# Generalizing the Partial Global Planning Algorithm *

Keith S. Decker
Victor R. Lesser
Department of Computer Science

University of Massachusetts
Amherst, MA 01003
CSNET: decker@cs.umass.edu, lesser@cs.umass.edu

June 21, 1993

* *To appear in the International Journal of Intelligent Cooperative Information Systems, 1(2), pp. 319–346, 1992.* This work was supported by DARPA contract N00014-92-J-1698, and partly by the Office of Naval Research under a University Research Initiative grant, number N00014-86-K-0764, and NSF contract CDA 8922572. The content of the information does not necessarily reflect the position or the policy of the Government and not official endorsement should be inferred.

**Abstract**

The distributed coordination problem can be described as *how should the local scheduling of activities at each agent be affected by non-local concerns and constraints.* Partial global planning (PGP) is a flexible approach to distributed coordination that allows agents to respond dynamically to their current situation. It is based on detecting relationships in the computational goal structures of the distributed agents. However, the detailed PGP mechanisms depend on the existence and availability of certain characteristics and structures that are idiosyncratic to the Distributed Vehicle Monitoring Testbed (DVMT). Generalized Partial Global Planning tries to extend the PGP approach by communicating more abstract and hierarchically organized information, detecting in a general way the *coordination relationships* that are needed by the partial global planning mechanisms, and separating the process of coordination from local scheduling. This new characterization of partial global planning has less communication overhead and can be more easily adapted and extended to new styles of problem solving and new multi-agent environments that have different characteristics than the original DVMT. This paper first describes the coordination problem as it was viewed by the PGP algorithm, and then extensions to that problem. It then briefly describes our model of task structures and coordination relationships. Finally, we show how the PGP algorithm, as an example, can be described using our method.

# 1 Introduction

The partial global planning (PGP) approach to distributed coordination increased the coordination of agents in the network by avoiding redundant activities, shifting tasks to idle nodes, and providing predictive results[16] — in short, by informing local scheduling of non-local considerations. It grew out of the observation (made while studying functionally accurate, cooperative distributed problem solving (FA/C)[26, 25]) that agents do not have to be totally coherent in their behavior in every situation to make a coordination approach worthwhile. The PGP mechanisms were based on the communication of a common, flat, and intermediate-level goal structure that indicated the near-term ordering, duration, importance, and predicted result of the goals. An agent then used this *partial* information about other agent's goals to construct a partial *global* view of network problem solving, by observing the interactions of all the goals from some subset of the agents (including its own goals). The resulting view was then used by an agent to *plan* or schedule the order that the agent's own goals would be accomplished.

The PGPlanner, as it was used for coordination in the distributed vehicle monitoring task (finding, identifying, and tracking moving vehicles using acoustic sensor signals[27]), 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 [14], but these relationships are not transmitted in the node plans. 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. Each relationship was defined narrowly in reference to the DVMT.

The basis of the PGP is the construction, maintenance, and exchange of the agent goal structures. Generalized Partial Global Planning tries to extend the PGP approach by communicating more abstract and hierarchically organized information, detecting in a general way the coordination relationships that are needed by the partial global planning mechanisms, and separating the process of coordination from local scheduling. Because generalized partial global planning depends only on an idealized model of each agent's activities, and a set of generalized relationships between those activities, it is a domain-independent coordination technique. Domain dependent knowledge is used only to discover or hypothesize the existence of these general relationships. A key point is that coordination relationships are abstractly defined, so we can catalogue coordination strategies and characterize them by potential features of an environment and task.

Regardless of a distributed agent's actual architecture, we can model its activity as a set of task environment structures.This structure is an extended task tree, including hierarchically organized information on task ordering, timing, importance, solution constraints, resource utilization, and expected result characteristics. When an agent receives a portion of another agent's task structure, it records the relationships between the structures. These relationships indicate the possible occurrence of redundant activities, the presence of new constraints, and predictions of future activities and interactions. An important characteristic to note is that some of these coordination relationships are 'soft', in that an agent may ignore them but will then execute tasks less efficiently. Another characteristic is that not all tasks will always

need to be done, or done completely, to effectively solve a given problem. Our model of coordination is based on detecting the specific, quantitative nature of domain-independent relationships between tasks involving interacting subproblems. The original PGP mechanism uses a domain-specific, reduced set of these relationships. Section 4 discusses task environment structures and coordination relationships, and gives examples of how this model can be applied to the Distributed Vehicle Monitoring Testbed.

For an example of this style of coordination (see Figure 1), consider that Agent 1 has an overall goal of accomplishing task $A_1$, and two concurrent subtasks, $B_1$ and $C_1$. Agent 2 has an overall goal of accomplishing task $A_2$, and two subtasks $B_2$ and $D_2$. In the generalized partial global planning algorithm, Agents 1 and 2 will exchange their top-level goals (called "system goals"). Assume that this information is enough to allow Agent 1 to determine that local subtask $C_1$ *facilitates* $A_2$. *Facilitates* is one of the generalized coordination relationships we discuss later. It indicates that achieving $C_1$ is somehow useful for the achievement of $A_2$. Information about subtask $C_1$ is sent to Agent 2; the result of achieving $C_1$ will also be sent when it is available. The *facilitative* nature of the remote subtask allows agent 2 the chance to rearrange its schedule to handle future information. Similarly, the fact that $C_1$ is facilitative for Agent 2 causes the task to become more preferred by Agent 1, subject to any local constraints. For example, if Agent 1 had two competing subtasks, one of which was $C_1$, and no other reason to prefer one over the other, the PGP coordination algorithm would cause Agent 1 to prefer to schedule $C_1$ at an earlier time than its competitor.

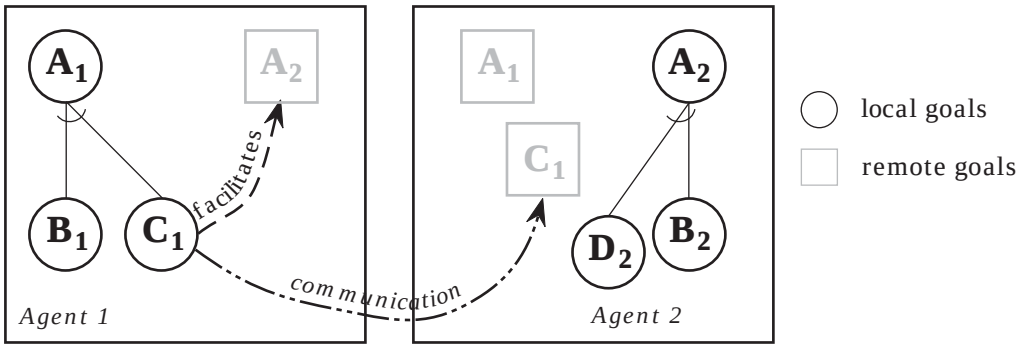


Figure 1: An example of agent-level task environment structures and coordination relationships

The task environment structure was not designed only to permit the communication of information needed for partial global planning, but also for the design and analysis of coordination algorithms for agents operating in environments with characteristics different from those that the PGP was designed for. We are interested in extensions that include support for heterogeneous agents (which may have different local problem solving criteria), dynamic agents (which have several different strategies and methods available for accomplishing goals, and a set of tradeoffs among them), and real-time agents (which may have hard or soft deadlines). Generalized PGP also supports (but does not prescribe) explicit negotiation among agents, and allows agents to communicate at different levels of detail. The goal structures constructed, maintained, and communicated by the original PGP mechanism are subsumed by this model.

