Plan Recognition for Real-World Autonomous Robots: Work in Progress

Marcus J. Huber and **Edmund H. Durfee***
Distributed Intelligent Agents Group (DIAG)
Artificial Intelligence Laboratory
The University of Michigan
Ann Arbor, Michigan 48109-2110
marcush@engin.umich.edu, durfee@engin.umich.edu

Abstract

An agent operating in the real world must be able to coordinate its activities with those of other agents. Traditionally, work in multiagent coordination has assumed that agents can communicate about their intentions or that they are coordinated through the efforts of a third party. In many environments, however, reliance on communication or on a coordinating agent is infeasible due to the unpredictable nature of the environment or to negative side-effects of communication. We have developed a multiple-resolution hierarchical scheme by which an agent can use observations to infer the high-level goals of other agents. The research domain to which we have applied our scheme is coordinated motion and navigation among multiple robots, both in simulation and in the real-world. Our hierarchical scheme makes probabilistic plan recognition possible in this domain, and it helps to further identify and begin solving crucial issues in plan recognition in physical domains.

Introduction

Autonomous agents[1] operating in the real world will not be able to perform their tasks in isolation. At some point in time, these robots will come into contact with other autonomous robots, either intentionally or coincidently. The robots will then not be working merely within the confines of the already very difficult dynamic and Murphy's Law-driven environment, but within a larger scheme wherein other intelligent agents must be dealt with. Conflict over shared resources (time, space, unique items, etc.) will eventually occur, and opportunities for synergistic cooperation might arise. If the agent is expected to accomplish its goals in multiagent situations, it must coordinate its plans with the plans of others.

To coordinate its plans with those of others, an agent must have a model of the plans of each of the other agents. Traditional techniques in multiagent planning and coordination [Cammarata *et al.*1983, Durfee and Lesser1987, Georgeff1983] allow agents to explicitly communicate about their intentions and plans. However, some environments might not admit to such explicit communication, either because the communication medium might be unreliable (sporadic interference) or using the communication medium might introduce new risks to the agents (such as being detected by unfriendly agents). For this reason, agents might prefer to attempt to infer the plans of each other by passively observing the actions or effects of other agents rather than actively communicating plans.

As in any plan recognition system, the goal is to infer the high-level plans of another agent based upon observations of the other agent's behaviors. While a predominant portion of plan recognition research has been in the context of such domains as discourse analysis and story understanding (e.g. [Charniak and Goldman1990, Lochbaum *et al.*1990]), very little research has gone into applying the paradigm to autonomous robots in real-world situations. We have implemented a plan recognition system for multiple autonomous robots in a simulator and in the real world to get a better understanding of the important issues that face coordination through observation among autonomous robotic systems.

The simulated system operates within a MICE [Montgomery and Durfee1990] controlled environment, where agents move to and fro in a two dimensional grid world. Sensors are simulated, noiseless, and precise. The implemented real-world system involves the operation of multiple mobile robots performing tasks in an enclosed area of the robotics laboratory at the University of Michigan. The robot is a Cybermotion mobile platform called CARMEL (see Figure 1) that is equipped with a computer vision system that it uses to observe the other agent(s) in its vicinity. The other agent(s) are TRC Labmate mobile platforms (see Fig-

*This research was sponsored in part by the NSF under grants IRI-9015423, IRI-9010645, and IRI-9158473, and by DARPA under contract DAAE-07-92-C-R012.

[1]We will use the words robots and agents synonymously throughout the paper as the given domain is a specific instance of the more general idea of multiple "intelligent" entities interacting in the real world.

