Extending the Partial Global Planning Framework for Cooperative Distributed Problem Solving Network Control [1] [2]

Keith Decker Victor Lesser
Computer and Information Science Department
University of Massachusetts

COINS Technical Report 90–81
October 30, 1990

Abstract

Generalized Partial Global Planning is an attempt to extend Partial Global Planning (PGP) to other domains and provide a framework for new coordination algorithms. It is based on the observation that the relationships the PGP mechanism uses to measure increased network coordination can be defined for, and detected in, arbitrary cooperating systems. This paper describes several issues that will be important in extending the PGP mechanism to heterogeneous, dynamic, and "real-time" agents, and in developing new mechanisms such as negotiation. A scenario is described that illustrates these issues and which can be used as a basis for experimenting with solutions to them. The approach taken emphasizes the detection and classification of goal relationships between agents.

[1]This work was supported by DARPA contract N00014-89-J-1877, and partly by the Office of Naval Research under a University Research Initiative grant, number N00014-86-K-0764, NSF-CER contract DCR-8500332.

[2]Portions of this technical report appeared in the proceedings of the Ninth and Tenth International Workshops on Distributed Artificial Intelligence, September 1989 and October 1990.

1 Introduction

The partial global planning (PGP) approach to distributed network control increased the coordination of agents in the network by avoiding redundant activities, shifting tasks to idle nodes, and providing predictive results that reduced overall problem solving time by estimating the duration of tasks to allow schedules to be built from interleaved plans [Durfee and Lesser, 1987b]. Generic Partial Global Planning tries to extend this approach by communicating more abstract information (goals, capabilities) and detecting the relationships that are needed by the partial global planning mechanisms. In the same way that contract nets [Davis and Smith, 1983] provide a domain-independent task allocation mechanism, we are proposing to build a generic network control system that will provide support for: modeling agents' capabilities, desires, and intentions; modeling the relationships between these; and building network controllers based on these models.

The global coherence problems we would like to address occur in many systems, such as the Pilot's Associate system [Smith and Broadwell, 1987], where situations occur that cause potentially complex and dynamically changing goal relationships to appear in goals that are spread over several agents. Each agent in the Pilot's Associate system has subgoals that other agents must fulfill, and receives subgoals from other agents that only it can fulfill.

For example, assume that we are in a tactical situation, so the tactical planner is in control (see Figure 1). It has two ordered subgoals: turn on the active sensors (request to situation assessment), and get a detailed route of the plane's movements during the tactical maneuver (request to the mission planner). Turning on active sensors causes a plane to become a transmitter, and thus become easily detected (most of the time the plane uses passive sensors). Since this is dangerous, the situation assessment agent will ask the pilot-vehicle interface (PVI) to ask for pilot confirmation of the use of active sensors. The pilot, upon seeing the request, asks the PVI to plot the escape route of the plane on the screen in case things go wrong. The PVI passes this goal to the mission planner.

Meanwhile, the tactical planner has asked the mission planner to produce the detailed route for the tactical maneuver. Which task does the mission planner act on first? From a local view, it may perhaps do the tactical planner request first because the tactical planner goals are a high priority. But from a global perspective, we see that unless the mission planner plans the escape route, which is needed by the pilot in order to authorize turning on the active sensors, which is needed for the tactical planner to do its job, the whole system goal of handling the tactical situation is in jeopardy. Hence the mission planner should do the escape route plan first.

To handle these interactions, we are developing a set of generic relationships among goals that fall into four general categories: domain relations, graph relations, temporal relations, and non-computational resource constraints. By communicating goals at different levels of abstraction we can reduce the amount of communication required between agents by communicating detail only when necessary.

Section 2 briefly reviews the existing PGP mechanisms and why they

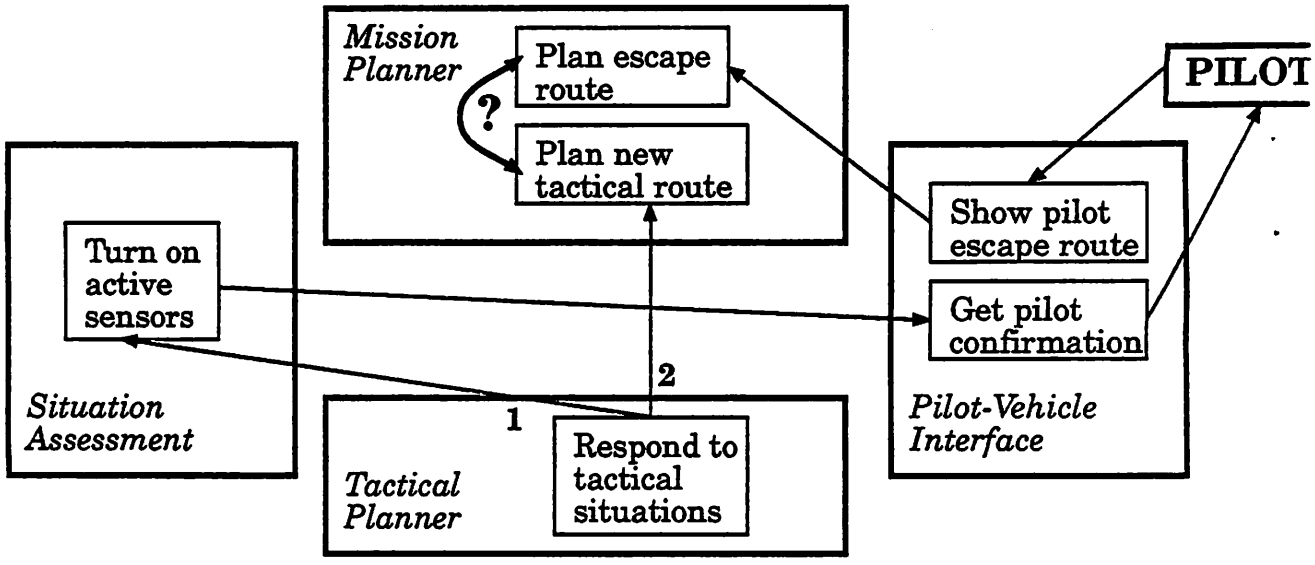

Figure 1: Dynamic Situations in Pilot's Associate

work. However, we want not only to apply the PGP approach to other areas, but also to extend it. In developing a complete approach to distributed coordination, we found several other areas in which we would like to extend the PGP mechanisms. These issues include *heterogeneous agents* (which may have different problem solving criteria), *dynamic agents* (which have several different strategies available for problem solving and several different methods for accomplishing goals), *real-time agents* (which have hard goal deadlines), and *negotiating agents* (which are a consequence of the conflicts inherent in any and all of the other issues). Section 3 discusses these issues in greater detail, defining them and how they relate to the original PGP mechanisms. A scenario is presented in Section 4 that illustrates these issues in the Distributed Vehicle Monitoring Testbed (DVMT). Finally, an initial approach is outlined in Section 5 that describes how we intend to integrate the goal relationship mechanisms into the DVMT and how to use them to achieve a more flexible PGP mechanism.

2 Partial Global Planning

Partial global planning [Durfee and Lesser, 1987b, Durfee and Lesser, 1989] was developed as a distributed control technique to insure coherent network problem solving behavior. It is a flexible approach to coordination that does not assume any particular distribution of subproblems, expertise, or other resources, but instead lets nodes coordinate in response to the current situation. Each node can represent and reason about the actions and interactions of groups of nodes and how they affect local activities. These representations are called **partial global plans** (PGPs) because they specify how different *parts* of the network *plan* to achieve more *global* goals. Each node can maintain its own

set of PGPs that it may use independently and asynchronously to coordinate its activities.