For example, in the office environment, both independent computerized agents (usually controlling access to resources) and intelligent assistants to office workers can exist[30]. In real

offices, activity is taking place semi-autonomously, and centralized control and detailed global views are rarely available or socially possible [21, 31]. In the intelligent office assistant domain, then, generalized PGP can be applied to providing guidance to an office worker about how to prioritize tasks, given discovered coordination relationships between one worker's goals and the goals of others [31][1] Coordination algorithms do not supply a solution for problems of negotiating outcomes or resolving disparate views, but rather try to avoid negative interactions between agent goals or planned actions (avoiding conflicting actions or inter-agent action sequences), and recognize positive or helpful interactions (such as the potential to do another agent a favor, or send some preliminary, facilitative information)[38]. Often the coordination process triggers a process of negotiation.

A distributed coordination algorithm specifies what information is to be communicated, when it is to be communicated, and how that information affects local task scheduling. This approach views the coordination mechanism as *modulating* local control, not supplanting it—a two level distributed search process that makes a clear distinction between coordination behavior and local scheduling[6, 25] (See Section 4 and Figure 3). By concentrating on the creation of local scheduling constraints, we avoid the sequentiality of scheduling in partial global planning that occurs when there are multiple plans. By separating coordination from local scheduling, we can also take advantage of advances in complex scheduling in general, and in real-time scheduling algorithms in particular, to produce CDPS systems that respond to real-time deadlines. By designing a coordination algorithm such as the PGP algorithm using domain-independent coordination relationships we also gain a significant separation of the coordination process from the details of the environment (domain). Our long-range goal is to develop a set of domain-independent coordination strategies (of which the generalized PGP is one). We expect that different strategies will be useful depending on the characteristics of the problem being solved and the domain or environment involved. Thus we are also developing a model of how the characteristics of the environment and task effect the coordination behavior of such a strategy[11]. For example, the DVMT environment is characterized by the presence of many overlapping and facilitative goals, while computer-supported design environments are overwhelmed by constraint relationships, and office environments by enablement and non-computational (physical) resource constraints.

This paper first describes the coordination problem as it was addressed by the original PGP mechanisms. Then it describes other issues that point out the necessity of extending the PGP view. Third, it describes our model of agent activity, which defines a set of coordination relationships that can be used to build domain-independent distributed coordination strategies. Section 5 describes the design and implementation of a particular example of a domain-independent coordination strategy, the generalized partial global planning mechanism, followed by examples of its use in a greatly modified DVMT.

# 2 Partial Global Planning

Partial global planning [16] was developed as a distributed control technique to insure coherent network problem solving behavior. It is a flexible approach to coordination that does not

[1] Some computer supported cooperative work (CSCW) research has concentrated on the act of the intelligent assistant discovering the goals of the human worker[3].

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[16]:

- 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.

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 generating interpretations of the same region in time and space. This occurs in the DVMT because, in the interest 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 facilitative 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 $G$ [15].

- 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).

- It is terminated externally when it first reaches the correct solution, rather than through some internal termination criteria.

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 [13]. The PGP framework does not deal with conflicts in physical resources.

# 3 Issues in Extending the PGP Mechanisms

The global coherence problems we would like to address occur in many systems other than the DVMT, such as the Pilot's Associate (PA) system [36], where situations occur that cause potentially complex and dynamically changing coordination relationships to appear between 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 2). 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.

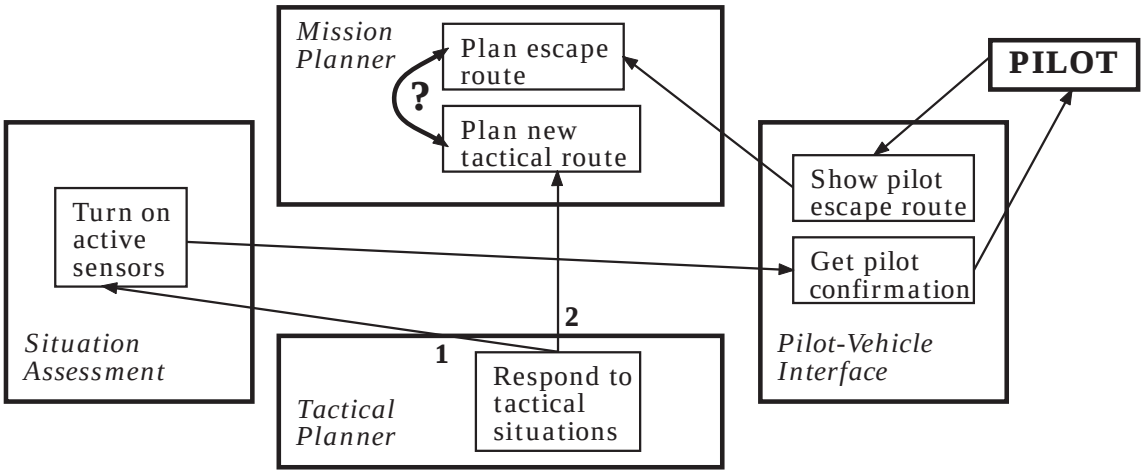


Figure 2: Dynamic Situations in Pilot's Associate

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. We will return to this example several times in this section.

## 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 [24], pilot's associate-style problems [36]). The PA scenario in Section 3 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 [14] 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 relationships as *hard*, *negotiable*, and *soft*. This allows the local scheduler to rule out certain impossible orderings (hard constraints), and to avoid those that may cause replanning at the target node (negotiable constraints). More importantly, we advocate a separation of coordination mechanisms and local scheduling mechanisms, so that nodes do not work out complete schedules for other nodes, but rather pass only scheduling 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 or anytime algorithm techniques [12, 2]. In the PA scenario in Section 3, 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 and potentially wasteful in more complex domains[23]; certainly the detection of the interactions of two goals ("partial global goals") will not always be simple [33]. 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 [13]. The node plan structure could also be expanded with contingency plans for "routine expectation failures" [7] to allow for predictability in the face of a changing environment. Agents can also make commitments to certain goal characteristics, and add explicit slack to schedules[11].

### 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 task characteristics) *and* the goal deadlines themselves[8, 17]. In the PA scenario in Section 3, 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 [28, 12].

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, and is one reason why we now advocate the separation of coordination and local scheduling. Our initial experiments with a real-time local scheduler[11] show that a large part of distributed real-time performance emerges from sophisticated local real-time scheduling capabilities[17].

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[8, 17].

### 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 3 occurs in too short a time-frame to allow negotiation between the agents, but other PA scenarios might profitably use negotiation techniques[2].

The PGP mechanism uses a shared global plan evaluation function that is parameterized. One extension is to allow the parameters (such as *redundancy* 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 [5] 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. Negotiation can interact with sophisticated real-time local schedulers that have the capability to analyze potential future schedules.