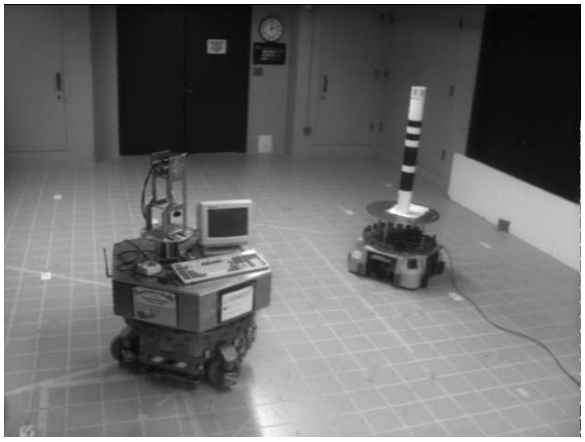

Figure 1: The real-world plan recognition testbed

ure 1—the tube on top is the means by which CARMEL identifies and tracks other agents). Given some observations, the observing robot tries to infer the goals of the other robots and then respond in the "correct" manner. Initially only one robot is being observed moving from a start location to a goal location. In both environments, the overwatching robot, having inferred possible final destinations, tries to move to a location that complements the most likely final destination (this could be the same location, an adjacent location, etc.)

The system's inferencing is done using a belief network implemented with the IDEAL system [Srinivas and Breese1989]. Evidence in the form of visually perceived actions is input into the network and then propagated through it. The high-level plans/goals of the watched agent can then be hypothesized from the results. These high-level goals are possible destination locations within the workspace. These are finite and enumerable, possibly some point of strategic or tactical interest to the watched robot. The watched robot's behavior is simply to determine the movement that will take it toward its final destination. Given that the overwatching robot can detect the location and motions of the watched robot, it can then try to infer the destinations that the robot is trying to reach and act appropriately.

In the following sections we will discuss the work done in developing the plan recognition system. We will start by discussing the representation scheme developed and used by the system, followed by a description of the belief network architecture and function, and then describe the results of several experiments that were performed to verify that the system worked as designed. In the final section we discuss some of the major extensions and modifications that we see for the immediate future.

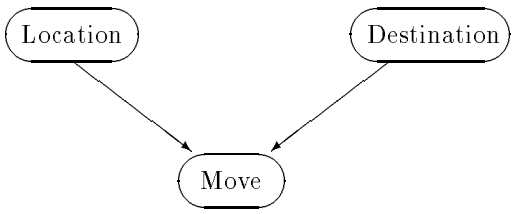

Figure 2: Belief Network Architecture

Plan Recognition

Belief Net Architecture

In order to perform the necessary accumulation of information and subsequent inferencing, we incorporated a belief network (also called a Bayesian network) [Pearl1988] into the system. This network permits the insertion and accumulation of sensor-based observations that get propagated throughout the network, resulting in the ability to hypothesize goals. The overwatching robot periodically senses the watched robot to determine the visible primitive actions (if any) that were performed by the observed robot. These observations are added to the history of previously observed actions, and the beliefs of the belief network updated. With consistent positive reinforcing evidence, belief in a possible destination increases, while, with consistent negative reinforcing evidence, belief in a possible destination decreases.

We show the simple belief network architecture in Figure 2. The states held by the *location* node are the region representations that describe the overall map. The states held by the *destination* node are the possible locations (in terms of x,y coordinates) that the overwatched agent might consider to be destinations. The *movement* node contains the primitive actions that the overwatching agent is looking for, namely movement in the cardinal directions of north, south, east, and west, as well as staying in the same position, from the last time observed. The belief network represents the idea that given a goal destination and the particular location that an agent is in, the next move is predictable. This assumes that the agent will move in a bee-line to the destination from the given location and therefore does not handle situations where the agent is trying to trick the watching agent by making feinting maneuvers or is taking a path that will take it around some obstacle that is in the way.

For the network:

1. prior probabilities have to be determined for each of the possible states of the independent nodes (the *location* and *destination* nodes).
2. conditional probabilities have to be determined for the dependent nodes (the *move* node).

For the *location* node, the state that represents the current position of the agent is set to the probability of 1.0 for the simulator version, since sensors are assumed to be perfect. We set this probability to some lesser value for the real world version, dependent upon how much we trust the accuracy of our sensor system. If the robot is computing the location relative to itself, which may be fairly accurate, the confidence in the sensor system may be quite high. On the other hand, if the calculated position is fixed to some absolute coordinate system, the errors accumulating over time in the watching robot's own odometry continually degrades the accurate positioning of the watched robot. In this case, the confidence level will drop over time until such time as the agent has a chance to get its bearings and reposition itself. The overwatching agent initially considers each of the destination locations as equiprobable, with adjustments to the beliefs occurring through the propagation of belief after evidence has been accumulated through observations.