A PGP contains an objective, a plan-activity-map, a solution-construction-graph and a status:

- The **objective** contains information about *why* the PGP exists, including its eventual goal (the larger solution being formed) and its importance (a priority rating or reasons for pursuing it).
- The **plan-activity-map** represents *what* the nodes are doing, including the major plan steps the nodes are concurrently taking, their costs, and expected results.
- The **solution-construction-graph** contains information about *how* the nodes should interact, including specifications about what partial results to exchange and when to exchange them.
- The **status** contains bookkeeping information for the PGP, including pointers to relevant information received from other nodes and when that information was received.

A PGP is a general structure for representing coordinated activity in terms of goals, actions, interactions and relationships.

When in operation, a node's PGPlanner scans its current network model (a node's representation of the goals, actions and plans of other nodes in the system) to identify when several nodes are working on goals that are pieces of some larger network goal (partial global goal). By combining information from its own plans and those of other nodes, a PGPlanner builds PGPs to achieve the partial global goals. A PGPlanner forms a plan-activity-map from the separate plans by interleaving the plans' major steps using the predictions about when those steps will take place. Thus, the plan-activity-map represents concurrent node activities. To improve coordination, a PGPlanner reorders the activities in the plan-activity-map using expectations or predictions about their costs, results, and utilities. Rather than examining all possible orderings, a PGPlanner uses a hill-climbing procedure to cheaply find a better (though not always optimal) ordering. From the reordered plan-activity-map, a PGPlanner modifies the local plans to pursue their major plan steps in a more coordinated fashion. A PGPlanner also builds a solution-construction-graph that represents the interactions between nodes. By examining the plan-activity-map, a PGPlanner identifies when and where partial results should be exchanged in order for the nodes to integrate them into a complete solution, and this information is represented in the solution-construction-graph.

The PGPlanner, as it was used for coordination in the distributed vehicle monitoring task, relied on the fact that the level of abstraction at which the node plans were communicated was a sequential sequence of intermediate goals (times and locations in which to extend a vehicle track). Each intermediate goal was an abstraction of the processing and integration work that each node planned for locally. These intermediate goals were ordered by the local node planners based on several criteria [Durfee and Lesser, 1988], but these relationships are not transmitted in the node plans. There was no

representation of temporal relationships between intermediate goals. The PGPlanner reorders node activities by hill-climbing in the space of costs of the present ordering of activities. The cost of an ordering is computed from relationships like redundancy, reliability, predictiveness, and independence of the activities.

Why does partial global planning work well in the DVMT? It is because:

- It avoids redundant work among nodes by noticing interactions among the different local plans. Specifically, it notices when two node plans have identical intermediate goals, i.e., when they are working on the same time region. This occurs in the DVMT because in the interests of reliability nodes have overlapping sensors.

- It schedules the generation of partial results so that they are transmitted to other nodes and assist them at the correct time. To do this it uses the estimates of the times that activities will take and the inferred relation that if node $\mathcal{A}$ estimates that it will take less time than node $\mathcal{B}$ to complete an intermediate goal, and the goals are spatially near, that node $\mathcal{A}$ can provide predictive information to node $\mathcal{B}$.

- It allocates excess tasks from overloaded nodes to idle nodes. Node plans provide the information needed to determine if a node is overburdened or underutilized. A node is underutilized if it is either idle or participates in only low-rated PGPs. A node is overburdened if its estimated completion time of a subgoal of goal G is much later than the completion time of all the other subgoals of G [Durfee and Lesser, 1989].

- It assumes that a goal is more likely to be correct if it is compatible with goals at other nodes. In the DVMT task, a goal represented a processing task to ascertain whether a vehicle was moving in a region r at time t. This goal could, in fact, be wrong — based on noise or errorful sensor data that was the basis for the preliminary task analysis that generated the goal. Nodes choose local plans to work on based on the highly rated PGPs they have received. Thus, if the intermediate goals of a node become part of a PGP, then they are worked on before other intermediate goals in other local plans the node may have (even though the node may have rated those local plans higher in its local view).

To control how they exchange and reason about their possibly different PGPs, nodes rely on a meta-level organization that specifies the coordination roles of each node. If organized one way, the nodes might depend on a single coordinator to form and distribute PGPs for the network, while if organized differently, the nodes might individually form PGPs using whatever information they have locally. The partial global planning framework lets nodes converge on common PGPs in a stable environment (where plans do not change because of new data, failed actions, or unexpected effects of their actions). When network, data, and problem-solving characteristics change and communication channels have delay and limited capacity, nodes can locally respond to new situations, still cooperating but with potentially less effectiveness because they have somewhat inconsistent PGPs [Durfee and

Lesser, 1987a]. The PGP framework does not, however, deal with conflicts in non-computational (physical) resources.

3 Issues in Extending the PGP Mechanisms

3.1 Heterogeneous Agents

How can the PGP mechanisms be extended to handle agents that have different local problem solving criteria? This can arise in several ways:

- Some agents in the system are humans with local (personal) decision criteria that cannot be adequately or fully modeled.
- Some agents in the system have different expertise, and hence different local decision criteria (cooperative design problems [Lander and Lesser, 1989], pilot's associate-style problems [Smith and Broadwell, 1987]). The PA scenario in Section 1 is a classic example of heterogeneous agents with shared global goals and differing local expertise.

The PGP mechanism assumes a shared local and global decision evaluation function (so that all agents, given the same information and enough time, will arrive at the same decisions). Conflict between agents comes about because some agents lack data or have out-of-date data. Agents do not have to exchange or negotiate about decision criteria. While this well-documented assumption simplified the PGP mechanism, a homogeneous agent assumption (where the local decision criteria are shared) is not always appropriate. The PGP mechanism also assumes that the agents will pursue one goal at a time — the goals are ordered, and if an agent has excess capacity it can fill it with tasks from lower-rated goals. Planning for the simultaneous achievement of multiple goals is not supported.

The modularity of the PGP mechanism (which separates the local agent's incremental planner [Durfee and Lesser, 1988] from the PGPlanner) comes close to permitting heterogeneous local decision criteria. The only problem arises when the PGPlanner reorders the node plan for another agent. The PGP plan evaluation function that was used to develop a global schedule contains terms to avoid upsetting the order of another agent's plan (*independence* measured the distance of the current ordering from the original node plan ordering, *locally-predicted* measured the distance of the current ordering from regular time order). In some domains a portion of this ordering may be fixed. We have suggested marking temporal goal relationships as *hard*, *negotiable*, and *soft* (see Section 5.2). This allows the plan evaluator to rule out certain impossible orderings (hard constraints), and to avoid those that may cause replanning at the target node (negotiable constraints).

3.2 Dynamic Agents

How can the mechanisms be extended to handle agents that have a great deal of latitude in the methods that they use to solve problems? Each method may have a different effect on the characteristics of the solution, such as

completion time or certainty. These agents can appear in human systems and systems where agents use approximate processing techniques [Decker *et al.*, 1990b]. In the PA scenario in Section 1, the mission planner might solve its dilemma by using different algorithms to respond to each plan request. A fast but inaccurate algorithm may suffice to give the pilot an idea of a corridor of escape, while a more complex and precise algorithm can be given the bulk of the computational resources with which to plan the near-term tactical maneuver.

Because only one method existed for accomplishing a goal (and no set of different criteria existed for determining what would be considered an acceptable solution), the PGP mechanism could equivalently exchange goals and the plans to accomplish those goals, at a single level of detail. The node plans that were exchanged indicated the goal of an agent to produce a track with certain characteristics (classes, sensed times, and regions) and a plan consisting of the ordering of the sensed times at which the agent would work (called *i-goals*), expected i-goal durations, and a mapping of the i-goal start and end times with respect to node problem solving time.