In order to examine all of the above issues more closely, a scenario is described in Section 6 which exhibits the issues mentioned above.

## 4 Modeling Task Environments

Our approach to distributed task coordination rests on a conceptual, causal model of the generalized PGP-style distributed coordination process (see Figure 3). Information flows from the environment through the agent's coordination mechanism and local scheduling mechanism, eventually influencing performance. First, a given environment and/or task domain, in conjunction with an agent architecture, induces certain coordination relationships (CRs) between tasks in that environment. An agent follows some *coordination algorithm* that detects or hypothesizes CRs and reacts accordingly—the algorithm produces certain behaviors, for instance the creation and refinement of local scheduling constraints (other possible behaviors include communication, negotiation, and the creation of internal data structures). We can relate how the behavior of the coordination algorithm (creating and refining constraints) affects the agent's scheduling behavior by basing our analysis on certain properties of the local scheduling mechanism. Finally, the scheduling behavior affects the performance of the agent and any organization of which it is a part not only by ordering and executing tasks but also through incurring costs, such as communication and time.

[2]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.

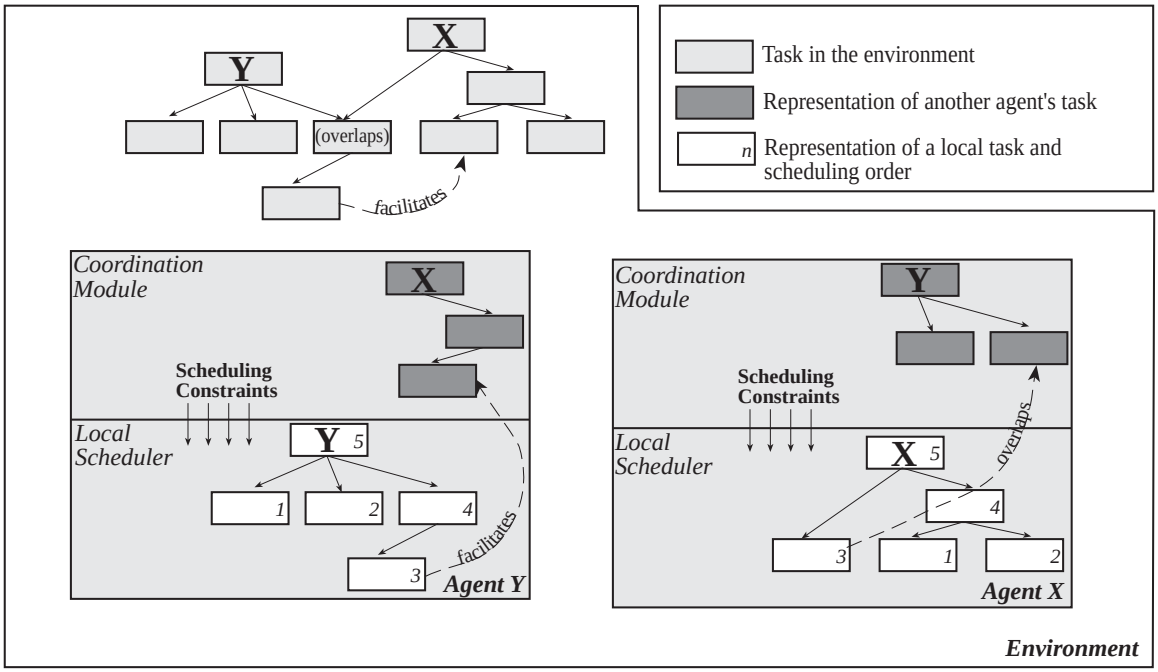


Figure 3: Conceptual view of the distributed coordination process

The primary interruption to this smooth flow of information is uncertainty. Uncertainty also flows from the environment (the open systems concept[22]), and through the above mechanisms to create local scheduling uncertainty. Less uncertainty in the environment means less uncertainty in the existence and extent of the CRs, less uncertainty in local scheduling, and therefore less complex coordination behavior (communication, negotiation, partial plans, etc) is needed (for example, one can have cooperation without communication [19]). A secondary problem is the lack of infinite computational resources/time, resulting in the necessity of satisficing, not optimal, behaviors. Even if agents had instant access to the complete global state of the entire system it does not mean that the environment provides sufficient computational resources or time to the agents with which to exploit that voluminous information[3].

For example, the abstract domain of distributed search contains several potentially uncertain coordination relationships [25]. The primary uncertainty lies in the subgoal relationship: how a particular task relates to the problem as a whole. Will this task be a part of a final solution? Are there multiple paths to the goal? How much effort will it take? A secondary uncertainty lies in the presence of a *facilitates* relationship: whether the amount of processing is greatly affected by the order in which goals are solved. Ignoring these two relationships will affect the amount of resources the agents use through undesirable redundant processing, idleness, inefficiency, and distraction. The *facilitates* relationship (along with *overlaps*) was a major contributor to the results achieved with partial global planning.

Informally, one task may *facilitate* another by allowing the second task to exploit (by decreasing duration, increasing quality, or both) a partial result. The *facilitates* relationship,

[3]Our description of the coordination process is consistent with social views of organizational coordination mechanisms or behaviors: the use of rules, regulations, and standards; the creation of supervisory and decision-making hierarchies; and specialization or departmentalization[32, 35]. Organizational structure can be viewed as part of the coordination algorithm.

therefore, has two parameters (called *power* parameters) $\phi_{\mathrm{d}}$ and $\phi_{\mathrm{q}}$, that indicate the effect on duration and quality respectively. The effect varies not only through the power parameters, but also through the quality of the *facilitating* task available when work on the *facilitated* task starts. Let $T_a$ *facilitate* $T_b$ with some duration power $\phi_{\mathrm{d}}$ (let $\phi_{\mathrm{q}} = 0$). If $T_a$ is completed with maximal quality, and the result is received before $T_b$ is started, then the duration $\mathbf{d}(T_b)$ will be decreased by a percentage equal to the duration power $\phi_{\mathrm{d}}$ of the *facilitates* relationship. If the result received is of less than maximal quality, $\phi_{\mathrm{d}}$ is reduced by the proportion $\frac{\text{quality received}}{\text{maximal quality}}$. Communication after the start of processing has no effect. A fuller treatment of the mathematical details can be found in [10].

The task environment structure is an abstract model of the agents' environment that can be used for analysis and simulation. The *objective* level describes the essential structure of a particular problem-solving situation or instance over time. It focuses on how task interrelationships dynamically affect the *quality* and *duration* of each task. Briefly, the basic model is that task groups $\mathcal{T}$ occur in the environment at some frequency, and induce tasks $T$ to be executed by the agents under study. Task groups are independent of one another, but tasks within a single task group have interrelationships. Each task group has a deadline $\mathcal{D}(\mathcal{T})$. The *quality* of the execution or result of each task influences the *quality* of the task group result $\mathcal{Q}(\mathcal{T})$ in a precise way [10]. These quantities, deadline and quality, can be used to evaluate the performance of a system.