The conditional probabilities (e.g. the probability that the observed agent, given its current location and assuming that it is trying to reach destination location 1, will move toward the east) are calculated based as a normalized discrete approximation of the relative likelihood of moving in a particular direction while moving to any location within a destination region from anywhere within the watched agent's current region.

Hierarchical Topology Representation

Our work to date has dealt with agents moving and navigating through a flat world. As such, we have needed a representation of the area in which the agents are moving, with particular distinction given to special goal locations (those deemed interesting for some reason). For operation in very small areas, or where the granularity of representation can be quite large, enumeration of possible locations (e.g. at centimeter intervals) might be useful. Larger areas, or the need for a finer granularity of representation, require a different approach—some form of abstraction—as the system can become bogged down by the great number of explicitly represented states required at each node of the belief network.

One early product of the research has been the development of a representation for the environment in which the robots will operate. Belief networks suffer greatly from the combinatorial explosion of the number of conditional probabilities required as the number of states in connected nodes increases. A finer resolution representation requires a larger number of current (location) and goal (destination location) states to represent the same area. While other hierarchical schemes already exist (e.g. oct/quad trees, K-d trees), a simple hierarchical representation scheme more suited to the domain was developed for the initial scenarios. A mechanism to dynamically modify the belief network was developed in conjunction with this.

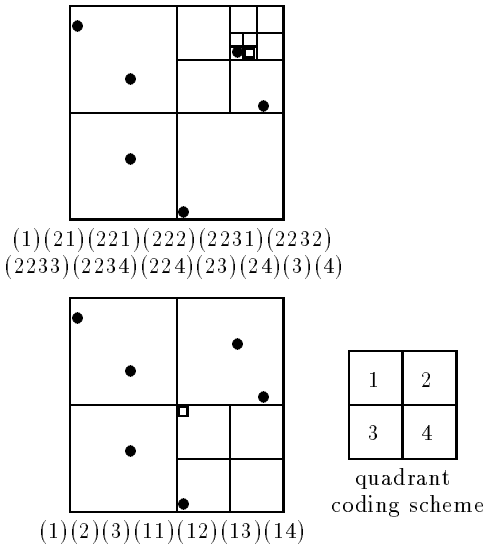

Figure 3: Examples of representations. The filled circles are possible destination locations, the hollow square is the observed robot.

We opted for a simple scheme reminiscent of quad trees. The entire area of operation is represented at multiple resolutions, with a section being marked as whether it is interesting (i.e. contains within its bounds a watched agent or a possible destination) or uninteresting. Each quadrant of a region (square for now) may be further divided into four sections. The level of resolution required for each portion of the entire map is kept at a minimum; a section of the map is further divided only if it contains within its bounds both a watched agent and one or more possible destinations. The overwatching agent needs the distinction between them so that it can determine whether or not a watched agent is sitting at a destination location. Examples of several maps and their representations are given in Figure 3.

Dynamic Modification

The world in which robots will work *will* change dynamically. A particular representation of the world, then, will only work for a limited time before it becomes inaccurate, invalid, etc. Representations of dynamic portions of the world, then, will need to change dynamically themselves. We have implemented the representation scheme so that it undergoes dynamic change as the robots move about (see Figure 4). At each sensor cycle (synchronized with agents' actions in the simulator but variable in the real implementation) the overwatching robot checks the current world model to see if the representation is valid and minimal. Validity depends upon the criterion of being able to distinguish between watched robots and destinations in the representation while a minimal representation is as described above. If either of these criteria are not met, then modifications to the representation are made, with correspond-

ing changes to the belief network that uses the representation.