Two extensions need to be made. First, communicating goals at a single level of detail is inappropriate in more complex domains; certainly the detection of the interactions of two goals ("partial global goals") will not always be simple [Robinson and Fickas, 1990]. Secondly, many different methods may exist for accomplishing a goal, each with its own effects on duration, precision, and other goal characteristics. This makes the existing PGP node plan structure change rapidly when problem solving methods are changing dynamically (as an agent reacts to the problem being solved). The node plan structure can be modified to hold ranges as well as a best current estimate for a value, but it is also likely that agents will have to reason and perhaps negotiate about predictability versus reliability issues as well [Durfee and Lesser, 1987a]. The node plan structure could also be expanded with contingency plans for "routine expectation failures" [Dean, 1987] to allow for predictability in the face of a changing environment.

3.3 Real-time Agents

What happens when time becomes an integral part of local and shared goals? Dynamic agents will be able to modify both task durations (perhaps trading them off for other goal characteristics) *and* the goal deadlines themselves. In the PA scenario in Section 1, the mission planner's dilemma arises from the fact that it is under real-time constraints — if there were no impending deadlines for the pilot and tactical planner, the mission planner would have little reason to prefer one allocation of its computational resources over another.

While the PGP mechanism estimated the times for tasks or goals to be completed in order to spot idle processing resources, it did not handle deadlines. I-goals had expected durations; node-plans anchored (mapped) the completion of the various i-goals to a plan activity map. Experiments were conducted with the local incremental planner that did indicate the ability to plan to meet deadlines in a single agent [Lesser *et al.*, 1988, Decker *et al.*, 1990b].

In extending the architecture to so-called "real-time" problem-solving, agents may have goals with hard deadlines, which add constraints to the

construction of a plan activity map. Furthermore, the addition of hard deadlines or other domain constraints changes the nature of the interaction between a node's local problem solving mechanism and the PGP mechanism — some of the local ordering will remain local preference but some may be due to hard constraints, as discussed in Section 3.1 above. From the classical perspective, real-time network control also means scheduling both periodic and non-periodic tasks to deadlines; the original PGP mechanism did not deal with periodic tasks. The existing hill-climbing algorithm for scheduling may no longer be appropriate.

Often in real time situations planning is *reactive*, where the current situation mostly controls an agent's actions (where the "current situation" may include both local and global information), rather than *reflective*, where a sequence of actions is planned out in some detail before execution. This is because the agent must respond quickly, but more importantly, the agent may be too uncertain of the outcomes of its actions and of the changing world state to plan too far into the future. However, an intelligent agent will make use of periodic tasks, which occur in a predictable fashion, and known non-periodic tasks, to build a opportunistic planning framework that can keep an agent from painting itself into a corner with purely reactive planning techniques, or from exhaustively planning uncertain future details with reflective planning techniques.

Many new mechanisms are being put into place in the DVMT to allow the control of real-time problem solving (with hard deadlines); many of these mechanisms should integrate easily with the ones described here. For example, the original PGP mechanism had to have its own time estimation routines. We envision that the real-time mechanisms being developed for a single agent will be able to provide such services to the new coordination mechanisms we are also developing.

3.4 Negotiating Agents

A direct consequence of heterogeneous, dynamic, and real-time agents is the need for negotiation to solve conflicts. Even with a known global decision evaluation function, conflicting decisions of equal global value may have very different local value to the agents. Often the character of an early partial solution will have an impact on what style of coordination is needed. For example, if early partial results show poor data and low beliefs, the coordination mechanism may want to encourage redundant derivations of results in areas shared by more than one agent, or the parallel derivation of a result by two agents using different algorithms. The PA scenario in Section 1 probably occurs in too short a time-frame to allow negotiation between the agents, but other PA scenarios might profitably use negotiation techniques[1].

The PGP mechanism uses a shared global plan evaluation function that is parameterized. One extension is to allow the parameters (such as *redundancy*

[1]For example, a sensor may overheat and be shut down by the system status module, even though it is a projected resource requirement for some tactical situation. The tactical planner and system status may negotiate over the amount of time that the damaged sensor can be used if the situation arises.

and *reliability*) to vary during problem solving. A negotiation facility could be developed to allow agents to usefully alter the global (or perhaps only semi-local — we are interested in agents that may develop only a partial view of what other agents are working on) decision criteria. Where the PGP mechanisms exchanged all local information, our extensions would allow for a multi-stage process [Kuwabara and Lesser, 1989] where agents would communicate only the information believed relevant to the issue at hand. Agents could ask for more contextual information when it is needed to resolve a conflict between agents. Agents would not automatically acquire information from other agents performing non-related problem solving activities.

In order to examine all of the above issues more closely, a scenario is described below which exhibits the issues mentioned above.

4 Scenario

The scenario is based on the basic task of the DVMT: to track vehicles moving through an area via their acoustic signatures. To describe the scenario we will describe the tasks involved, the agents who will carry out those tasks, the environment within which they are acting, and the desired interactions.

The "vehicles" in the scenario are various aquatic creatures and waterfowl. One of the tasks involves protecting fish from fish-eating ducks. This includes detecting both fish and ducks and notifying the fish of potential duck attacks in time for the fish to escape. Another task involves simply tracking any pigeons in the area and displaying the tracks accurate to within certain spatial and temporal guidelines. These tasks are specified by "system goals" (see Section 5.1 and Appendix B).

The scenario demonstrates the issues described earlier in Section 3. Fish-protectors and pigeon-trackers are heterogeneous agents that have different utilities for the same data (see Section 5.1.1). The hard time deadlines will allow us to experiment with real-time performance and use approximate processing methods to give each dynamic agent a choice of different problem solving methods to choose from. The environmental data for the scenario will give the agents just cause for altering their plans during problem solving. Finally, the distribution and timing of the scenario provides several opportunities for the agents to negotiate. These issues are revisited in Section 4.4 after the scenario itself.

4.1 The Agents

There are 4 standard DVMT tracking agents with identical domain knowledge. Agents 1 and 3 are fish-protectors, and agents 2 and 4 are pigeon-trackers. Agent 3 has a faulty sensor, which will produce a large amount of noisy data. The four agents are shown pictorially in Figure 2. We will omit the definition of the agent's sensors and their precise coordinates. Also note that we can expand the scenario to more agents by tiling pigeon- and fish-agents in a natural way.

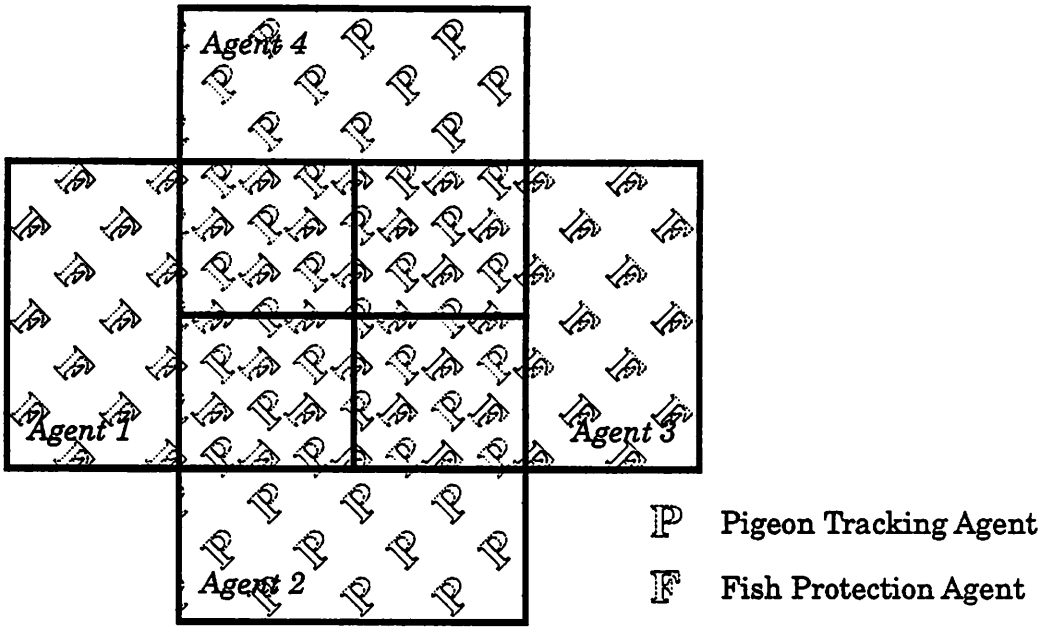

Figure 2: Four DVMT Agents: Two fish-protectors and two pigeon-trackers

4.2 The Environment

We will describe the environment by describing the patterns and vehicle tracks that are given to the environment simulator. These tracks can be seen visually in Figure 3.