An individual task that has no subtasks is called a method $M$ and is the smallest schedulable chunk of work (though some scheduling algorithms will allow some methods to be preempted, and some schedulers will schedule at multiple levels of abstraction). There may be more than one method to accomplish a task, and each method will take some amount of time or other resources and produce a result of a some *quality*. The term *quality* in the model summarizes several possible properties of actions or results in a real system: certainty, precision, and completeness of a result, for example[12]. Quality of an agent's performance on an individual task is a function of the timing and choice of agent actions ('local effects'), and possibly previous task executions ('non-local effects'). When local or non-local effects exist between tasks that are known by more than one agent, they become *coordination relationships*. These relationships include the following:

**Basic Domain Relationships:** These include relationships such as inhibits, cancels, constrains, facilitates, causes, enables, and parent task/subtask (from which many useful graph relations can be computed). These relations provide *task ordering* constraints, which can be represented by temporal relations on the tasks (see below). Partial global planning uses the *subtask* relationship: task B is a *subtask* of task A if B is required for some method of achieving A. It also uses the *facilitates* relationship: task A *facilitates* task 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.

**Graph Relationships:** Some generalized coordination relationships can be derived from the parent task/subtask graphical structure of tasks and subtasks, for example: overlaps, necessary, sufficient, extends, subsumes, and competes. Partial global planning uses the *overlaps* relationship: task A *overlaps* B if there exists a task G such that A is a parent task of G and B is a parent task of G.

**Temporal Relationships:** These depend on the timing of tasks—their start and finish times, estimates of these, and real and estimated durations. From Allen [1], these include before, equal, meets, overlaps, during, starts, finishes, and their inverses. Temporal relations may be preferences (soft constraints) as well as absolute relations. There are three levels of temporal constraints. *Hard* constraints cannot be violated. The inability to satisfy hard constraints means that the problem is overconstrained. *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* 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 PGP are soft constraints; generalized partial global planning allows any type of temporal constraint.

**Non-computational Resource Constraints:** A final type of relation is the use of physical, non-computational resources. This is the major coordination relationship in some domains, such as factory scheduling and office automation[37, 34]. Partial global planning requires information about the resources each agent has to collect data.

The original partial global planning algorithm requires only a fraction of these relationships, to be discussed in the next section. We have found several of these relationships to be useful in scheduling tasks for parallel execution as well [9]. The basic framework of the objective model is to formally specify how the execution and timing of tasks affect this measure of quality. See [10] for details and more formal definitions of the task environment model, and also Section 6.1.

## 4.1 Building an Augmented Goal Model for the DVMT

In generalizing the PGP algorithm for the DVMT, we will replace the PGP notion of i-goals with "augmented goals", patterned after out task environment model structures. What information is needed in an augmented goal? Our particular implementation of the augmented goals for the DVMT is based on the tasks in that domain—tracking vehicles moving through an area via their acoustic signatures. Most instantiations of the augmented goal structure model will include the following information in a goal:

- A specification of the desired result. This includes the desired and minimum solution characteristics. In the DVMT we use a specification of the desired *completeness*, *precision*, and *certainty* (see below).
- A specification of the current projected result. Some solution methods may only provide partial solutions. In the DVMT, nodes use different plans to produce different results, so the name of the current plan is sufficient to specify the current projected result.
- Physical resources needed, for computing non-computational resource constraints. The DVMT, for example, uses information about the locations and types of sensors a node has.

- Timing information. In modeling DVMT node activity using augmented goals, we include information about the projected start times, end times, and durations, as well as any associated deadlines.

We model the overall problem-solving responsibility of a DVMT node as a single "system goal" that contains information about the *capabilities* of a node (what type of sensors the node has and where they are located), the desired *completeness* (what types of vehicles the node is interested in and how much it is interested), required *precision* (how precise an answer is necessary — this may depend on the type of vehicle), and the *certainty* or minimum belief desired (which may also vary with vehicle type). Satisfaction of such a goal is not black-and-white, but is a continuum of degrees of satisfaction with different solutions.

As problem solving commences, each DVMT node builds a hierarchical, BB1-style control plan [20] that describes the set of concurrent tasks that the node is currently pursuing, for example, a node might be tracking two vehicles (using different methods) and looking out for any new vehicles (see Figure 4 and Figure 5). As the control plan is built, we model it by a series of control goals, which are augmented goals built hierarchically from the system goal. Control goals contain much of the same information as the system goal, but modified to their specific purpose and they do not carry *capability* information. In addition, they carry timing information (projected start time, end time, and durations) and information about any associated deadlines. They also carry information about what the projected result looks like (which may be different from the most satisfactory result). In some cases, control goals may also represent future goals of the system, whenever these can be ascertained. This structure, a hierarchically ordered set of augmented goals, is the model of current and future problem solving activity at a DVMT node.

The coordination relationships used by the generalized partial global planner are computed in this structure as follows:

**Subtask:** This information comes from the structure of the augmented goal tree itself. Since parts of the tree are communicated hierarchically, it is relatively straightforward in the DVMT to connect new communicated goals into the proper subtask relationship to old goals.

**Facilitates:** In the DVMT, one node may be tracking a vehicle, and the projected vehicle path enters another node's sensor area. The tracking goal of the first node then *facilitates* the second node's goal to find new vehicles. Another example, familiar from the original PGP, is when one node (perhaps because it has less data to work on) is able to process data more quickly and more carefully than another node, even though both have access to the data. The section of the augmented goal hierarchy in the first node *facilitates* the analogous section in the second node.

**Overlap:** In the DVMT, *overlaps* can be computed from the result specification in the goal.

Figure 4 shows how augmented goal structures and coordination relationships model activity at two DVMT nodes. Node 1 has two concurrent tasks: the first is to find new vehicles that may be picked up by its sensors, and the second is to track vehicle $V_1$ of type "X". Before the current state, Node 1 had discovered $V_1$, and had tracked it while determining what type of

vehicle it was, and then began tracking it as a "X". Node 2 also has two tasks: the first to find new vehicles that may appear to its sensors, and the second to track and identify a new vehicle (tentatively named $V_2$) that just appeared.