As the representation changes to accommodate the above criteria, the beliefs accumulated up to that time are redistributed from the old representation to the new representation. When abstraction of a quadrant occurs, the belief in the parent region is the sum of the beliefs of the child regions. Similarly, when a region of the representation is broken into its corresponding quadrants, the belief assigned to the parent region is equally divided between the child quadrants that potentially contain destination locations.

Representational Effects on Performance

Early experiments were conducted using belief networks in which the entire mapped area was represented at a single, high, resolution. The resulting performance was very poor and became intolerable once larger areas were tried. A lower resolution of representation was an option, but not a viable one, as only correspondingly large motions could then be detected.

The largest single factor in the performance of the system is the number of states held at each node in the network. The number of conditional probabilities required is the cross product of all of the states in the *location*, *destination*, and *move* nodes. Even more significantly, the time required to propagate evidence throughout a belief network increases exponentially, so that representing a large map at a single, high resolution would be impractical.

The multiple resolution scheme implemented dramatically improves the performance of the system. The number of states required to represent a region increases only by three for every doubling of the size of the region. This compares to a fourfold increase for a single resolution scheme. At the time that the change in the representation took place we were running experiments (in MICE) with agents in a 15x15 grid. The new representation scheme was tried using a 32x32 grid, with a resulting performance improvement of a minute-and-a-half compared to four seconds, a 22.5 times improvement. And this was a direct comparison; the constant resolution scheme was never run on a 32x32 grid. Had it been, two orders of magnitude of improvement would not have been surprising.

While the multiple resolution representation scheme results in a large increase in the performance level of the system, there exists at least one major deficiency; there exists a potential for a dramatic change in the representation whenever the watched agent crosses a quadrant boundary. The map representation remains constant as long as no borders in the current representation are crossed by the overwatched agent. But, as described above, the system dynamically changes the representation throughout a run. The significance of the representation change depends upon how close to a possible destination the overwatched agent is after it crosses a boundary, compared to how close to a destination it was just before crossing the boundary. This is illustrated in Figure 4. For this reason we are looking at alternative representations that do not suffer from this effect but maintain the same level of performance increase as the implemented scheme.

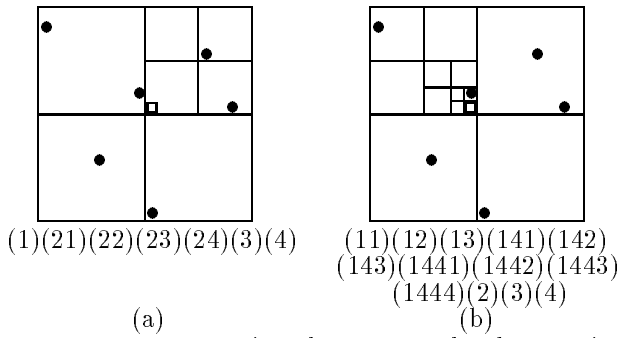

Figure 4: Representation change upon border crossing. The circles are possible destination locations, the square the observed robot moving from right to left. Note the change in representation of region (1), in which it goes from being represented in its entirety to being broken into 10 regions.

The Real World

Any robot hoping to autonomously perform even simple tasks in a real-world environment, especially one in which other autonomous agents exist, must have some sensing capabilities. This may not be the case in a static environment where the robot may be given a thorough and complete model of the world and can therefore be assured that a generated plan will not fail due to something changing outside of its control. The real world, however, seldomly stays the same for any length of time and cannot be thoroughly and completely modeled. The ability to sense the environment and reason about the results is therefore crucial if a robot is to be flexible enough to perform effectively.

An autonomous agent working with other agents in the real world can use its sensing capabilities to facilitate coordination and cooperation between agents. The information gained from the sensors can augment any existing information that agents have of each other and can even possibly detect contradictions between the information sources. The problem with sensor-based information is the inherent noise and uncertainty that accompanies it. No sensor returns perfect information and the data-to-symbol conversion also results in information loss. Dealing with such issues is a necessity for any robot working in the real world.

