificial Intelligence*, 1704–1710. San Mateo, CA: AAAI Press/The MIT Press.

Pollack, M. 1990. Plans as complex mental attitudes. *Intentions in Communication* 163–170.