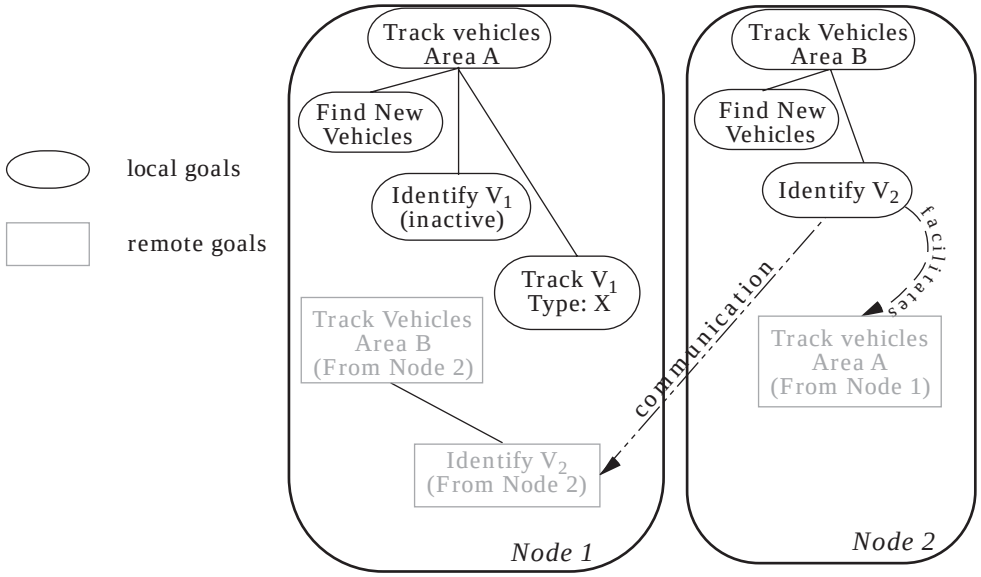


Figure 4: An example of using augmented goal structures and coordination relationships in the DVMT

The generalized partial global planning coordination algorithm (described in detail in Section 5) directs the nodes to communicate the top-level augmented goals. This single augmented goal forms the core of one node's model of the other. Each new local abstract goal is then compared to this model, and certain relationships will trigger further communication. In Figure 4, the new goal at node 2 (to identify $V_2$) *facilitates* the top-level goal of node 1 (which is node 2's model of activity at node 1). The calculation of the *facilitates* relation is done by projecting the vehicle path and comparing it to the sensor characteristics listed in the resource constraints of node 1's top-level goal. The generalized partial global planning algorithm states that this goal at node 2 should be communicated to node 1. Node 1 will add this goal to its model of node 2. The facilitative nature of the result in this case is a warning to node 1 that work will appear in the future, which node 1 can add to its future schedule of tasks. The facilitative relationship also effects node 2, which will commit to sending the result of the goal to node 1 at the time specified in the augmented goal. If something should go wrong (suddenly 10 new vehicles show up on node 2's sensors) node 2 will have to negotiate with node 1 if it must reschedule the earlier goal. Node 1 has not communicated any of its own subgoals to node 2 because it has discovered no relationships between its own subgoals and its model of node 2's activities that impact on coordination.

# 5 Generalized Partial Global Planning

The last item that we need to specify in our approach is the PGP algorithm itself. The original PGP algorithm orders intermediate goals according to their cost as computed from the cross-product of a vector of seven computable factors (such as redundancy, reliability, etc.) and global cooperation parameters that give a weight to the corresponding term in the calculation. In our

approach, this information has been captured symbolically by the coordination relationships between augmented goals. We could re-implement most of the original PGP algorithm by using a hill-climbing local scheduler, and assigning a priority according to the original PGP evaluation function — however, we are instead using a more sophisticated local scheduler that can understand more complex constraints such as delays and deadlines. The exact relationship of each of the seven factors used in the original PGP evaluation function to our new representations is as follows:

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

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

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

**Predictiveness:** Any goal with a duration of $x$ could 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 generalizes this concept to the *facilitates* relationship and considers it to be a domain-dependent relation. While we could use the original algorithm to compute the presence of this relationship, we in fact use a more complex algorithm that also takes into account future projections of vehicle tracks.

**Locally-predicted:** Was 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 local scheduler can use the hierarchical construction of the goal hierarchy to avoid reordering the steps in constructing a track.

**Independence:** Was how many goals occurred 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 local scheduler receives local ordering preferences, and so it knows what local orderings were necessary, which may be negotiated, and which are only preferences. We are not supplanting the local scheduler with a separate PGPlanning scheduler, thus this relationship is not directly needed.

**Diversity:** The diversity value of a goal was 0 if it did not derive redundant information, or if all goals following it derived redundant information. Otherwise, the diversity of a goal was measured by the minimum activity distance between the goal and the later non-redundant goals. The effect was to plan to do non-redundant work before redundant

work. The GPGP coordination mechanisms detect redundancy through coordination relationships instead.

Any domain-independent coordination algorithm, based on our model of agent activity, must describe: what subset of the coordination relationships will be used; when an augmented goal should be communicated, and to whom; when a goal-satisficing result should be communicated, and to whom; when a coordination relationship is detected, what is done about it.

In designing the generalized partial global planning algorithm, we used the original PGP algorithm to make decisions about these questions. However, we wanted to take advantage of new characteristics of our environment that did not exist when the original PGP was designed, such as a real-time scheduler and approximate data processing algorithms[12]. Therefore we detect some coordination relationships that could never have occurred in the original system, and handle them appropriately. Thus the resulting algorithm is not only "generalized" because it is domain-independent, but also because it has been extended to react to new goal interactions.

We also wanted to take advantage of the hierarchical structure of the agent's activity model and reduce the amount of communication between agents by making decisions with the smallest context possible. Rather than communicate by broadcasting all active goals, as in the original PGP, agents broadcast only the roots of their augmented goal trees, and communicate more detailed information as it is needed, as indicated by the appearance of new coordination relationships.

*What subset of the coordination relationships will be used?* The generalized PGP algorithm is based on the *subtask*, *facilitates*, *overlaps*, and the temporal relationships (see Section 4).

*When should an augmented goal be communicated, and to whom?* Top level goals ("system goals") are broadcast to all agents, in order to establish long term relationships and agent capabilities. Any augmented goal that *overlaps* or *facilitates* another agent's augmented goal should be sent to that agent. This notifies another agent of a potential interaction, based on the current perceived activities or role of the remote agent.

*When should a goal-satisfying result be communicated, and to whom?* If an agent has previously sent a goal to a remote agent because it detected a coordination relationship, and that relationship still holds, then the satisfying hypothesis should be sent to the remote agent.

*When a coordination relationship is detected, what is done about it?* If a remote goal is received that *facilitates*, but does not *overlap*, a local goal, then the remote goal represents future work on the facilitated local goal. The agent receiving this information may not even know that it has (will have) the facilitated local goal. The receiving agent should add a representation of the work required at the predicted time to the current local schedule, potentially delaying the facilitated local goal (see [11] for a discussion of how long to delay), and potentially changing the predicted duration or result quality of the goal. The remote agent who has the facilitating goal will commit to a communication deadline and minimum quality for the communicated result. If the remote goal both *facilitates* and *overlaps*, then there is redundant work occurring (because of the overlap) and the other agent can provide information to help the local agent (usually because it will complete the overlapping work sooner). The local agent should postpone work on the local goal until the remote agent sends its result. Finally, if a remote goal simply *overlaps* a local goal, then redundant work is occurring. If the remote goal is temporally *before* the local goal, the system can proceed as in the *overlaps and facilitates* case; otherwise, the system does

not have enough context with which to make a decision. The original PGP algorithm would have used the static organizational structure of the agents to break the tie; we would prefer to enter into a negotiation with the remote agent, deciding who should handle the overlap based on several factors: which agent is less busy (measured in excess time available during the overlap), which agent will use less resources (measured in time allocated to accomplishing this goal during the overlap), and which agent will provide the best result (measured by the amount of goal satisfaction provided by the planned result). Note that it is possible that *both* agents working together may achieve the best result. It is fine for agents to be redundant if it increases their joint certainty, for example[4]. This case also covers the way the original PGP passed tasks to idle nodes — the idle node broadcast that it was idle (in this case, the goal to do any work), which would *overlap* with many local goals, and the decision was made by appealing to the static organizational structure.