At times 1–10 a fish F_1 travels through the area seen only by agent 1. The fish is initially meandering.

At times 1–10 a pigeon P_1 travels through the area of agent 2. It begins in the non-shared area and crosses into the area shared by agents 2 and 3 at time 6. The pigeon is in a meandering pattern.

At times 1–10 a duck D_1 travels through the area of agent 3. It begins in the non shared area and crosses into the area shared by agents 3 and 4 at time 3, and into the area shared by agents 1 and 4 at time 9. The duck is meandering.

At times 4–10 a duck D_2 travels through the areas of agents 4 and 1. It begins in agent 4's area only from times 4–5, then in the shared area of agent 4 and 1 at time 6 and 7, and then into agent 1's non-shared area. Duck D_2 is in an attack pattern with fish F_1 as the potential victim.

At times 6–10 a pigeon P_2 travels through the area of agent 1 and 4. It remains in the area covered by agent 1.

The actual grammar specifying the characteristics of these vehicles has been omitted. Several other things have been left deliberately unspecified, most notably the mapping from "real-world time" to processor time (so that we can experiment with how tightly pressed for time the agents are).

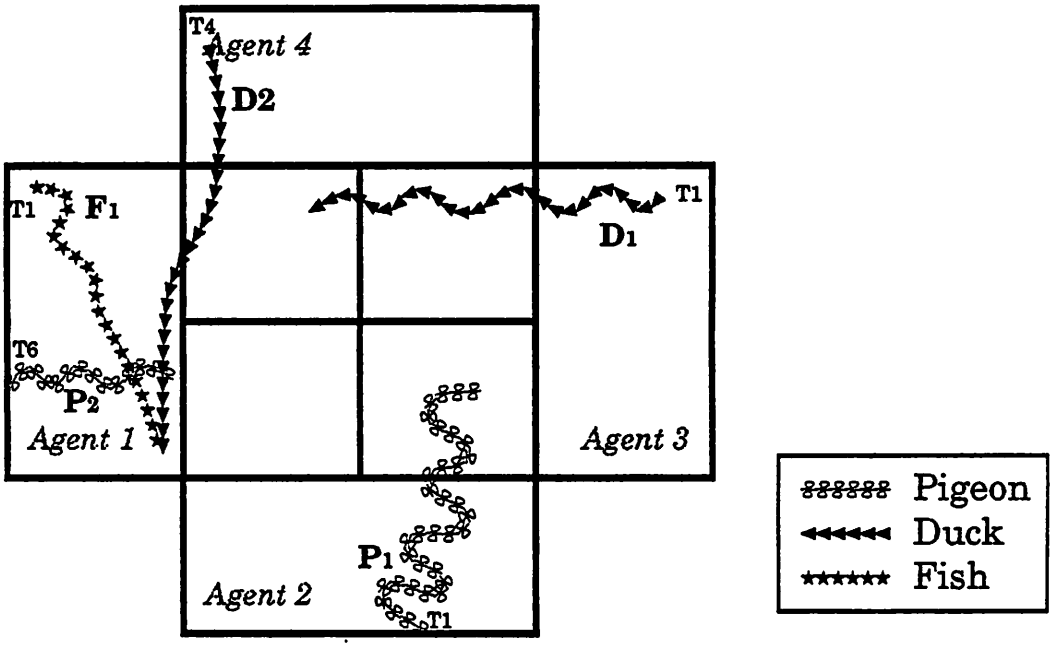

Figure 3: The CDPS scenario environment

4.3 The Intended Interactions

There are four major interactions that are brought about by this environment. A detailed account can be found in Appendix A. These interactions also show up clearly in the goal relationship example in Figure 4, page 17.

1. Agent 4 can help agent 3 disambiguate the data from its faulty sensor by tracking the duck D_1.
2. Agent 4 can notify agent 1 about the impending duck D_2. It might provide other predictive data, for example, it might try to track the duck precisely even though it is a pigeon tracking agent. This can be accomplished through negotiation between agents 1, 3, and 4.
3. Agent 2 can help agent 3 by taking care of the pigeon data in the shared area (thus reducing the load on agent 3 and its faulty sensor).
4. Agent 1 can notify agents 4 and 2 about the impending pigeon, providing predictive information (but note that agent 1 will be under severe time pressure).

4.4 Revisiting the important issues

The scenario above expresses the issues that were listed earlier:

Heterogeneous Agents: Having two different system goals cause the agents' local views of "what is important to do next" to change. Given the same data, Agent 1 and Agent 4 would treat it differently, in isolation, because they have different local decision functions (because Agent 1 is

a fish-protector and Agent 4 is a pigeon-tracker). If it were not for the communication of the goals of Agent 1 to track ducks, Agent 4 might pay little attention to the duck it detects. The attention and effort that Agent 4 puts into the duck must come from some pre-specified or dynamically constructed global decision function that describes how Agent 4 should cooperate with Agent 1. Hence the scenario forces us to deal with the issues of different local views, of exchanging views, and of dynamically constructing global views. See also *Negotiating Agents* below.

Dynamic Agents: The agents have a set of adaptable methods for reaching their goals, each of which have different characteristics. Agent 3 will be using methods very different from those used by agent 1, even though both agents have the same system goal. An agent such as agent 4 might change the processing method on some data (say duck D_1) when new data (duck D_2) appears. The overloaded Agent 1 is also likely to switch strategies as it becomes overloaded. Hence the scenario allows us the opportunity to use agents with various problem solving methods and agents that change strategies during problem solving (with the commensurate difficulties in coordination).

Negotiating Agents: Agents 1 and 3, for example, may have different initial ideas of what agent 4 should be processing (given agent 4 has nothing to do on its own), and certainly different from that of agent 4 itself. Agent 4 must develop some partially global view in relation to Agents 1 and 3 and all of their local goals. Agents 2 and 3 also have data in their overlapping areas. The scenario presents the opportunity to dynamically modify the global view of coordination, changing parameters to fit the situations that develop between Agents 4 and 3, then 4, 1 and 3, and finally 2 and 3. The scenario also presents conflicts between local views and the developed global views — see *Real-time*, below.

Real-time: Both of the system goals require solutions by a deadline. Agent 1 must forego helping agent 4 when pigeon P_2 appears — it is too busy. If a pigeon pops up in agent 2's sensed region, it may not be able to keep its commitments to agents 1 and 3. Thus the scenario provides goals with deadlines and conflicts between local and global views.

5 Approach

The approach we will take is a refinement and extension of that used in the PGP mechanism. The basic mechanism remains the exchange of information that allows each agent to independently affirm (in the case of anticipated domain relationships or "settled questions" [Gasser *et al.*, 1989]) or discover (in the case of unanticipated "open system" interactions [Hewitt, 1986]) its relationship to other agents in the system.