The belief network architecture that we have implemented handles these additional problems naturally. Due to the probabilistic basis for the system, the un-

certainty that pervades the real world can be incorporated quite easily. As an example, instead of knowing with surety (i.e. with probability of 1) the location of another agent, there may be some degradation in this certainty (some probability less than 1) and nearby locations now have some chance of being correct. The probabilities actually employed depend upon a combination of the accuracy of the sensing system used to gather the information, the process by which the information is abstracted, the dynamic nature of the environment, etc.

Factors that influence the plan recognition process when operating in the real world include:

- Sensor Noise - Every real sensor system adds some amount of noise to the information. Sometimes this may be known and modeled, but the model is generally not perfect. Therefore the noise is not totally eliminated from the data. This may have a great influence on the effectiveness of a plan recognition system if the observing agent needs to detect subtle differences between the actions and/or behaviors of other agents.
- Sensor (in)accuracy - All real sensors are limited in their accuracy (e.g. due to finite resolution), with results like that above, an agent may have difficulty differentiating between actions.
- Sensor failure - Real sensors fail in any of a number of ways. For example, they might simply fail to work, rendering any sensor-based plan recognition system useless (discourse or other information source-based plan recognition would still be possible, of course). Or a sensor failure may result in an an unexpected change in the information in a manner that is very difficult to detect. Or they might fail spuriously, causing the sensing system to occasionally respond with wildly erroneous results.
- Sensor to symbol inaccuracy - The abstraction process by which raw sensor data is symbolized may not be completely accurate, creating small discrepancies between what happened in the the real world and what the robot actually thinks happened. This may have effects similar to that in 1. and 2. above.
- Sensor to symbol error - The process of abstracting the raw data to the symbolic level might not be correct due to lack of complete knowledge of the complex world.
- Sensor to symbol loss of information - Whenever raw sensor data is abstracted to the symbolic level some amount of information is inevitably lost. This loss of information may or may not have some effect upon the plan recognition process.
- Dynamic environment - The world in which an autonomous robot will live will be constantly changing. Topography may change dramatically in a small amount of time so that (at least in this domain) an agents high-level goals (i.e. goal locations) might change just as dramatically. If the overwatching agent is not able to deal with this it may become hopelessly confused trying to figure out the high-levels goals of overwatched agents.
- Odometry error - A robot will naturally build up some amount of error in its own position over time. Without compensation for this, (by periodic redefinition of its global position, modeling of the progression of the error, etc.), the robot will not be able to accurately calculate the current positions of observed agents or goal locations.
- Time constraints - The real world may not be quite as forgiving as a simulator when dealing with time. There is a need to process incoming information quickly enough to deal with changes in the environment in a timely manner. The inability to do this may result in not observing some of an overwatched agents' actions/behaviors.
- Agent concurrence - Agents in the real world generally work in a truly parallel, asynchronous manner. Simulator-based assumptions of pseudo-parallelism and synchronization of agent's actions must be thrown out, perhaps resulting in a radical modification of a simulator-developed system that did not look forward to this issue.
- Commitment - Early commitment to what erroneously appears to be the watched agent's high-level goal may lead to behavior on the part of the overwatching agent that is very costly, perhaps irrevocable, and which places the overwatching agent in a position from which it cannot accomplish its task. Conservatism, at some level, therefore, must be embedded within the system, so that the correct balance is made between the cost of waiting for hypotheses with more surety and the benefit from quickly inferring the correct high-level goal.

The Plan Recognition Cycle

The plan recognition system that has been implemented operates in a cyclical manner. CARMEL makes an observation of the other robot with the computer vision system. The resulting image is processed, extracting the relative orientation and distance of the TRC from CARMEL. Absolute coordinates are then calculated for the TRC based upon these values and CARMEL's own position (based upon dead-reckoning). CARMEL then calculates the primitive action(s) performed by the TRC since last being seen. This information is then fed into the belief network as observations and propagated through the network to update the destination location beliefs.

Experimental Results

A series of experiments were run in the simulator version of the system to explore the tradeoff of commitment versus cost. In the domain in which we conducted