# 6 A Scenario

Our implementation of the generalized partial global planning algorithm for the DVMT is 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, as described above. We have done everything except implementing the overlapping negotiation protocol, because we want to evaluate several approaches to negotiation. The DVMT architecture used in the DVMT experiments is shown in Figure 5, except that we did *not* use the real-time scheduler (instead, we used the old priority-based scheduler). The experiments done with the simulator, described next in Section 6.1 *were* done with the real-time scheduler.

There were five PGP-like rules, implemented as control knowledge sources:

**Avoid useless redundancy with other nodes.** Whenever a goal overlap is detected between a local and external goal, then in many cases only one agent should continue work on the shared goal (the major exception being reliability, discussed next). Which node should continue to work on the goal can be decided from other PGP heuristics (like 'minimize durations' below), or from applicability to the system goals (the goal may be more useful to one node than the other).

**Build reliable solutions.** By this we mean be redundant to increase certainty in something. 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. PGP never really did this.

**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.

**Provide predictive information to other nodes.** Whether a goal is predictive with respect to another is one of the basic domain goal relationships.

**Perform regular problem solving.** Follow the local control plan.

[4]Note also that the excess time available at a node is not a hard constraint, because the agent may be able to rearrange its schedule to change how busy it appears.

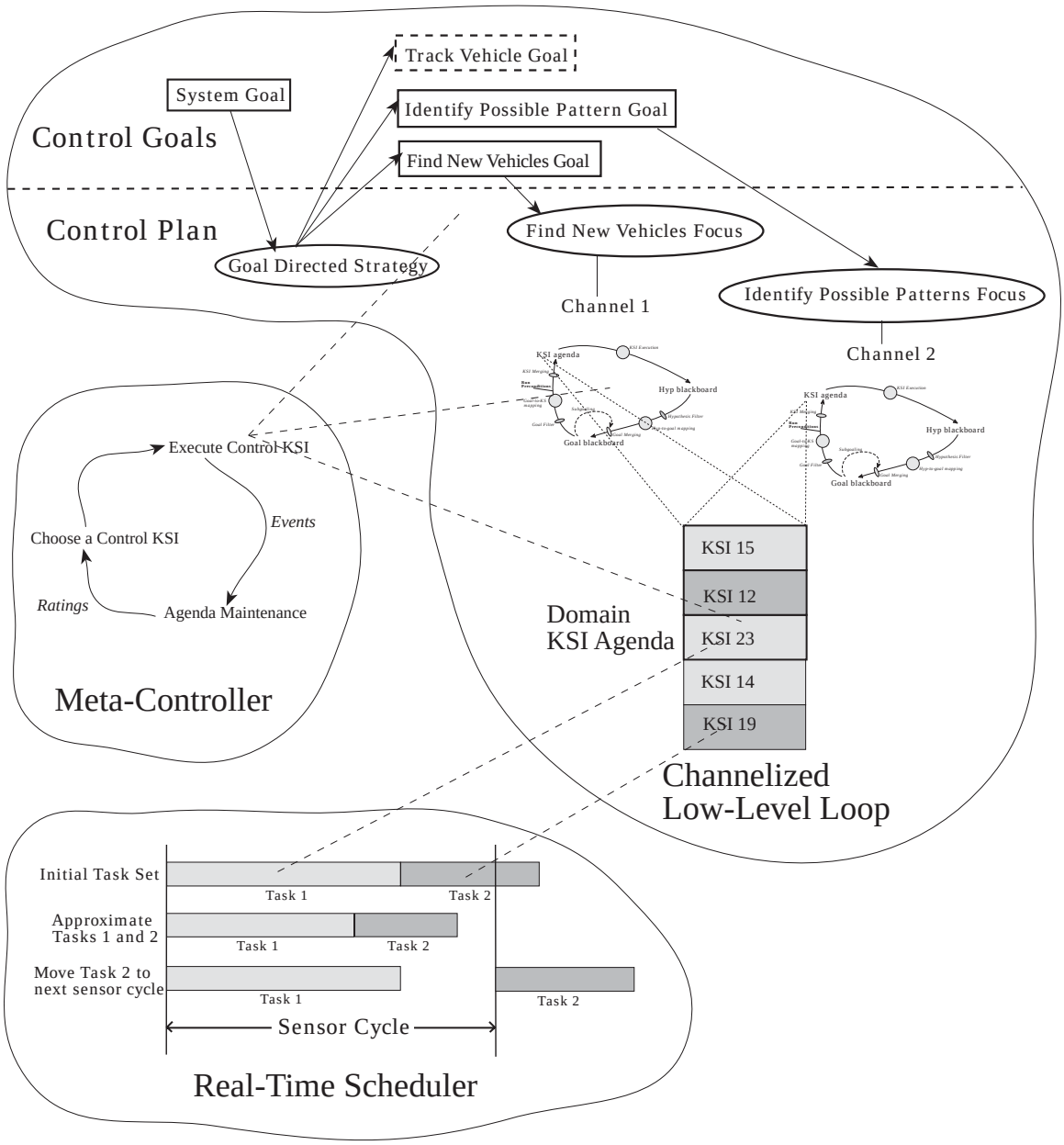


Figure 5: ***The real-time blackboard architecture. At the lowest level it consists of a parameterized low-level blackboard loop, guided by a BB1-style control plan and by control goals. Multiple copies of this loop (each known as a channel) are created to control each aspect of problem-solving that might require separate control. A meta-controller executes a control KSI loop that constructs the control plan and goals, and modifies the parameters of the low-level loop. A real-time scheduler ensures real-time performance by monitoring problem-solving at the channel-task level and fixing schedules that go over time. The scheduler constructs future schedules based on projected channel tasks and fills in those channel tasks with domain KSIs as they appear on the agenda. The dashed lines between modules indicate points of interaction.***

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.

To evaluate our algorithm in the DVMT domain, we constructed the scenario shown in Figure 6. The "vehicles" in the scenario are various aquatic creatures and waterfowl. One of the high-level system 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 4).

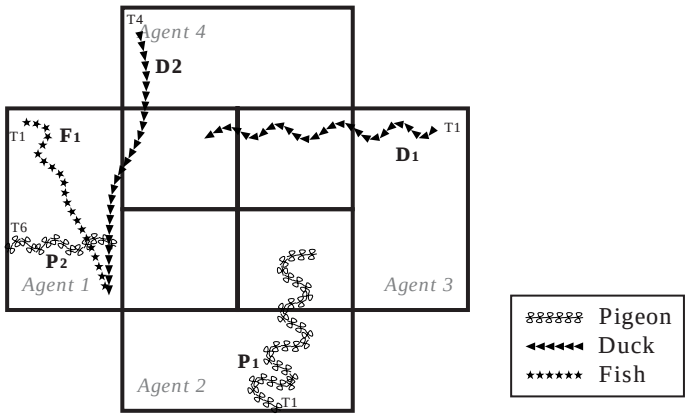


Figure 6: The CDPS scenario environment: Agents 1 and 3 are fish-protectors, Agents 2 and 4 are pigeon-trackers. Agent 3 has a noisy sensor.