Rather than exchange node plans, we exchange agent goals of various types and levels of detail. By detecting the relationships between its own goals and the goals of other agents it may interact with, an agent may locally schedule

actions while taking into account non-local goals. When domain problem solving requires multi-agent interactions, node plans may still be exchanged, as well as agent *capabilities*, which describe an agent's potential long-term goals. By using hierarchical goal structures and not always exchanging plans we can reduce the amount of communication required and focus what communication does take place to reduce the number of global views developed under the PGP mechanism — we can produce (partially global) schedules rather than partial (global schedules).

5.1 Goals

In extending the PGP mechanism, a major difficulty we encounter is in how goals should be specified for the agents. Most AI programs represent goals as either satisfiable logical formulae or *ad hoc* symbols. How can agents understand each others' goals, and to what extent do they need to?

We require the ability to recognize goal relationships, and the ability to recognize to what degree a goal has been satisfied. These two abilities are sufficient for "scheduling" coordination (reordering tasks locally), but not for multi-agent domain problem solving (where we may exchange or share tasks between nodes). For the latter, goals (and plans) must be recognizable (able to be acted upon) by the underlying agent problem solving structure, or the coordination mechanism must be able to translate them as such.

In the DVMT the agents do have a shared language for domain goals, and a language for control goals is under development. The new coordination mechanism, in order to remain aloof from the specifics of the DVMT, should not take too much advantage of the shared language. Rather, it should rely on the detection of goal relationships and degree of satisfiability directly, where these will be easy to implement because of the existence of a shared goal language. The highest goal specification for an agent is called a *system goal*.

5.1.1 System Goals

A *system goal* is a high-level goal describing the desired solution characteristics for an agents' problem solution. It includes:

Completeness: What parts of a full solution are the most important? To what degree does a partial solution fulfill a goal? In the DVMT, completeness specifies the level at which a hypothesis is a solution, its length, and its event classes.

Precision: How specific must be the data contained in the solution? DVMT hypotheses have a well-defined precision measure.

Certainty: How certain must an agent be that the hypothesis it is putting forth is a solution? Since DVMT belief uses a 4-tuple, this is a bit more complex than it sounds.

Deadline: By what time must the solution be produced?

These desired solution characteristics must be considered along with their interactions — to allow agents to trade off characteristics in a controllable manner. For example, is a solution with twice the precision and half the certainty as good as one half as precise and twice as certain? A simple way to specify this is to use an evaluation function to rate the solution, but this may not allow explicit reasoning about tradeoffs between solution characteristics.

5.2 Goal Relationships

Four classes of goal relationships have been identified that will be useful; a small subset of these relationships have been used to improve the scheduling of tasks in a single agent blackboard system where tasks can be executed in parallel [Decker *et al.*, 1990a]:

Domain Relations: This set of relations is generic in that they apply to multiple domains, and domain dependent in the sense that they can be evaluated only with respect to a particular domain — inhibits, cancels, constrains, predicts, causes, enables, and supergoal/subgoal (from which many useful graph relations can be computed). These relations provide *task ordering* constraints, represented by temporal relations on the goals (see below).

Graph Relations: Some generic goal relations can be derived from the supergoal/subgoal graphical structure of goals and subgoals, for example, overlaps, necessary, sufficient, extends, subsumes, competes. The *competes* relation is used to produce *task invalidation* constraints.

Temporal Relations: These depend on the timing of goals — their start and finish times, estimates of these, and real and estimated durations. From Allen [Allen, 1984], these include before, equal, meets, overlaps, during, starts, finishes, and their inverses.

Non-computational Resource Constraints: A final type of relation is the use of physical, non-computational resources. This is the major relation in some domains, such as factory scheduling and office automation [Sadeh and Fox, 1989, Martial, 1989].

5.2.1 Domain Dependent Relations

This set of relations are *generic*, in that they apply to multiple domains, but *domain dependent* in that they can be evaluated only with respect to a particular domain.

supergoal/subgoal: Goal B is a *subgoal* of goal A if B is required for some method of achieving A.

inhibits: Goal A *inhibits* goal B if when goal A is accomplished goal B cannot be accomplished. This definition ignores the time component. A might either *permanently* or *temporarily* inhibit B.

cancels: Goal A *cancels* B if when goal A is achieved, goal B is *no longer achieved*. Notice that this is subtly different from A *inhibiting* B; Inhibition implies that B cannot be accomplished, canceling implies B is no longer achieved, but not that it cannot be achieved.

If B *cancels* B then B is a *recurring* goal, one that undoes itself. Of course any set of goals may be placed in a recurring relationship if they cancel each other in sequence.

constrains: Goal A *constrains* goal B if the two goals are related somehow at the domain level by the need to exchange information from A to B in order to solve B. This may or may not be a time constraint. This is necessary information. If A constrains B then A [>,m,o,oi,d,di] B (A is before, meets, overlaps, is overlapped by, is during, or surrounds B [Allen, 1984]). Note that the "constraint relation" does not tell you what the constraint is, but that one exists. It means that the two subgoals are interacting subproblems.

predicts: Goal A *predicts* goal B if information about the solution of A is useful for the solution of B but not necessary. This could be information used to reduce uncertainty, or to guide search.

causes: Goal A *causes* B if the completion of goal A physically entails the occurrence/completion of B.

enables: Goal A *enables* B if the completion of goal A must occur before goal B can be satisfied. Obviously this implies a strong temporal constraint.

5.2.2 Graph Relations

The generic graph relations can be derived simply from the subgoal/supergoal structure of the goal hierarchy, without using any internal information about a goal or any domain-dependent information. We can view the goal hierarchy as an acyclic AND/OR graph. Two nodes are *equivalent* if they have *equivalent* subgoals, or consist of identical primitive actions. We may also want to assume algorithmically that equivalent nodes are only represented once in the goal tree (which will then be a goal acyclic graph).

Just because a goal relation can be derived from a graph does not mean that it is trivial; when adding a new goal node to the graph a great deal of computation may need to go on to see how that goal really relates to the others. Part of this computation comes in recognizing repeated goals (that occur multiple times in the graph), so that you cannot blindly expand a goal into a set of subgoals.

overlap: Goal A *overlaps* B if there exists a goal G such that A is a supergoal of G and B is a supergoal of G.

necessary: Goal B is *necessary* for goal A if goal A cannot be accomplished without accomplishing goal B (B is an AND subgoal in an AND/OR tree).

sufficient: Goal B is *sufficient* for goal A if accomplishing goal B accomplishes goal A (goal B is either an OR subgoal or A, or is sufficient for an OR subgoal of A).

extends: Goal A *extends* B if there exists a *supergoal* G such that A and B are both *necessary* for G.

subsume: Goal A *subsumes* B if B is a *subgoal* of A or if B is *obviated* by A. B is *obviated* by A if A is *sufficient* for the *parent* of B.

competing: A *competes* with B if A and B are *n-competing* for some n. Inductively, A and B are *0-competing* if A and B are in different disjuncts for each of their parents. A and B are *n-competing* if every pair of parents of A and B are *i-competing* for $0 \leq i \leq n-1$. This captures the intuitive idea that two goals *compete* if there is no possible way that both goals *must* be fulfilled. This does not mean that they cannot both be fulfilled, or that they interfere with one another in any way. The concept is graph-theoretic, not domain dependent.

5.2.3 Time Based Relations

A third set of relations are not strictly domain dependent, but do depend on the timing of goals. There are several possible features of goals that are applicable to timing: actual start and finish times, estimated start and finish, deadlines, and (estimated, actual) lengths. Any one of these numbers alone will allow at least some limitation of the possible temporal relations between two goals. Furthermore, temporal relations may be preferences (soft constraints) as well as absolute relations. We envision three levels of temporal constraints:

Hard: Hard constraints cannot be violated. Inability to satisfy hard constraints means that the problem is overconstrained. Overconstrained problems may result in negotiation, for example, in real-time problems where a node may have an alternate solution path that takes less time but is also less certain (approximate processing).