The scenario demonstrates several issues that were not relevant to the original PGP and DVMT, and that were discussed in Section 3. Fish-protectors and pigeon-trackers are heterogeneous agents that have different utilities for the same data (see the discussion of *satisfaction* in Section 4). The hard time deadlines allow experiments with real-time performance and the use of approximate processing methods to give each dynamic agent a choice of different problem solving methods. The mapping from "real-world time" to processor time is left deliberately unspecified — experiments can be constructed by varying how tightly pressed for time the agents are. 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.

In our initial experiments, which did not include negotiation, we concentrated on the number of communications that occurred during this scenario. Table 1 shows the actual amount of communication that occurs at each node using the generalized PGP algorithm, and the amount of communication that would occur using the original PGP algorithm (computed by hand simulation).

These initial experiments show the reduction in communication achieved by using a hierarchical goal structure. Only communications between nodes were measured; because we hand simulated PGP we cannot state anything about the quality of the solution (but see the next section). They also show that it is possible to conceptualize the problem in the terms of

| | PGP | Generalized PGP |
|---|---|---|
| Node 1 | 13 | 10 |
| Node 2 | 17 | 9 |
| Node 3 | 14 | 9 |
| Node 4 | 13 | 9 |

Table 1: Node Communication at each sensor cycle: PGP vs. GPGP

generic coordination relationships. They do not show the effect of real-time, dynamic methods, heterogeneous agents, or negotiation, but the experiments described in the next section add the real-time local scheduler and dynamic choice from multiple methods.

## 6.1 Simulation

Rather than arrive at a version of the GPGP coordination algorithm that was optimized for the DVMT, we are currently fleshing out our conceptual model into a set of analytic components based on general characteristics of the problem solving environment, the task being performed, the agents, and quantitative properties of the coordination relationships (such as how likely they are to appear and how significant are their effects). To experimentally verify our formulations we are taking a tack somewhat between the analytical but perhaps too simplified approach of Malone [29] and the non-analytical but probably too specialized approach of the DVMT, by using a statistical simulation of a large class of CDPS environments.

The simulation is driven by a task environment model constructed as briefly discussed in Section 4. In the experiments below, there were two task classes. One class of tasks had a mean time between arrivals of 40% less than the other. Facilitation relationships are generated between tasks with a base probability that decreases linearly with the difference between task arrivals.

Each agent uses a highly sophisticated "design-to-time" (DTT) real-time local scheduler based on the concept of approximate processing[17, 12]. The DTT scheduler will choose a method for a task based on the amount of time available for that task and the other tasks currently on the agenda. The DTT scheduler may change the method being used during execution as a task monitoring point; in the experiments described here 50% of the work done before changing methods is lost. The DTT scheduler is boundedly rational for deadline constraints, and was modified to be boundedly rational for delay constraints. Each class of tasks had 5 methods of varying quality and duration; each task execution was monitored by the DTT scheduler three times. The actual duration and quality values for each method for each task are randomly generated from normal distributions with means equal to the estimated values and variances as specified for the method/task combination. The shared global utility function by which agents are judged is the total quality of their individual task solutions. The design-to-time scheduling algorithm used by each agent ensures that under normal circumstances (a required utilization of less than 3) less than 2% of the tasks ever miss a deadline. This is a property of the existence of low cost/low quality approximations.

Observations in the simulation experiments were made from average responses over 5 statistically generated runs of 1000 simulated world time units. The number of tasks that

actually are generated depends on their arrival rate, which was varied. We also varied the *a priori* likelihood of a *facilitates* relationship between two tasks of the same class and the significance of the effect it has on the time of the *facilitated* tasks (called the *power* of the CR and measured by the percent reduction in time facilitated).

Each run consists of two sets of 4 agents. Each of the two agent-sets receives exactly the same set of tasks at exactly the same times — one agent-set uses the original DTT scheduling algorithm and no communication, the other agent-set uses the DTT scheduling algorithm modified to be *delay*-boundedly rational and to *always* detect and communicate coordination relationships. Each of the four agents in each agent-set receives precisely the same set of tasks as its counterpart in the other agent-set. The coordinating agents do not enter into negotiation upon failures but rather simply break a failed commitment.

The experiments focused on two of the quantitative properties of *facilitates*: how likely it is to appear and how significant is its effect. The primary performance metric is the average percentage increase in quality (APQI) in each pair of agents that received the same task set (a paired response). The other variable that was manipulated was the mean time between arrivals of the tasks; we can then compare relative increases in quality based on the average utilization required by that set of tasks. When the average required utilization is greater than 1, it means that it is impossible to complete all tasks without approximating some of them.

There are three characterizations of this simulated environment that make it different from the actual environment of a system such as the DVMT:

- In interpretation environments like the DVMT, not all tasks need to be done, but in our simulation they do.
- Our simulation assumes that approximating a result has only a local effect on quality, as opposed to reducing the quality of a whole group of related tasks.
- Our simulation does not contain a model of the relationship between the fact that a task does or doesn't need to be done, and the fact that a *facilitates* relationship does or doesn't exist (goal compatibility).

On the other hand, our measurements are against a real-time scheduling algorithm that likewise does not take these factors into account. Finally, the addition of such structure to tasks begs the addition of new coordination relationships (especially *subtask*).

Our initial results [11] are based on the detection of, and coordination by, the *facilitates* coordination relationship only. They show that the effect of detecting this CR on overall performance goes up with both the strength-of-effect and *a priori* likelihood of the CR, and that coordination can have a negative effect when the strength-of-effect is low (see Figure 7). Our results also show that coordination via *facilitates* is more important when agents are under time pressure.

# 7 Conclusions

This paper presented an approach to the design of distributed coordination strategies using a set of domain-independent coordination relationships and augmented goal structures. The

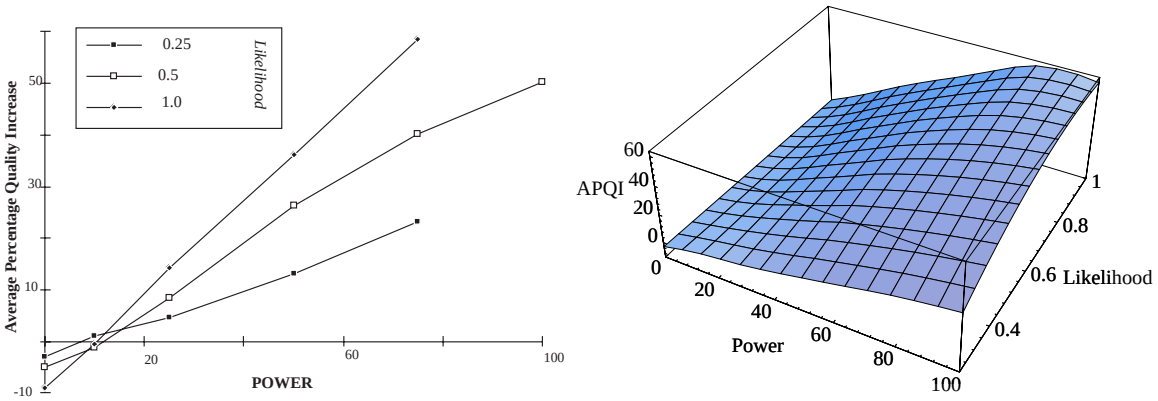


Figure 7: The effect of the strength of the *facilitates* relationship (**power**) on the average percentage quality increase (**APQI**) at different *a priori* likelihoods of the presence of the *facilitates* relationship (cubic interpolation).