Negotiable: Negotiable constraints are preferences that a local node does not want violated needlessly. Inability to satisfy a negotiable constraint means that the constraining node must be part of the decision to modify the constraint.

Soft: Soft constraints are preferences that a node has but do not require negotiation in order to violate. For example, local orderings of intermediate goals in the DVMT are soft constraints.

Temporal relations have been studied before, and very precise definitions have been put forward by James Allen [Allen, 1984]. We give the graphical suggestion of a definition as presented by Allen in lieu of the formal definitions to be found in Allen's papers. Remember that each relation has an inverse and that in the presence of limited information a set of these relations may hold between any two timed goals:

before: $\frac{xxxx56789}{12345yyyy}$

temporally equal: $\frac{123xxxx89}{123yyyy89}$

meets: $\frac{12xxxx7890}{123456yyyy}$

overlaps: $\frac{123xxxx890}{12345yyyy0}$

during: $\frac{123xxxx890}{1234yy7890}$

starts: $\frac{123xxx789}{123yyyyyy}$

finishes: $\frac{12345xxx90}{12yyyyyy90}$

5.2.4 Using Goals

The new DVMT control architecture [Decker *et al.*, 1990b] uses the specification of the system goal(s) to choose a strategy for satisfying the goal. The strategy posts a set of goals whose specification in turn allows the choice of an appropriate substrategy, etc. At some point a goal can trigger the creation of a *focus* (the lowest leaf in an expanded control plan) which specifies a set of low-level control parameters and rating heuristics. These focus parameters and heuristics are associated with a *channel* in the low-level domain problem solving system, existing concurrently with other active foci and their associated channels.

It is this hierarchical set of control goals, starting with the system goal, that are communicated in our approach. Goal relationships detected between local goals and the goals received from other agents allow us to coordinate scheduling. Figure 4 shows all four agents in the scenario, and all of their control goals. Note that not all of these goals are active simultaneously, in particular, the *identify* goal for a hypothetical vehicle always comes strictly *before* the *tracking* goal for the newly identified vehicle. The example in Section 5.4 shows how Figure 4 is constructed for Agents 1 and 4.

5.3 Coordination Rules

The last item that we need to specify in our approach is the PGP algorithm itself. Initially both a PGP-like scheduling algorithm and the organizational information needed to use it will be encoded as a special set of control knowledge sources that handle when to send and receive goals and hypotheses and how to modify the local schedule given this information.

Some of these rules are generic (like "don't do redundant work") but as in the original PGP mechanism, they must be operationalized for the particular domain. Each rule should also be considered a "temporarily settled question," subject to being reopened in the domain setting [Gasser *et al.*, 1989].

The PGP algorithm orders intermediate goals according to their cost as computed from the cross-product of a vector of computable factors (such as redundancy, reliability, etc.) and global cooperation parameters that give a weight to the corresponding term in the calculation. These factors were the following:

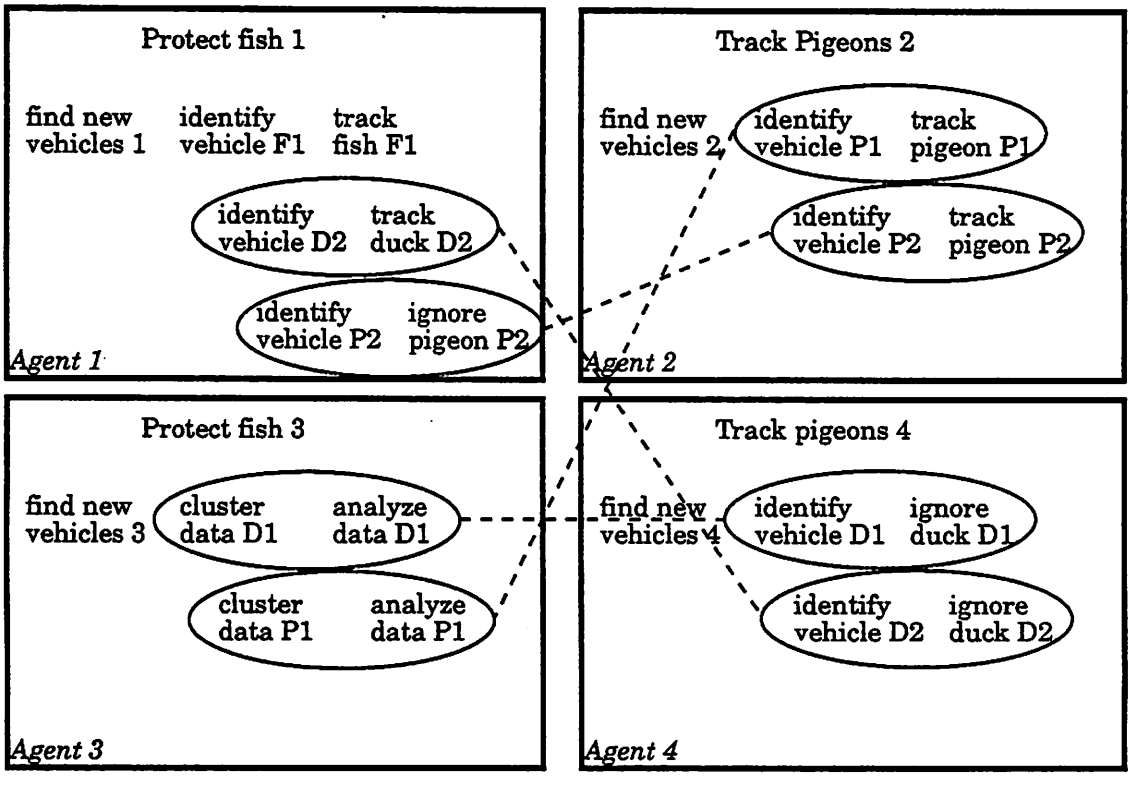

Figure 4: Goal *overlaps* relationships (dotted lines) between agents in the CDPS scenario

Redundancy: The number of nodes that can perform this goal. The GPGP scheduler can use the equality and subgoal relationships to examine redundancy.

Reliability: The number of nodes that cannot perform this goal. This is the inverse of redundancy.

Duration: The duration of the goal. This measure can be used by the GPGP scheduler as well.

Predictiveness: Any goal with a duration of x can provide predictive information for a goal of duration y if $x < y$. The predictiveness measure was then the minimum activity distance (minimum number of sensed times) between the two goals. GPGP considers the predictiveness relation to be a domain-dependent relation.

Locally-predicted: The minimum activity distance between a goal and any goal to be executed before it. The effect of this measure was to keep a node from jumping around between times in constructing a track; extending an existing partial solution was preferred. The GPGP scheduler can use the hierarchical construction of the goal hierarchy to avoid reordering the steps in constructing a track.

Independence: how many goals occur before a given goal in the initial local node plan. This measure was intended to keep the PGPlanner from straying too far from the original local node plan ordering. The independence measure for each goal is constant, because it depends only on the initial local node plan, not on the position of the goal in the re-ordered plan. Goals later in the initial ordering have higher independence measures. Because the PGP algorithm used a swapping procedure to create a new ordering from the old, the higher independence measure made later goals harder to swap. The GPGP scheduler receives local ordering preferences, and so it knows what local orderings were necessary, which may be negotiated, and which are only preferences. Thus this relationship is not directly needed.

Diversity: The diversity value of a goal is 0 if it does not derive redundant information, or if all goals following it derive redundant information. Otherwise, the diversity of a goal is measured by the minimum activity distance between the goal and the later non-redundant goals. The effect is to plan to do non-redundant work before redundant work, and a GPGP scheduler can detect redundancy through goal relationships.

General coordination rules:

1. Broadcast system goals (establish long term relationships among nodes).
2. Broadcast capabilities (establish basic node capabilities).
3. Send any goal that *overlaps* another agent's goal to that agent (notify nodes of potential interactions based on perceived roles).

4. Send hyps that satisfy (partially satisfy?) another agent's goals to that agent.

PGP-like rules:

1. Avoid useless redundancy with other nodes.
2. Build reliable solutions. It is OK to be redundant in order to increase certainty in a solution. Two nodes with the same system goal might both analyze the data in an overlapping area, whereas agents with different system goals might let one or the other handle it. The PGP mechanism never really did this.
3. Minimize durations. In the case where more than one agent has an equivalent goal, then let the one with the shorter predicted duration do it.
4. Provide predictive information to other nodes.
5. Perform regular problem solving (follow the local control plan).

The following PGP relations are handled by the local control plan: keep local order if possible (independence), extend tracks in time order (locally-predicted), avoid intra-node redundancy (diversity). These deal with reordering local plans — but we are not constructing plans, but merely exchanging goals.

Finally, there is still the question of how to integrate the goals of other agents into an agent's local problem solving; when to accept goals that will cause work at the local node, and when to split goals to allow them to be shared between agents. We are pursuing these questions using the PGP mechanisms as an initial starting point.

5.4 Example 1

These coordination rules will only go part way toward our goals, but will serve for an illustrative example. This brief example shows Agent 4 providing predictive information to Agent 1 (a track hypothesis for duck D_2).

The four agents have the same domain knowledge and the same meta-control knowledge. This meta-control knowledge consists of two basic strategies: a *goal-directed strategy* and a *clustering strategy*. The goal-directed strategy consists of substrategies to find new vehicles, identify vehicles, track a vehicle, and 'ignore' a vehicle. The clustering strategy consists of substrategies for finding new areas, clustering data, tracking, and analyzing the tracks. All of the substrategies have one or more foci that implement them, and these foci may differ by the amount of time they take and the precision or certainty of their result. For example, *find-new-vehicles* can use a threshold on signal strength to ignore weakly sensed vehicles, *identify-vehicle* can do more or less work in making an identification, and so on.

First all agents broadcast their system goals and capabilities to the other agents. Each agent will begin in the default (goal-directed) strategy to satisfy its system goal, and will post a *find-new-vehicles* goal. Since *find-new-vehicles*

can potentially output any type of vehicle, this goal overlaps the system goals of the other agents (indicating the potential for coordinated scheduling) and is broadcast to them. The agents also recognize that the *find-new-vehicles* goals overlap in the shared sensed areas. If this goal were made up of smaller ones, then the smaller ones would be exchanged, but in this case this goal is the lowest level. Actually reasoning about and splitting goals that overlap like this is an area for future work.

For this example we will just look at the interaction of Agents 1 and 4.

Agent 1 soon detects a vehicle (F_1) and starts a new goal to identify it. While Agent 1 has not identified the vehicle yet, it does have enough information to predict that the vehicle is not a pigeon (the fish signals do not overlap very much with bird sounds) and so it does not transmit the identification goal to Agent 4 (it could change its mind about this later). The *find-new-vehicles* goal is not retransmitted and it remains active.

When Agent 1 identifies the fish F_1, it begins tracking it. The tracking goal is not transmitted.

Later (time 4), while Agent 1 is tracking the fish, Agent 4 detects a new vehicle (D_2). When Agent 4 starts up a new goal to identify this vehicle, it sends the goal to Agent 1, for the goal is potentially a duck, which Agent 1 is interested in. At this point Agent 1 can only believe that Agent 4 has detected something that may be a duck, and it knows where it is and how strongly Agent 4 believes that it is a duck.

While Agent 4 is identifying the duck, the duck crosses into the shared sensor area of agents 1 and 4. Agent 1 detects the data as a new vehicle, and creates a goal to identify it. This goal is transmitted to Agent 4, since it could be a pigeon. Agent 4 realizes that Agent 1's *identification* goal overlaps its own as it specifies an area coincident with the identification track being developed at Agent 4. Thus redundant work is occurring. In this case having both agents work on the data should increase certainty in their conclusions, because neither agent is using a provably dominated algorithm (in fact, the algorithms are the same, the data arise from independent sensor readings). Thus the identification goals become equivalent, and this recognition is communicated to Agent 1. Agent 4 will complete its identification before Agent 1, because it started earlier.

When Agent 4 completes its identification, the resulting hypothesis (that there is a duck on a track extending from times 4 – 7) is transmitted to Agent 1. This hypothesis satisfies Agent 1's *identification* goal.

6 Current Work

We are currently building the initial implementation of the approach above, including the scenario. About half of the scenario is complete and running without communication between nodes. Our first set of experiments will be to develop a set of baseline measurements on this system without communication. The measurements will include:

- Measuring how well the system goals are satisfied. Each system goal has a quantified measurement (see Appendix B). Each part of the system goal

(completeness, precision, certainty, deadline) can be measured separately.

- Measuring the 'cycles' used at each node. In the new DVMT, where data appears during processing, simply measuring the time when processing is completed is no longer appropriate. What is important is how how much of the available processing resources are taken up over a given time slice. The relationship between processing cycles and time passing in the simulated real-world can be controlled. Initially, we will set this parameter so that Agent 1 in the baseline system cannot process all three vehicles (the fish F_1, duck D_2, and pigeon P_2) that are present after time 6 without causing the real-time subsystem to react in some way (the current version of the real-time subsystem drops work on the pigeon until the duck track is built and the fish is notified).

After adding the PGP coordination heuristics to the control level, the same measurements can be taken. We expect better goal satisfaction measurements but higher utilization of available processing resources. Further experiments can then be conducted to examine the following:

- The effect of tighter or looser deadlines. This can be achieved by modifying the ratio of processing cycles to simulation time.
- The effect of more or less nodes. The performance of a given number of agents must also be indexed by the average number of agents that need to interact during the expanded scenario.

All of this work is related to the task of characterizing coordination, and how the PGP fits into that concept.

7 Conclusion

We have presented an approach to extending the partial global planning mechanisms for coordination that relies on the detection and representation of goal relationships. This extends the partial global planning approach to deal with heterogeneous agents, agents with dynamic problem solving abilities, agents with hard real-time deadlines, and that can participate in more sophisticated negotiation behaviors. We will evaluate our approach experimentally, using the scenario detailed here.

In the future we will provide details of our implementation, and we will have a testbed with which to experiment with other coordination algorithms, with negotiation algorithms, and with the interaction of our coordination mechanisms and the new hard real-time problem solving mechanisms we are also building.

References

[Allen, 1984] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[Davis and Smith, 1983] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, January 1983.

[Dean, 1987] Thomas Dean. Planning, execution, and control. In *Proceedings of the DARPA Knowledge-based Planning Workshop*, December 1987.

[Decker *et al.*, 1990a] Keith S. Decker, Alan J. Garvey, Marty A. Humphrey, and Victor R. Lesser. Effects of parallelism on blackboard system scheduling. In *Proceedings of the Fourth Annual AAAI Workshop on Blackboard Systems*, Boston, August 1990.

[Decker *et al.*, 1990b] Keith S. Decker, Victor R. Lesser, and Robert C. Whitehair. Extending a blackboard architecture for approximate processing. *The Journal of Real-Time Systems*, 2(1/2):47–79, 1990. Also COINS TR-89-115.

[Durfee and Lesser, 1987a] Edmund H. Durfee and Victor R. Lesser. Planning coordinated actions in dynamic domains. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, pages 18.1–18.10, December 1987. Also COINS-TR-87-130.