contents of the goal structures and the methods used to compute the relationships are domain-dependent. They represent the activities of distributed agents and the subproblem interactions between them. The model was realized in the DVMT. A domain-independent distributed coordination strategy, called generalized partial global planning, was shown based on the original PGP algorithm, but was extended to reduce the amount of communication required. The strategy is currently being extended to handle a more complex environment, and to take advantage of more complex problem-solving methods. Experiments are currently being conducted to examine the effect of each coordination rule on each agent's computational load, communications load, and problem-solving results[11].

Each coordination rule should be considered a "temporarily settled question," subject to being reopened in the domain setting [18]. This idea leads to extending a coordination strategy so that agents can negotiate about the strategy itself. For instance, in a real-time situation, an agent with a predicted bottleneck might restrict messages from other agents during the bottleneck.

One of the most basic questions that one can ask about the behavior of an intelligent agent, or the behavior of a collection of agents, is the effect that the environment, task, and architecture have on their individual or collective behavior. We would like to explore the nature of the interactions between the problem solver, the problem being solved, and the environment. We could re-run all the previous PGP experiments on GPGP, but for what purpose? Since GPGP is designed to be applicable in a wide range of environments and task domains, we believe it is more useful to try to build a model of the effects of coordination relationship that arise and the coordination behaviors the algorithm produces. Such a model will enable us to predict how the algorithm would perform in situations different from the DVMT, and also help explain the situations under which it gives poor performance. In order to build predictive models of how to best coordinate in an environment, we need to decide how to characterize the environment with respect to the behaviors in which we are interested [4]. This process includes both deciding what aspects need to be represented, and to what accuracy. We are performing such an environmental assessment for what we contend is a large class of agent architectures

and tasks, using a simulation model. This simulation model is more complex than an analytical queueing theory based model [29] because it allows for non-independent tasks with multiple possible solution methods, and heterogeneous agent characterizations.

Finally, we are examining formal characterizations of local agent behavior, such as local scheduling algorithms. These characterizations include both standard conceptions of rational scheduling behavior and boundedly rational scheduling behavior where computer or human agents may have to satisfice and deal with non-local uncertainties in local scheduling constraints.

## Acknowledgments

We would like to thank Marty Humphrey for his work on the massive architectural changes in the support of the new DVMT environment, and running experiments, and Alan Garvey for support of the real-time aspects of the DVMT.

## References

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

[2] Mark Boddy and Thomas Dean. Solving time-dependent planning problems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, August 1989.

[3] C. A. Broverman, K. E. Huff, and V. R. Lesser. The role of plan recognition in design of an intelligent user interface. In *Proceedings of the IEEE Systems, Man, and Cybernetics Conference*, pages 863–868, Atlanta, Georgia, 1987.

[4] Paul R. Cohen. A survey of the eighth national conference on artificial intelligence: Pulling together or pulling apart? *AI Magazine*, 12(1):16–41, Spring 1991.

[5] S. E. Conry, K. Kuwabara, V. R. Lesser, and R. A. Meyer. Multistage negotiation for distributed constraint satisfaction. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6), November 1991.

[6] Daniel D. Corkill and Victor R. Lesser. The use of meta-level control for coordination in a distributed problem solving network. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 748–755, August 1983.

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

[8] Keith Decker, Alan Garvey, Marty Humphrey, and Victor Lesser. A blackboard system for real-time control of approximate processing. In *Proceedings of the 25th Hawaii International Conference on System Sciences*, January 1992. Extended version to appear in the International Journal of Pattern Recognition and Artificial Intelligence 7(2) 1993.

[9] Keith S. Decker, Alan J. Garvey, Marty A. Humphrey, and Victor R. Lesser. Effects of parallelism on blackboard system scheduling. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 15–21, Sydney, Australia, August 1991. Extended version to appear in the International Journal of Pattern Recognition and Artificial Intelligence 7(2) 1993.

[10] Keith S. Decker, Alan J. Garvey, Victor R. Lesser, and Marty A. Humphrey. An approach to modeling environment and task characteristics for coordination. In Charles J. Petrie, Jr., editor, *Enterprise Integration Modeling: Proceedings of the First International Conference*. MIT Press, 1992.

[11] Keith S. Decker and Victor R. Lesser. Analyzing a quantitative coordination relationship. *Group Decision and Negotiation*, 2(3):195–217, 1993.

[12] 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.

[13] E. Durfee and V. Lesser. Predictability vs. responsiveness: Coordinating problem solvers in dynamic domains. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 66–71, August 1988.

[14] 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):647–662, September 1988.

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

[16] E.H. Durfee and V.R. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167–1183, September 1991.

[17] Alan Garvey and Victor Lesser. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6), 1993. Special Issue on Scheduling, Planning, and Control.

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

[19] M. R. Genesereth, M. L. Ginsberg, and J. S. Rosenschein. Cooperation without communication. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 51–57, Philadelphia, PA., August 1986.

[20] Barbara Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26:251–321, 1985.

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

[22] Carl Hewitt. Open information systems semantics for distributed artificial intelligence. *Artificial Intelligence*, 47(1):79–106, 1991.

[23] Craig A. Knoblock. Search reduction in hierarchical problem solving. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, Anaheim, July 1991.

[24] 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.

[25] V. R. Lesser. A retrospective view of FA/C distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1347–1363, November 1991.

[26] Victor R. Lesser and Daniel D. Corkill. Functionally accurate, cooperative distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):81–96, January 1981.

[27] Victor R. Lesser and Daniel D. Corkill. The distributed vehicle monitoring testbed. *AI Magazine*, 4(3):63–109, Fall 1983.

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

[29] Thomas W. Malone. Modeling coordination in organizations and markets. *Management Science*, 33:1317–1332, 1987.

[30] Thomas W. Malone and Kevin Crowston. Toward an interdisciplinary theory of coordination. Center for Coordination Science Technical Report 120, MIT Sloan School of Management, 1991.

[31] Sergei Nirenburg and Victor Lesser. Providing intelligent assistance in distributed office environments. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 590–598. Morgan Kaufmann, 1988.

[32] Charles Perrow. *Complex Organizations*. Random House, New York, 1986.

[33] 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.

[34] 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.

[35] W. Richard Scott. *Organizations: Rational, Natural, and Open Systems*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.

[36] 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.

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

[38] Frank v. Martial. A conversation model for resolving conflicts among distributed office activities. In *Proceedings of the Fifth Conference on Office Information Systems*, pages 99–108, Cambridge, MA, April 1990.