[Durfee and Lesser, 1987b] Edmund H. Durfee and Victor R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, August 1987.

[Durfee and Lesser, 1988] Edmund H. Durfee and Victor R. Lesser. Incremental planning to control a time-constrained, blackboard-based problem solver. *IEEE Transactions on Aerospace and Electronic Systems*, 24(5), September 1988.

[Durfee and Lesser, 1989] Edmund H. Durfee and Victor R. Lesser. Negotiating task decomposition and allocation using partial global planning. In M. N. Huhns and L. Gasser, editors, *Distributed Artificial Intelligence, Vol. II*. Pitman Publishing Ltd., 1989.

[Gasser *et al.*, 1989] Les Gasser, N. F. Rouquette, R. W. Hill, and J. Lieb. Representing and using organizational knowledge in distributed AI systems. In M. N. Huhns and L. Gasser, editors, *Distributed Artificial Intelligence, Vol. II*. Pitman Publishing Ltd., 1989.

[Hewitt, 1986] Carl Hewitt. Offices are open systems. *ACM Transactions on Office Information Systems*, 4(3):271–287, July 1986.

[Kuwabara and Lesser, 1989] K. Kuwabara and V. R. Lesser. Extended protocol for multi-stage negotiation. In *Proceedings of the Ninth Workshop on Distributed AI*, September 1989.

[Lander and Lesser, 1989] Susan Lander and Victor R. Lesser. A framework for the integration of cooperative knowledge-based systems. In *Proceedings of the 4th IEEE International Symposium on Intelligent Control*, pages 472–477, September 1989.

[Lesser *et al.*, 1988] Victor R. Lesser, Jasmina Pavlin, and Edmund Durfee. Approximate processing in real-time problem solving. *AI Magazine*, 9(1):49–61, Spring 1988.

[Martial, 1989] Frank V. Martial. Multiagent plan relationships. In *Proceedings of the Ninth Workshop on Distributed AI*, September 1989.

[Robinson and Fickas, 1990] William N. Robinson and Stephen Fickas. Negotiation freedoms for requirements engineering. Technical Report CIS-TR-90-04, Department of Computer and Information Science, University of Oregon, April 1990.

[Sadeh and Fox, 1989] N. Sadeh and M. S. Fox. Preference propagation in temporal/capacity constraint graphs. Technical report CMU-RI-TR-89-2, Robotics Institute, Carnegie Mellon University, January 1989.

[Smith and Broadwell, 1987] David Smith and Martin Broadwell. Plan coordination in support of expert systems integration. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, pages 12.1–12.6, December 1987.

A Detailed Account of the Scenario Interactions

The account of processing below includes all of the choice points for the agents, and describes the criteria that the agent must use to make a decision at that point. It may be helpful to refer back to Figure 3 on page 10.

A.1 Agent 1

Time 1: The fish signals are received, and agent 1 goes about processing them to determine that there is, in fact, a fish present.

Times 2–4: Agent 1 continues to track the fish, while simultaneously scanning for ducks and other fish. By projecting the fish track, it decides that the fish is not heading towards agent 3 and so it does not notify agent 3 about the fish.

After Time 4: Agent 4 will detect the duck D_2 at time 4. As soon as it believes that it is a duck, it should transmit this data to agent 1, as it is not doing anything for its own system goals and it is very important for agent 1. At this point, agent 1 should prepare for the possibility that the duck is attacking the fish (it can already estimate a deadline for fish notification). Because Agent 4 is also helping Agent 3 at this point, Agent 1 and Agent 4 may enter into a negotiation about global responsibilities that may leave Agent 1 with the sole responsibility for tracking the duck D_2 within the overlapping sensed region between Agents 1 and 4.

Time 6: Agent 1 senses the pigeon P_2. It will have to process it somewhat to know that it *is* a pigeon. It should send this information off to agents 2 and 4. However, it will already be under the predicted threat of severe time pressure tracking duck D_2. The duck D_2 will also be initially sensed directly

by agent 1 at this time. Agent 1 has run into a conflict between the global goals of cooperation and its local goals of protecting the fish within the deadline.

A.2 Agent 2

Time 1: The pigeon signals (P_1) are received, and agent 2 goes about processing them to determine, in fact, that there is a pigeon present.

Time 5: The pigeon P_1 crosses into the shared region of agents 2 and 3. Agent 2 will know that this is a pigeon and not of interest to agent 3 (which has a faulty sensor anyway) and should continue tracking the pigeon. Agent 2 and 3 will develop a partially global view where Agent 2 will process the data in their shared overlapping sensed area.

Time 8: The pigeon P_2 arrives. Agent 2 may or may not have been warned by Agent 1.

A.3 Agent 3

Time 1: The duck D_1 appears, but Agent 3, with its noisy sensor, will not be able to tell this immediately. Agent 3 will have to use an alternate processing strategy to deal with the noise.

Time 3: The duck D_1 crosses into the shared area of agents 3 and 4. Agent 4 should be able to tell quickly that the signal is caused by a duck; this information should allow agent 3 to focus its processing better — it will also notify agent 1 of the presence of a nearby duck. Agents 3 and 4 need to arrive at a partial global view of their roles in tracking this duck (D_1); because of the noisy sensor at Agent 3 they may decide to increase reliability and both work in the sensed area.

A.4 Agent 4

Time 1: No data, no processing.

Time 3: Duck data from duck D_1 appears — agent 3 may have requested agent 4's help in determining the vehicle type as discussed above.

Time 4: More duck data (D_2) arrives. Agent 4 must balance helping agent 3, processing the new data (in case it is a pigeon) and also working with agent 1 on the new data once it is discovered that it is a duck. Agent 4 must develop a partial global view of the relationships at Agents 1, 4, and 3.

B Example System Goals

B.1 System Goal 1: Fish Protection

The first system goal is to protect fish from ducks. To do this, a node must find the fish and ducks and notify a fish about a potential duck attack in time for the fish to respond and escape.

Completeness: Solution must be a pattern track hypothesis; it must be at least 4 time units long; Only duck-attacking-fish event classes are useful.

A pattern track is composed of related vehicle tracks. A duck vehicle track might be part of the pattern *duck-attacking-fish* (which would also include a fish vehicle track), or it might only be *meandering*.

Precision: The further away the duck is from the fish, the less precise a solution must be to be acceptable. See *deadline* below.

Certainty: A solution with certainty less than 0.5 is unacceptable. Acceptability rises from 0 to 1 linearly with higher certainties, i.e., certainty acceptability can be defined as:

$$\mathrm{CA}(x) = \begin{cases} 0 & \text{if hyp-certainty} < 0.5 \\ 2(\text{hyp-certainty}) - 1 & \text{if hyp-certainty} \geq 0.5 \end{cases}$$

Deadline: If the hyp precision is below 10 (is precise), the fish can escape with as little as 3 time units warning. If the hyp precision is above 10 (is imprecise), then the fish needs at least 5 time units of warning (the hyp precision measure is an inverse measure with a precise point giving a measure of 0). Any warning after these times is of no utility. Any notification before 5 units is also of no use (the fish cannot track the duck itself)

A local utility function can be constructed by multiplying the four above utility measures together. Multiplication in this case simulates an **AND** combination function, since the evaluation will be 0 if any term is 0. Therefore no tradeoff can violate the deadline.

B.2 System Goal 2: Pigeon Tracking

The second system goal is to track pigeons. Agents with this goal must keep a pigeon tracking display updated with the most current pigeon position information.

Completeness: Only pattern tracks of meandering pigeons are important, regardless of length.

Precision: Precision should be below 10.

Certainty: Certainty should be above 0.7.

Deadline: The display should never be more than 2 locations behind the current pigeon positions.

A local utility function can be constructed by multiplying the four above utility measures together. In this case, it has a value of either 1 or 0.